MOVIE TICKET BOOKING SYSTEM

PROJECT REPORT

Submitted by

DHARANIPATHY R

Register No: 31022P08002

in partial fulfillment for the award of the degree

of

MASTER OF COMPUTER APPLICATIONS

in

PG DEPARTMENT OF COMPUTER APPLICATIONS

GOVERNMENT THIRUMAGAL MILLS COLLEGE

GUDIYATTAM, VELLORE DIST - 632 602

MARCH/APRIL – 2024

GOVERNMENT THIRUMAGAL MILLS COLLEGE, GUDIYATTAM.

VELLORE DIST – 632 602

(Affiliated to Thiruvalluvar University)

BONAFIDE CERTIFCATE

Certified that this project titled "**MOVIE TICKET BOOKING SYSTEM"** is the bonafide of work done by **DHARANIPATHY R (REG NO: 31022P08002)** to **Government Thirumagal Mills College, Gudiyattam** in partial fulfillment of the requirement for the award of the degree of **Master of Computer Applications** is a record of bonafide work carried out by her under my guidance. The project requirement as per the regulations of this institute and it meets the necessary standards for submission.

Mrs. B. MANIVANNAN
(Project Guide)

Dr. K. ARULANANDAM
(Head of the Department)

Submitted for the IV – semester examination project work held on ……...

---

1. External Examiner

---

2. External Examin

DECLARATION

I affirm that the project work titled **MOVIE TICKET BOOKING SYSTEM** being submitted in partial fulfillment for the award of **Master of Computer Applications** is the original work carried out by me. It has not formed the part of any other project work submitted for award of any degree or diploma, either in this or any other University.

DHARANIPATHY R
31022P08002

I certify that the declaration made above by the candidate is true

Mrs. B. MANIVANNAN MCA., M. Phil., B.Ed., (SET)

ACKNOWLEDGEMENT

I express my gratitude to our beloved Principal **Dr. G. KRISHNAN M.A., M.Phil., M.ED, Ph.D.,** Government Thirumagal Mills College Gudiyattam, for providing the resource facilities and encouraging gesture for completion of my project.

I wish to express my deep sense of gratitude and heartiest thanks to our professor **Dr. K. ARULANANDAM, MCA., M.Phil., Ph.D.,** Head of the Department of MCA, Government Thirumagal Mills College, Gudiyattam.

I am much privileged to have **Mrs. B. MANIVANNAN MCA., M. Phil., B.Ed., (SET)** lecturer in MCA Department, as my guide for this project. I heartily thank him for his overall help and support at all stage.

I also take an opportunity to thank all the TEACHING and NON - TEACHING STAFF for their fullest and valuable assistance provided in this project work.

Last but not the least, I own my special thanks to my beloved all my friends and family member for their help in finishing this project successfully.

Place: Gudiyattam Submitted by

Date: **DHARANIPATHY R**

TABLE OF CONTENTS :

ABSTRACT:

In the contemporary digital age, the convenience of online services has become integral to various aspects of daily life, including entertainment. The Movie Ticket Booking System is a web-based application designed to streamline the process of booking movie tickets, enhancing the overall cinema experience for users. Developed using JavaServer Pages (JSP), HTML, and CSS within the NetBeans IDE, this system offers an intuitive and user-friendly interface for both customers and administrators.

The system allows users to browse through a comprehensive list of movies currently showing, along with detailed information such as synopsis, genre, showtimes, and available seats. Through an interactive and visually appealing interface, users can select their preferred movie, showtime, and seating arrangement. Real-time seat availability is displayed to ensure seamless booking.

Furthermore, the system incorporates user authentication mechanisms, ensuring secure access to personal information and transactions. Users can create accounts, manage their profiles, view booking history, and make payments securely through integrated payment gateways.

Administrators have access to an administrative panel where they can manage movie listings, theater information, seat availability, and user accounts. Additionally, they can generate reports to analyze booking trends and optimize theater operations.

The implementation of responsive design principles ensures compatibility across various devices, enabling users to access the system conveniently from desktops, laptops, tablets, and smartphones. The use of HTML and CSS facilitates the creation of visually appealing layouts, enhancing the overall user experience.

In summary, the Movie Ticket Booking System offers a comprehensive solution for both users and administrators, leveraging modern web technologies to simplify the movie booking process and elevate the cinema experience to new heights.

CHAPTER 1
INTRODUCTION

INTRODUCTION:

Movie Ticket Booking System is a Java JSP and MySQL web based project to manage various aspects of Seats, Customer, Shows, Booking, and Payment. Its primary objective is to streamline administrative tasks and reduce the need for manual intervention in managing and overseeing the operations of a Movie Ticket Booking System. With its user-friendly interface, the system grants administrators exclusive access to monitor and supervise the entire process of the Movie Ticket Booking System. Comprehensive projects with complete source code for the Java JSP-based Movie Ticket Booking System can be found, offering step-by-step guidance on Java JSP Projects. Free source code for the Movie Ticket Booking System in Java JSP is also available for download.

This project Movie Ticket Booking System, is a Java JSP and MySQL Project which runs on the tomcat server, you can also run this project in Eclipse and Netbeans. We have developed this Java JSP and MySQL Project on Movie Ticket Booking System for automating the process of Movie Ticket Booking System. The main features of this project is to manage Seats, Shows, Booking, Payment, Customer and User. This is a Major Project In Java, which is suitable for students who are looking for final year Java projects, the main modules of the project is Movie module, Category module, Seats module, Shows module and Seats module, which performs all the operation in their respective domains.

Java Project on Movie Ticket Booking System is secured web application which run inside the JVM. You can download java project on Movie Ticket Booking System from our web- site. Also we provide Movie Ticket Booking System Project report, PPT and synopsis along with project. If you are looking for Spring, EJB, hibernate Movie Ticket Booking System project then we can develop it also according to your requirements. This is a web application projects in java free project download, which you can run easily on any of web browser. Movie Ticket Booking System is a simple java projects with source code for beginners from which students can learn,develop a good projects in Java. We also provide major java projects for final year and mini java projects for semester project.

OBJECTIVES:

Developed using Java JSP and MySQL, the Movie Ticket Booking System seeks to eliminate manual errors by establishing a computerized framework for managing a Movie Ticket Booking System. This system empowers users to seamlessly handle operations related to Movies, Category, User, Seats, Customer, and Shows in a centralized and organized manner. With its intuitive interface and robust functionalities, the Movie Ticket Booking System serves as an optimal solution to efficiently manage Customer, Shows, ensuring a smooth experience for administrators and users within the Movie Ticket Booking System.

The Movie Ticket Booking System represents innovative and practical Java JSP and MySQL projects suitable for final-year endeavours. These Java JSP Projects pertaining to the Movie Ticket Booking System are meticulously designed to provide hands-on experience in developing a resilient Movie Ticket Booking System. Designed with user-friendly interfaces and advanced capabilities.

Enhance your final-year project through our expert guidance. Gain access to the source code and database of our Movie Ticket Booking System to expedite your project development. By acquiring the source code and database, you gain insights into the implementation of crucial features related to Seats, Shows, Booking, and Payment. In search of free, high-quality Movie Ticket Booking System Projects? Our selection includes an array of free projects accompanied by detailed documentation and source code of Movie Ticket Booking System.

Discover our collection and select projects aligned with your requirements. Our Movie Ticket Booking System projects feature comprehensive documentation and synopsis, furnishing step-by-step instructions on the system's operations, functionalities, and advantages. The documentation and synopsis offer insights into the project's architecture, modules, and potential for customization to meet specific project demands.

CHAPTER 2

SYSTEM ANALYSIS

2.1 Proposed system:

Designing a movie ticket booking system using JSP (JavaServer Pages) involves several components and functionalities. Here's a proposed system outline along with its key features:

User Interface (UI):

- Homepage: Displays current movies, upcoming releases, and special events.
- Movie Listing: Allows users to browse movies, view details, and select showtimes.
- Seat Selection: Provides an interactive seating map for users to choose their seats.
- Booking Form: Collects user information, including name, email, and payment details.
- Confirmation Page: Displays booking details and provides a confirmation number.

User Authentication:

- User Registration: Allows new users to create accounts.
- Login: Existing users can log in to access their account and view booking history.
- Movie Management:
- Admin Dashboard: Allows administrators to add, edit, or remove movies and showtimes.
- Movie Details: Admins can update movie information, such as title, description, and
- duration.

Booking Management:

- Reservation Handling: Tracks seat availability and manages bookings.
- Payment Integration: Supports secure payment processing using payment gateways.

- Booking Confirmation: Sends email confirmations to users after successful bookings.

Search and Filtering:

- Search Bar: Allows users to search for movies by title, genre, or release date.
- Filtering Options: Users can filter movies based on various criteria like genre,
- language, or rating.

Notification System:

- Email Notifications: Sends reminders for upcoming bookings and notifications for special events.
- SMS Alerts (Optional): Optionally, integrate SMS notifications for users.

Reporting and Analytics:

- Booking Reports: Generates reports for daily, weekly, or monthly bookings.
- Analytics Dashboard: Provides insights into user behavior, popular movies, and peak booking times.

Security:

- Secure Authentication: Uses encryption for user passwords and sensitive data.
- HTTPS: Ensures secure communication between the client and server.
- Input Validation: Validates user inputs to prevent SQL injection and XSS attacks.

Performance Optimization:

- Caching: Implements caching mechanisms to improve system performance.
- Load Balancing: Distributes traffic across multiple servers to handle high loads.
- Database Indexing: Optimizes database queries for faster retrieval of data.

Scalability:

- Scalable Architecture: Designs a modular and scalable architecture to accommodate future growth.
- Cloud Deployment: Utilizes cloud services for easy scalability and resource management.

Accessibility and Internationalization:

- Accessibility Features: Ensures the system is accessible to users with disabilities.
- Multi-language Support: Supports multiple languages for a diverse user base.

Feedback and Support:

- Feedback Form: Allows users to provide feedback and suggestions.
- Help Desk: Provides customer support for inquiries, issues, and assistance.

This proposed system provides a comprehensive solution for movie ticket booking, catering to both users and administrators. Implementation involves integrating these features using JSP along with backend technologies like Servlets, JDBC for database connectivity, and possibly frameworks like Spring MVC for robust development. Additionally, consider using HTML/CSS for frontend development and JavaScript for client-side interactivity.

2.2 FEASIBILITY STUDY:

A feasibility study for a movie ticket booking system using JSP (JavaServer Pages) involves assessing various aspects to determine whether the project is viable and worth pursuing. Here's how you can conduct a feasibility study for such a system:

Technical Feasibility:

- Skill Set: Evaluate whether your team possesses the necessary skills in JSP, Servlets, and related technologies to develop the system.
- Technology Stack: Assess the availability and suitability of the required technologies, frameworks, and databases for implementing the system.
- Integration: Determine if necessary integrations with payment gateways, databases, and external APIs are feasible and readily available.

Market Feasibility:

- Market Analysis: Conduct market research to understand the demand for online movie ticket booking systems in your target region.
- Competitive Analysis: Identify existing competitors and analyze their offerings, strengths, weaknesses, and market share.
- User Demand: Assess whether there is sufficient demand from users for the convenience of booking movie tickets online.

Financial Feasibility:

- Cost Estimation: Estimate the costs involved in developing, deploying, and maintaining the system, including infrastructure, development resources, and licensing fees.
- Revenue Projection: Calculate potential revenue streams, such as ticket sales, advertisement placements, and partnerships.
- Return on Investment (ROI): Evaluate the projected ROI based on estimated costs and revenue streams over a specific period.

Operational Feasibility:

- User Acceptance: Determine whether users, including moviegoers and theater operators, are likely to accept and adopt the online booking system.
- Operational Impact: Assess how the system will integrate with existing processes and workflows at movie theaters and ticketing agencies.
- Scalability: Evaluate whether the system can scale to handle increasing user demand and transaction volumes over time.

Legal and Regulatory Feasibility:

- Compliance: Ensure that the system complies with relevant laws and regulations governing online transactions, data privacy, and consumer rights.
- Intellectual Property: Assess potential legal issues related to trademarks, copyrights, and licensing agreements for movie content and booking services.

Schedule Feasibility:

- Project Timeline: Develop a realistic timeline for the project, considering the complexity of features, resource availability, and potential setbacks.

- Milestones: Define key milestones and checkpoints to track progress and ensure timely delivery of the system.
- Dependencies: Identify dependencies on external factors such as third-party APIs, regulatory approvals, and vendor contracts.

Based on the findings of the feasibility study, you can make an informed decision about proceeding with the development of the movie ticket booking system using JSP. If the study indicates positive feasibility across technical, market, financial, operational, legal, regulatory, and schedule aspects, then the project may be deemed viable and worth pursuing.

CHAPTER 3

SYSTEM SPECIFICATIONS

3.1 HARDWARE REQUIREMENTS:

This section lists the requirements that are needed to run the system efficiently. The operating system needed for the system to run effectively, the interface to run application, the integrated development environment to develop the environment and the other tools used for editing purpose areas follow:

- Operating System : Windows 10
- Database : MySQL.
- Platform Used : JSP
- Editor Used : NETBEANS
- Browser Supported : Google Chrome, Internet Explorer, Safari, Microsoft Edge.

3.2 SOFTWARE REQUIREMENTS:

To develop a movie ticket booking system using JSP (JavaServer Pages), you'll need a set of software tools and technologies to support the development, testing, and deployment of the application. Here's a list of essential software requirements:

Java Development Kit (JDK) - Required for developing Java-based web applications like JSP. Includes Java compiler, runtime environment, and libraries.

Integrated Development Environment (IDE) - Choose an IDE that supports Java web development for ease of coding, debugging, and testing. Popular options include Eclipse, IntelliJ IDEA, and NetBeans.

Web Server - Apache Tomcat or another Java EE-compliant web server for hosting JSP pages and serving dynamic web content. Ensure compatibility with your chosen IDE for seamless deployment and debugging.

Database Management System (DBMS) - Choose a relational database system for storing movie, user, booking, and other related data.Options include MySQL, PostgreSQL, Oracle Database, or Microsoft SQL Server.Consider factors like scalability, performance, and licensing costs.

Servlet Container - Servlet containers like Apache Tomcat are required to run Java Servlets, which complement JSP in handling requests and generating dynamic content. Ensure compatibility with your chosen web server and IDE.

Frontend Technologies - HTML, CSS, and JavaScript for designing and implementing the user interface. Consider using frontend frameworks like Bootstrap or Materialize for responsive design and layout.

Java EE Technologies - Java Servlets: Handle HTTP requests and responses, manage session data, and interact with backend components. JSP (JavaServer Pages): Dynamically generate HTML content to render web pages with embedded Java code.

Java Database Connectivity (JDBC) - JDBC drivers for connecting to and interacting with the chosen database management system. Provides APIs for executing SQL queries, retrieving results, and managing database transactions.

Version Control System (VCS) - Git, Subversion (SVN), or another VCS for tracking changes to source code and collaborating with team members. Host repositories on platforms like GitHub, GitLab, or Bitbucket for centralized management and collaboration.

Build Automation Tool - Apache Maven or Gradle for automating the build process, managing dependencies, and generating deployment artifacts. Simplifies project configuration and ensures consistent builds across environments.

Testing Frameworks - JUnit for unit testing Java classes and components. Selenium or another web testing framework for automated UI testing. Consider integration testing frameworks like Cucumber for behavior-driven development (BDD).

Deployment Tools - Tools for deploying the application to production or staging environments. Containerization platforms like Docker for packaging and deploying microservices. Continuous integration and continuous deployment (CI/CD) tools like Jenkins for automating build, test, and deployment pipelines.

Monitoring and Logging Tools - Monitoring tools like Prometheus or Grafana for

tracking system performance and resource usage. Logging frameworks like Log4j or SLF4J for capturing and analyzing application logs.

Ensure that all software components are compatible with each other and meet the requirements of your movie ticket booking system. Additionally, keep in mind factors such as licensing costs, community support, and scalability when selecting software tools for your project.

CHAPTER 4

SYSTEM ARCHITECTURE

4.1 SYSTEM ARCHITECTURE:

Presentation Layer (View):

JSP Pages: These pages will be responsible for rendering the user interface. They will contain HTML along with embedded Java code (scriptlets), JSP tags, and expressions. CSS and JavaScript: Used for styling and client-side interactions respectively.

Controller Layer:

Servlets: These act as controllers in the MVC (Model-View-Controller) architecture. They receive requests from the client, process them, invoke appropriate business logic, and finally, forward the request to the appropriate JSP page for rendering.

Business Logic Layer (Model):

Java Classes: These classes contain the business logic of the application, such as handling user authentication, managing movie listings, managing booking details, etc. Data Access Objects (DAOs): Responsible for handling interactions with the database. They abstract the database operations and provide a clean interface for the business logic layer to interact with the database.

Data Access Layer:

Database: Stores data related to movies, users, bookings, etc. You can use a relational database like MySQL, PostgreSQL, or an object-relational mapping (ORM) tool like Hibernate to interact with the database.

Integration Layer:

JDBC (Java Database Connectivity): Used to connect to the database and perform CRUD (Create, Read, Update, Delete) operations. ORM Framework (Optional): If you choose to use an ORM framework like Hibernate, it will handle the mapping between Java objects and database tables, making database interactions easier and more object-oriented.

External Services :

Payment Gateway Integration: If you want to enable online payments, you may integrate with a payment gateway service like PayPal, Stripe, etc.Email/SMS Notification: Integration with email or SMS services to send booking confirmations, reminders, etc.

Security Layer:

Authentication and Authorization: Implement mechanisms for user authentication (login) and authorization (access control) to ensure that only authenticated users can perform certain actions. Input Validation: Validate user inputs to prevent common security vulnerabilities like SQL injection, XSS (Cross-Site Scripting), etc.

Logging and Monitoring:

Logging Framework: Integrate a logging framework like Log4j or java.util.logging to log important events and errors for troubleshooting and monitoring purposes.

Testing:

Unit Testing: Write unit tests to test individual components like servlets, business logic classes, etc. Integration Testing: Test the integration between different layers of the application. User Acceptance Testing (UAT): Test the application with real users to ensure it meets their requirements and expectations.

4.2 SETUP INSTRUCTIONS:

Step 1: Setup Environment

- Install NetBeans IDE if you haven't already.
- Ensure that Java Development Kit (JDK) is installed on your system.

Step 2: Create a New Web Application Project

- Open NetBeans IDE.
- Go to File > New Project.
- Select Java Web category and Web Application project type.

- Click Next and provide a name for your project (e.g., Movie Ticket Booking System).
- Click Finish to create the project.

Step 3: Design Database

- Design the database schema for storing movie details, user information, booking details, etc.
- Create necessary tables (e.g., Movies, Users, Bookings) using a database management tool like MySQL Workbench or through NetBeans IDE itself.

Step 4: Create JSP Pages

- Right-click on the Web Pages folder in your project.
- Select New > JSP... and create JSP pages for various functionalities like:
- Login (login.jsp)
- Registration (register.jsp)
- Movie Listing (movies.jsp)
- Booking (booking.jsp)
- Confirmation (confirmation.jsp)
- Error Handling (error.jsp, optional)

Step 5: Implement Servlets

- Right-click on the Source Packages folder in your project.
- Select New > Servlet... and create servlets for handling user actions and business logic.
- Example servlets: LoginServlet, RegistrationServlet, MovieServlet, BookingServlet, ConfirmationServlet.
- Implement doGet() and doPost() methods in each servlet to handle corresponding HTTP requests.

Step 6: Implement Business Logic

- Create Java classes for business logic such as user authentication, database operations, etc.
- Implement methods for user registration, login authentication, fetching movie details, booking tickets, etc.

Step 7: Integrate Database

- Establish database connection within servlets or business logic classes.
- Use JDBC or ORM frameworks like Hibernate to perform database operations (CRUD).

Step 8: Implement Frontend Logic

- Write HTML markup and JSP tags in JSP pages to create the user interface.
- Use CSS for styling and JavaScript for client-side validation and interaction.

Step 9: Handle User Sessions

- Implement session management to maintain user sessions across multiple pages.
- Use HttpSession object to store and retrieve session attributes like user login status, shopping cart items, etc.

Step 10: Test and Debug

- Test your application thoroughly for functional and non-functional requirements.
- Debug any issues encountered during testing.

Step 11: Deploy and Run

- Deploy your application on a servlet container like Apache Tomcat.
- Run the application and test it in a web browser.
- Ensure that it works as expected and handles various scenarios gracefully.
- By following these steps, you'll be able to create a basic movie ticket booking system using JSP in NetBeans. Remember to continuously refine and improve your application based on feedback and changing requirements.

4.3 FRONTEND DEVELOPMENT

Frontend development for a movie ticket booking system using JSP involves creating JSP pages that generate HTML and CSS content to render the user interface. Additionally, JavaScript can be used to enhance user interactions and provide dynamic behaviour to the web application. Let's dive into each aspect in detail:

JSP Pages:

JSP (Java Server Pages) allows embedding Java code within HTML to dynamically generate content on the server-side. Here's how you can structure JSP pages for the movie ticket booking system:

Login Page (login.jsp):

- Contains a form for user login.
- Validates user input on the client-side (using JavaScript) and server-side (in servlet).
- Includes links for user registration and password recovery.

Registration Page (register.jsp):

- Presents a form for user registration.
- Validates user input (e.g., username availability, password strength) using JavaScript.
- Submits registration data to the server for processing.

Movie Listing Page (movies.jsp):

- Displays a list of available movies along with their details (title, genre, release date, etc.).
- Allows users to select a movie and proceed to booking.

Booking Page (booking.jsp):

- Shows movie details selected by the user.
- Offers options for selecting seats, date, and time.
- Validates user input (e.g., seat availability) using JavaScript.
- Provides a button to confirm the booking.

Confirmation Page (confirmation.jsp):

- Confirms successful booking and displays booking details (movie name, seats, date, time, etc.).
- May include a button to go back to the movie listing page or perform other actions.

Error Handling Page (error.jsp):

- Displays error messages in case of any unexpected errors.
- Redirects users to the appropriate page after displaying the error message.
- HTML and CSS:
- HTML (Hypertext Markup Language) is used to structure the content of the web pages, while CSS (Cascading Style Sheets) is used for styling and layout. Here's how you can utilize HTML and CSS.

HTML Structure:

- Use semantic HTML elements to structure the content logically (e.g., <header>, <nav>, <main>, <footer>).
- Organize forms, tables, lists, and other elements to provide a user-friendly interface.

CSS Styling:

- Apply CSS rules to style HTML elements (e.g., fonts, colors, margins, padding, borders).
- Use CSS frameworks (e.g., Bootstrap) or custom stylesheets to achieve a consistent and visually appealing design. Ensure responsiveness by using media queries to adapt the layout to different screen sizes (e.g., desktop, tablet, mobile).

JavaScript:

- JavaScript adds interactivity and dynamic behavior to web pages. Here's how you can use JavaScript in the movie ticket booking system.

Form Validation:

- Validate user inputs in forms to ensure they meet certain criteria (e.g., required fields, valid email address, strong password).
- Provide instant feedback to users by displaying error messages or highlighting invalid fields.

Dynamic Content Updates:

- Update the content of the page dynamically without reloading the entire page (e.g., updating available seats based on user selection).

Event Handling:

- Handle user interactions such as button clicks, form submissions, mouse movements, etc.
- Implement event listeners to respond to user actions and trigger appropriate actions.
- AJAX Requests:

- Use AJAX (Asynchronous JavaScript and XML) to make asynchronous requests to the server without reloading the page.
- Fetch data from the server (e.g., movie details, seat availability) and update the page dynamically.
- Implement features like autocomplete, tooltips, sliders, etc., to enhance the user experience and usability of the application.

4.4 BACKEND DEVELOPMENT:

Backend development for the movie ticket booking system involves implementing Java Servlets to handle HTTP requests and responses, as well as using JDBC (Java Database Connectivity) to interact with the database. Let's discuss Java Servlets and JDBC in detail:

Java Servlets:

Java Servlets are Java classes that extend the capabilities of servers that host applications accessed through a request-response programming model. Servlets are particularly useful for handling HTTP requests, such as GET and POST, and generating dynamic web content. Here's how you can utilize servlets in the movie ticket booking system:

Handling HTTP Requests:

- Servlets typically override the doGet() and doPost() methods to handle GET and POST requests respectively.
- In the movie ticket booking system, servlets can handle requests for functionalities like user authentication, movie listing, booking, etc.

Processing Form Data:

- Servlets receive form data submitted by users via HTTP POST requests.
- They can extract form parameters using the request.getParameter() method and process them accordingly.

- For example, in the registration servlet, you can extract username, password, email, etc., from the registration form and save them to the database.

Generating Dynamic Content:

- Servlets can dynamically generate HTML content by writing directly to the response.getWriter() object.
- They can also forward requests to JSP pages for generating HTML content with embedded Java code.

Session Management:

- Servlets can manage user sessions using HttpSession objects.
- They can create, retrieve, and invalidate sessions to maintain user state across multiple requests.

Interacting with Other Components:

- Servlets can interact with other components such as business logic classes, database access objects (DAOs), and external services.
- They delegate business logic tasks to other components and focus on request handling and response generation.

JDBC (Java Database Connectivity):

JDBC is a Java API for connecting and executing SQL queries against a database. It provides a standard interface for accessing relational databases from Java applications. Here's how you can use JDBC in the movie ticket booking system:

Establishing Database Connection:

- Use JDBC to establish a connection to the database using the DriverManager.getConnection() method.
- Provide the database URL, username, and password to establish the connection.

Creating and Executing SQL Queries:

- Use JDBC Statement or PreparedStatement objects to create and execute SQL queries.
- Statement objects are used for executing static SQL queries, while PreparedStatement objects are preferred for parameterized queries to prevent SQL injection attacks.

Retrieving and Processing Results:

- Execute SQL queries to fetch data from the database (e.g., movie details, user information, booking details).
- Use ResultSet objects to retrieve query results and process them in Java code.

Handling Transactions:

- JDBC supports transactions for executing multiple SQL statements as a single unit of work.
- Use Connection objects to manage transactions and commit or rollback changes as needed.

Handling Exceptions:

- Handle exceptions that may occur during database operations (e.g., connection errors, SQL syntax errors).
- Use try-catch blocks to catch exceptions and handle them gracefully (e.g., logging errors, displaying error messages to users).
- By leveraging Java Servlets and JDBC, you can implement the backend logic of the movie ticket booking system effectively, handling HTTP requests, interacting with the database, and providing dynamic content to users.

4.5 DATA BASE MANAGEMENT:

MySQL is a popular open-source relational database management system (RDBMS) that is widely used for building scalable and reliable database-driven applications. It offers various features and functionalities that make it suitable for a wide range of use cases. Let's explore MySQL in detail:

Relational Database Management System (RDBMS):

MySQL follows the relational database model, where data is organized into tables with rows and columns. It supports SQL (Structured Query Language) for defining, manipulating, and querying data stored in the database.

Features of MySQL:

- Multi-Platform Support:
- MySQL is available for various platforms including Windows, Linux, macOS, etc., making it versatile and suitable for different environments.
- Scalability:
- MySQL is designed to handle large datasets and high traffic loads. It supports features like replication, clustering, and sharding for scalability and high availability.
- Performance:
- MySQL offers efficient query execution and indexing mechanisms for optimizing database performance. It supports various storage engines (e.g., InnoDB, MyISAM) with different performance characteristics.
- Security:
- MySQL provides robust security features including user authentication, access control, encryption, and SSL/TLS support to ensure data confidentiality and integrity.
- High Availability and Replication:
- MySQL supports replication, allowing you to create replicas of the database for load balancing and fault tolerance. It also supports features like master-slave replication and master-master replication.

- Transactional Support:
- MySQL supports ACID (Atomicity, Consistency, Isolation, Durability) transactions to ensure data integrity and reliability. It provides support for both implicit and explicit transactions.

Components of MySQL:

- The MySQL server is the core component responsible for storing, managing, and processing data. It includes the MySQL database engine and various server components.
- MySQL Client:
- The MySQL client provides command-line tools and graphical user interfaces (GUIs) for interacting with the MySQL server. Examples include the MySQL Command-Line Client and MySQL Workbench.
- Storage Engines:
- MySQL supports multiple storage engines that handle data storage and retrieval. Commonly used storage engines include InnoDB (transactional), MyISAM (non-transactional), MEMORY (in-memory), etc.
- Utilities:
- MySQL provides various utilities for database administration, backup, and maintenance. Examples include mysqldump for backup, mysqlimport for data loading, mysqlcheck for database checking and repair, etc.

Data Types in MySQL:

- MySQL supports various data types to store different types of data efficiently. Common data types include:
- Numeric types (e.g., INT, BIGINT, DECIMAL)
- Character and string types (e.g., VARCHAR, CHAR, TEXT)
- Date and time types (e.g., DATE, TIME, DATETIME)
- Binary and spatial types (e.g., BLOB, GEOMETRY)

Querying Data with SQL:

- MySQL uses SQL (Structured Query Language) for querying and manipulating data. SQL allows you to perform various operations such as selecting, inserting, updating, and deleting data from tables. Examples of SQL statements include:
- SELECT: Retrieve data from one or more tables.
- INSERT: Insert new rows into a table.
- UPDATE: Update existing rows in a table.
- DELETE: Delete rows from a table.
- JOIN: Combine data from multiple tables based on a related column.

Conclusion:

MySQL is a powerful and versatile relational database management system that offers scalability, performance, security, and a wide range of features for building database-driven applications. Its ease of use, reliability, and strong community support make it a popular choice for developers and organizations worldwide.

CUSTOMER FEATURES

- Detailed explanation of user-related functionalities.
- Customer registration process.
- Secure customer login system.
- Displaying movie listings and showtimes.
- Booking movie tickets, seat selection, and payment integration.
- Viewing booking history.
- Choosing favorite theaters.

ADMIN FEATURES

- Description of admin-specific functionalities.
- Admin panel for managing movies, showtimes, seats, and users.
- Adding, editing, or deleting movie details.
- Managing showtimes and seat availability.
- Admin only have permission to access the database.
- Admin have rights to add or remove employees.
- Admin can able to edit all databases.

CHAPTER 5

PROJECT DESCRIPTION

5.1 PROBLEM DEFINITION:

Problem Definition for Movie Ticket Booking System using JSP: Development of an Online Movie Ticket Booking System using JSP

Introduction - The rapid growth of the entertainment industry, particularly the film sector, has led to increased demand for convenient and efficient methods of booking movie tickets. Traditional ticket booking methods are often time-consuming and inconvenient for users, leading to the need for an online solution that offers ease of access and flexibility.

Problem Statement - There is a lack of a comprehensive online platform for booking movie tickets, which results in inconvenience for moviegoers and potential revenue loss for theaters. Current ticket booking systems may lack user-friendly interfaces, real-time availability updates, and seamless payment integration, leading to a suboptimal user experience.

Objectives - Develop an intuitive and user-friendly web-based application for booking movie tickets using JSP (JavaServer Pages) technology. Provide a platform where users can easily browse movies, view showtimes, select seats, and complete bookings seamlessly. Implement real-time updates on seat availability, ensuring accurate and up-to-date information for users. Integrate secure payment gateways to facilitate hassle-free online transactions and enhance user trust.Offer administrators a comprehensive dashboard for managing movies, showtimes, bookings, and user accounts efficiently.

Scope - The system will allow users to register and log in to their accounts, browse available movies, select showtimes, choose seats from an interactive seating map, and complete bookings securely. Administrators will have access to a backend dashboard for managing movie listings, showtimes, seating arrangements, booking records, and user accounts. The system will support multiple theaters, movie screenings, and payment methods to cater to diverse user preferences. Integration with third-party APIs for real-time updates on movie schedules, seat availability, and payment processing will be considered within the scope of the project.

Methodology - The development will follow an iterative and incremental approach, starting with requirements gathering, system design, implementation, testing, and deployment phases. Agile methodologies such as Scrum or Kanban may be adopted to facilitate collaboration, prioritize features, and ensure timely delivery of incremental updates. Regular feedback sessions with stakeholders, including end-users and administrators, will be conducted to gather input, validate assumptions, and refine the system accordingly.

Deliverables - A fully functional online movie ticket booking system deployed on a web server accessible to users. User documentation providing instructions for registration, booking tickets, and navigating the system. Administrator documentation detailing system administration tasks, such as managing movies, showtimes, and user accounts. Source code repository hosted on a version control system (e.g., Git) for code management and collaboration.

Constraints - The project budget and timeline may impose constraints on the scope and complexity of features that can be implemented within the given resources. Compatibility with existing theater infrastructure, legacy systems, and third-party APIs may present technical constraints that need to be addressed during development.

Expected Benefits - Enhanced user experience: Users will benefit from the convenience of booking movie tickets online, saving time and effort compared to traditional methods. Increased revenue: The system will enable theaters to reach a wider audience and capitalize on online ticket sales, leading to potential revenue growth. Improved operational efficiency: Administrators will have access to tools for streamlining ticket management, reducing manual effort, and optimizing resource allocation.

By defining the problem statement, objectives, scope, methodology, deliverables, constraints, and expected benefits, stakeholders can gain a clear understanding of the proposed Movie Ticket Booking System using JSP and align their expectations accordingly.

5.2 OVERVIEW OF THE PROJECT:

Introduction - The Online Movie Ticket Booking System is a web-based application designed to streamline the process of booking movie tickets for users. Utilizing JSP (JavaServer Pages) technology, the system provides a user-friendly interface for browsing movies, selecting showtimes, choosing seats, and completing bookings.

Objective - The primary objective of the project is to develop a comprehensive online platform that offers convenience, efficiency, and flexibility in booking movie tickets. By leveraging JSP technology, the system aims to provide real-time updates on movie schedules, seat availability, and secure payment processing.

Features

- User Registration and Login: Users can create accounts and log in to access personalized features and booking history. Movie Listings: Displays a list of available movies along with details such as title, genre, synopsis, and showtimes.
- Seat Selection: Provides an interactive seating map for users to select their preferred seats in the theater.
- Booking Process: Guides users through a seamless booking process, including seat selection, personal details entry, and payment confirmation.
- Real-time Updates: Ensures up-to-date information on movie schedules, seat availability, and booking status. Administrator Dashboard: Offers

administrators tools for managing movie listings, showtimes, seating arrangements, bookings, and user accounts. Secure Payment Integration: Integrates with secure payment gateways to facilitate online transactions securely. User Feedback: Allows users to provide feedback and ratings for movies and overall experience.

Scope - The system caters to both users and administrators, providing distinct interfaces and functionalities tailored to their respective roles. Users can browse movies, select showtimes, choose seats, complete bookings, and access booking history. Administrators have access to a backend dashboard for managing movies, showtimes, seating arrangements, bookings, and user accounts. Integration with third-party APIs for movie schedules, payment processing, and notification services is within the scope of the project.

Technology Stack - Frontend: HTML, CSS, JavaScript. Backend: JSP (JavaServer Pages), Java Servlets. Database: MySQL, PostgreSQL, Oracle Database Web Server: Apache Tomcat. Development Environment: Eclipse, IntelliJ IDEA, NetBeans

Methodology - The project follows an iterative and incremental development approach, allowing for continuous feedback and refinement. Agile methodologies such as Scrum or Kanban may be adopted to manage project tasks, prioritize features, and ensure timely delivery. Regular communication and collaboration among team members, stakeholders, and end-users are essential for project success.

Deliverables - A fully functional web-based application deployed on a web server accessible to users. User and administrator documentation providing instructions for system usage and administration tasks. Source code repository hosted on a version control system (e.g., Git) for code management and collaboration.

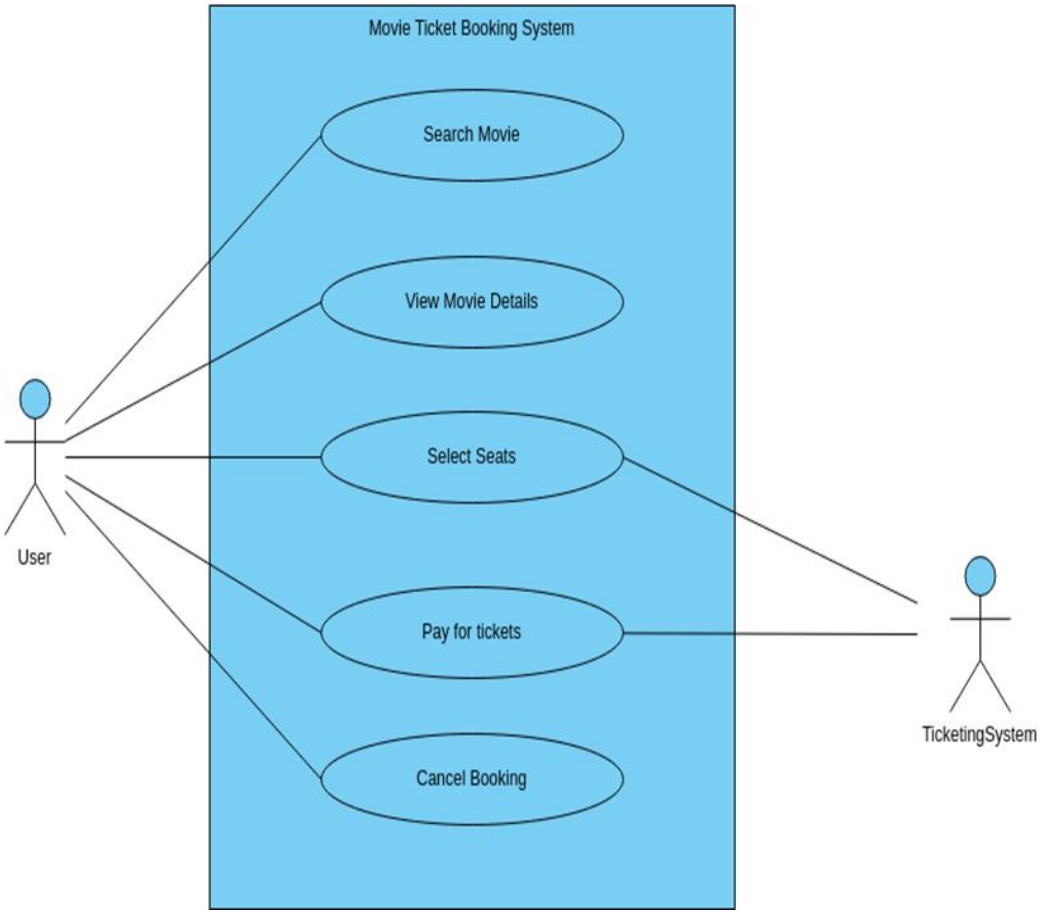
Expected Benefits

- Enhanced user experience: Users benefit from the convenience of booking movie tickets online, saving time and effort.
- Increased revenue: The system enables theaters to reach a wider audience and capitalize on online ticket sales.

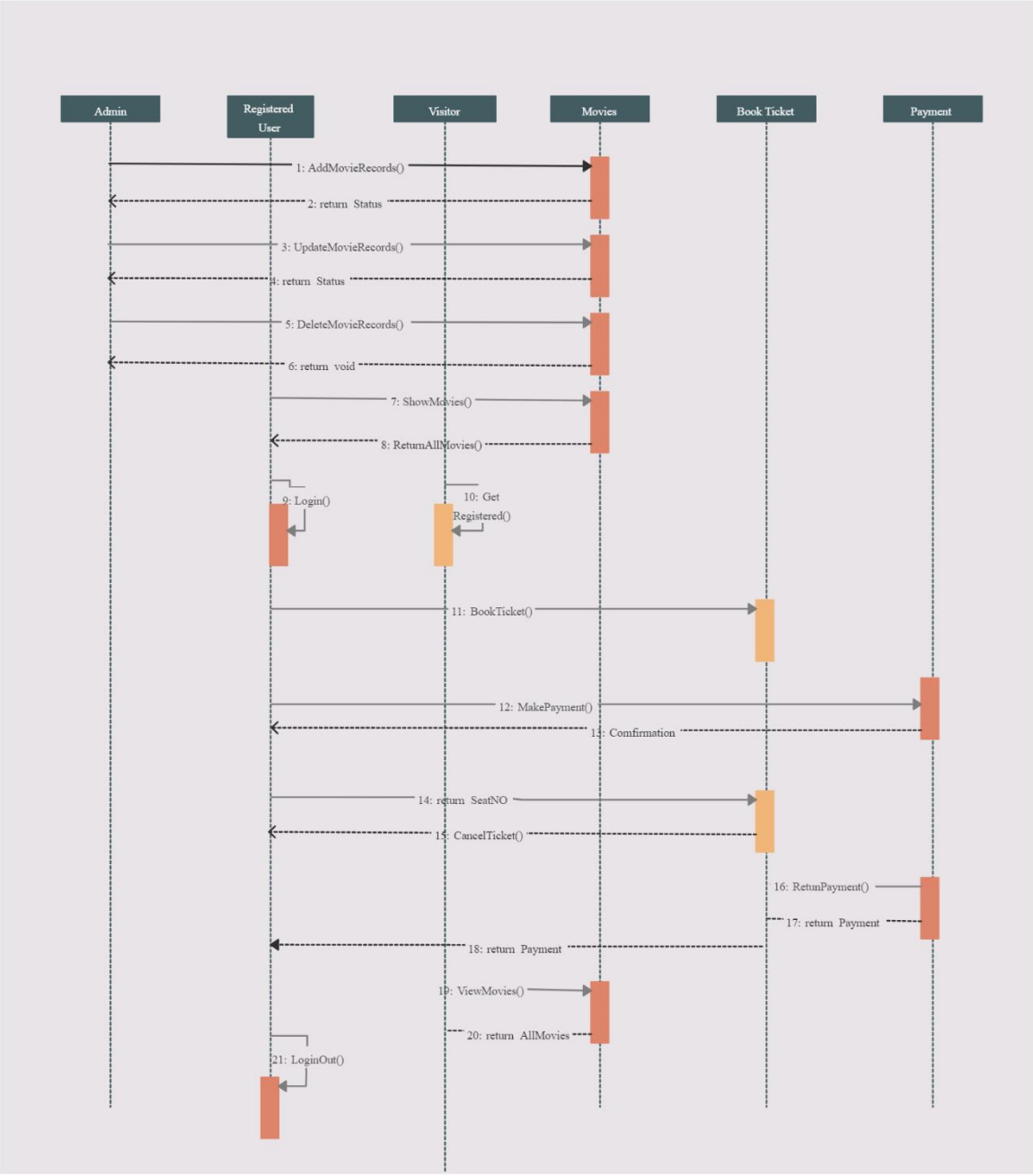
- Improved operational efficiency: Administrators have tools for streamlining ticket management, reducing manual effort, and optimizing resource allocation.

- By providing an overview of the project, stakeholders can gain a clear understanding of the goals, features, scope, technology stack, methodology, deliverables, and expected benefits of the Movie Ticket Booking System developed using JSP.
- Enhanced user experience: Users benefit from the convenience of booking movie tickets online, saving time and effort.

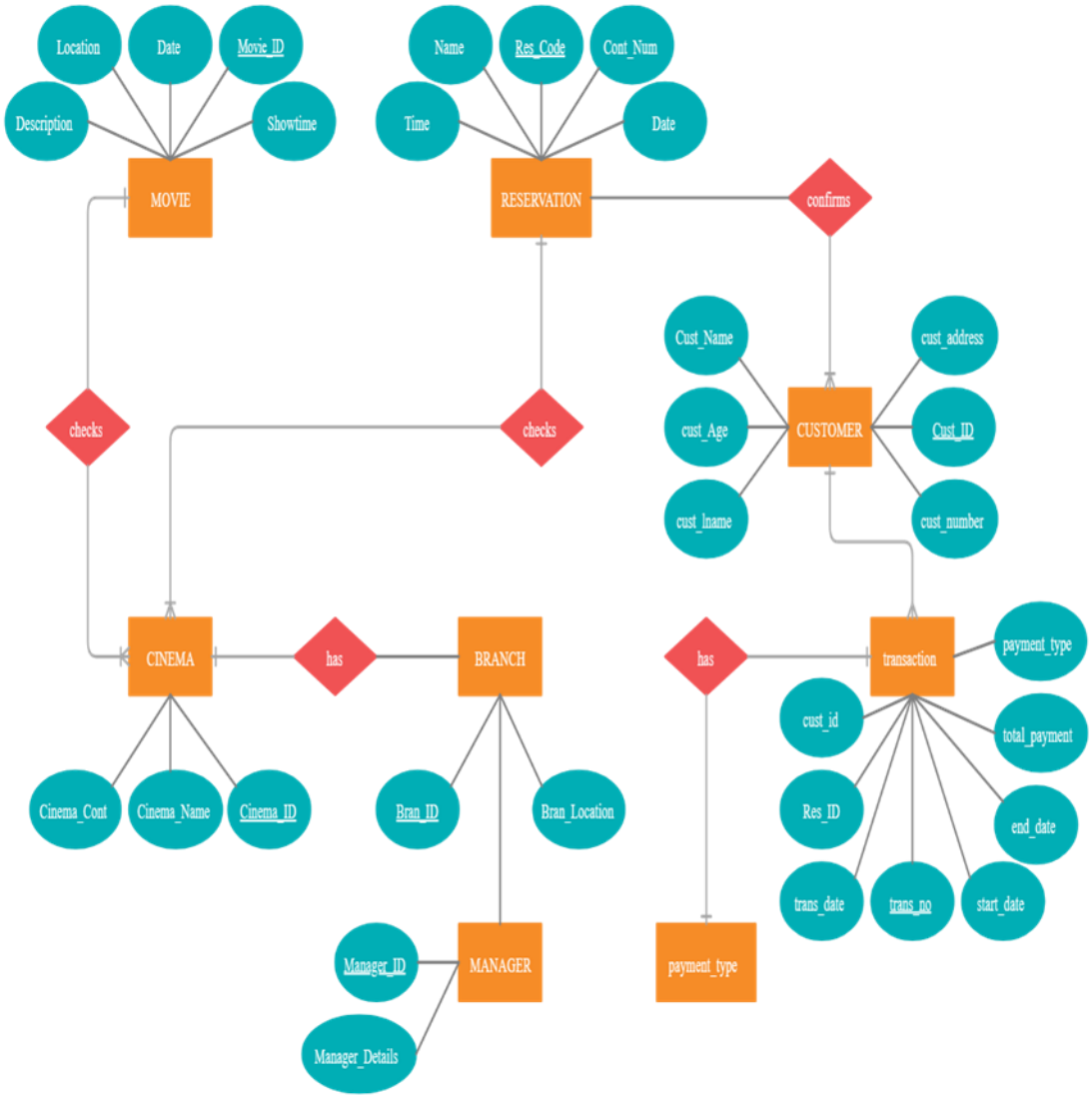
5.3 USE-CASE DIAGRAM :



5.4 FLOW DIAGRAM:



5.5 INPUT AND OUTPUT DESIGN:



CHAPTER 6

SOFTWARE TESTING

6.1 UNIT TESTING:

Objective - Verify the correctness of individual components, such as classes, methods, and functions.

Test Cases

- Verify that user registration functionality creates a new user account with correct details.
- Test seat selection functionality to ensure it correctly updates the selected seat status.
- Validate that payment processing functionality securely processes payments using mock payment gateways.
- Test movie listing functionality to ensure it displays accurate movie details fetched from the database.

6.2 INTEGRATION TESTING:

Objective - Verify interactions and data flow between different components/modules of the system.

Test Cases

- Test the integration between the frontend and backend components to ensure seamless communication.
- Verify the integration between the database and application layers for accurate data retrieval and storage.
- Test API integrations with third-party services for real-time updates on movie schedules and payment processing.
- Tools: Postman (for API testing), Selenium (for end-to-end testing)

6.3 ACCEPTANCE TESTING:

Objective - Validate that the system meets specified requirements and is ready for deployment.

Test Cases

- Test user registration and login functionality to ensure users can create accounts and log in successfully.
- Verify that users can browse movies, select showtimes, choose seats, and complete bookings without errors.
- Test administrator functionalities, such as managing movie listings, showtimes, and user accounts.
- Validate payment processing to ensure transactions are completed securely and accurately.
- Tools: Manual testing, User acceptance testing (UAT)

6.4 TEST CASES:

User Registration and Login:

- Verify that users can register with valid details.
- Test login with registered credentials to ensure successful authentication.
- Validate error handling for invalid or duplicate registration information.

Movie Listing and Selection:

- Test that the system displays accurate movie listings with relevant details.
- Verify that users can select preferred movies, genres, and showtimes.
- Validate filtering and sorting functionalities for movie listings.

Seat Selection and Booking:

- Test seat selection functionality to ensure users can choose seats from an interactive seating map.
- Verify that selected seats are reserved and unavailable for other users.
- Validate the booking process, including entry of user details, seat confirmation, and payment processing.

Administrator Dashboard:

- Test administrator login functionality to ensure secure access.
- Verify that administrators can manage movie listings, showtimes, and seating arrangements.
- Validate user management functionalities, including account creation, deletion, and role assignment.

Payment Processing:

- Test payment gateway integration to ensure secure and seamless transaction processing.
- Verify that payment details are encrypted and securely transmitted.
- Validate handling of payment failures and error scenarios.

By conducting thorough unit testing, integration testing, and acceptance testing, the Movie Ticket Booking System can be rigorously evaluated to ensure its reliability, functionality, and readiness for deployment.

CHAPTER 7

CONCLUSION & FUTURE ENHANCEMENT

7.1 CONCLUSION:

In conclusion, the development of an Online Movie Ticket Booking System using JSP presents a significant opportunity to enhance the movie-going experience for users while streamlining operations for theater administrators. Throughout the project, various aspects have been addressed, including the problem definition, objectives, scope, technology stack, methodology, and testing strategies.

By leveraging JSP technology along with Java Servlets, HTML, CSS, and JavaScript, the system offers a user-friendly interface for browsing movies, selecting showtimes, choosing seats, and completing bookings seamlessly. Real-time updates on movie schedules, seat availability, and secure payment processing further enhance the user experience and operational efficiency.

The project has been approached using an iterative and incremental development methodology, allowing for continuous feedback, refinement, and adaptation to evolving requirements. Unit testing, integration testing, and acceptance testing have been employed to ensure the correctness, reliability, and readiness of the system for deployment.

Overall, the Online Movie Ticket Booking System promises to deliver enhanced convenience, efficiency, and flexibility for users, while enabling theaters to reach a wider audience and capitalize on online ticket sales. With careful planning, thorough execution, and ongoing support, the system is poised to make a positive impact on the movie ticket booking industry.

7.2 Future Enhancements

For future enhancements of a movie ticket booking system developed using JSP in NetBeans, you can consider implementing various features and improvements to enhance the user experience, security, performance, and functionality of the application. Here are some ideas for future enhancements:

User Account Management:

- Implement user profiles where users can manage their personal information, view booking history, and update preferences.
- Introduce features like password reset, email verification, and two-factor authentication for enhanced security.

Social Media Integration:

- Allow users to sign in/register using their social media accounts (e.g., Facebook, Google).
- Implement social sharing features to allow users to share movie details or booking confirmations with friends.

Advanced Search and Filtering:

- Enhance the movie listing page with advanced search and filtering options based on genres, language, release date, etc.
- Implement sorting options to allow users to sort movies by popularity, rating, or release date.

Recommendation System:

- Develop a recommendation engine that suggests movies based on users' viewing history, preferences, and ratings.
- Implement collaborative filtering or content-based recommendation algorithms to provide personalized recommendations.

Real-Time Seat Availability:

- Display real-time seat availability information to users during the booking process.

- Implement a dynamic seating chart to visualize available and booked seats in the theater.

Mobile App Development:

- Build a mobile application for Android and iOS platforms to provide users with a convenient way to book tickets on the go.
- Ensure synchronization between the web application and mobile app for seamless user experience.

Payment Gateway Integration:

- Integrate with multiple payment gateways to offer users a choice of payment methods (e.g., credit/debit card, net banking, digital wallets).
- Implement secure payment processing with encryption and fraud detection mechanisms.

Admin Dashboard:

- Develop an admin dashboard for theater owners or administrators to manage movie listings, theater schedules, and bookings.
- Provide analytics and reporting features to track sales, occupancy rates, and customer feedback.

Localization and Internationalization:

- Support multiple languages and currencies to cater to a global audience.
- Implement localization and internationalization features to adapt the application's content and format based on users' preferences.

Performance Optimization:

- Optimize database queries, server-side code, and client-side resources to improve application performance and responsiveness.
- Implement caching mechanisms to reduce load times and server overhead.

Feedback and Rating System:

- Allow users to rate movies, theaters, and overall booking experience.
- Collect user feedback and reviews to identify areas for improvement and enhance customer satisfaction.

Integration with External APIs:

- Integrate with external APIs for weather forecasts, traffic updates, or event notifications to provide additional value-added services to users.
- These are just a few ideas for future enhancements of a movie ticket booking system. Prioritize features based on user feedback, market trends, and business goals to continuously improve and evolve the application over time.

CHAPTER 8
APPENDIX

8.1 SOURCE CODE:

index.jsp

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN""http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Cinema World Home page</title>
    <meta http-equiv="Content-Type" content="text/html;
charset=utf 8" />
    <%--<link href="themes/default/dream.css" rel="stylesheet" type="text/css"/>--
%>
    <link href="css/style.css" rel="stylesheet" type="text/css" />
    <link href="css/ie6.css" rel="stylesheet" type="text/css" />
    <scriptsrc="js/jquery-1.4.2.min.js" type="text/javascript"></script>
    <script src="js/cufon-yui.js" type="text/javascript"></script>
    <script src="js/cufon-replace.js" type="text/javascript"></script>
  <scriptsrc="js/Gill_Sans_400.font.js"type="text/javascript"></script>
    <script src="js/script.js" type="text/javascript"></script>
    <script type="text/javascript" src="js/ie_png.js"></script>
    <script type="text/javascript">ie_png.fix('.png, .link1 span, .link1');</script>
    <link href="css/ie6.css" rel="stylesheet" type="text/css" />
      <link href="css/dream.css" rel="stylesheet" type="text/css" />
    <style type="text/css">
```

```
<!--
.style1 {font-size: 36px} -->
</style>
</head>
<div id="header">
<div class="row-1">
<div class="fleft"><a href="index.jsp">Cinema <span style="color:red">World</span></a></div>
<ul>
<li><a href="index.jsp"><img src="images/icon1-act.gif" alt="Home Page" /></a></li>
<li><a href="contact-us.html"><img src="images/icon2-act.gif" alt="Gmail" /></a></li>
<li><a href="about-us.html"><img src="images/icon3-act.gif" alt="About-us" /></a></li>
</ul>
</div>
</div>
<body id="page1">
<!-- START PAGE SOURCE -->
<div class="tail-top">
<div class="tail-bottom">
<div id="main">
<div id="header">
<div class="row-2">
```

```
          <ul>
            <li><a href="index.jsp">Home</a></li>
            <li><a href="about-us.html">About</a></li>
            <li><a href="articles.html">Articles</a></li>
            <li><a href="contact-us.html">Contacts</a></li>
          </ul>
        </div>
      </div>
        </div>
      </div></div>
<table width="100%">

          <tr><td>
                  <table><tr><td  valign="top">
                          <table id="navigation"  width="170">
                                  <tr><td><a
href="adminLogin.jsp">Admin</a></td></tr>
                          <tr><td>
<a href="employeelogin.jsp">Employee</a></td></tr>
                                  <tr><td>
<a href="CustomerLogin.jsp">Customers</a></td></tr>
                                  <tr><td>
<a href="cinema.html">Cinemas</a></td></tr>
 <%--<tr><td><a href="#">Movies</a></td></tr>--%>
                                  <tr><td>
```

Schedule</td></tr>

<%--<tr><td>

Time Slot</td></tr>--%>

<%--<tr><td>

Ticket</td></tr>--%>

<%--<tr><td>Log Out</td></tr>--%>

</table>

</td>

<td id="databar">

</td>

</tr></table></td></tr></table>

<table id="footer" width="100%" style="background-color: black" >

<tr>

<td style="background-color: black" >

Home

Schedule |

Privacy Policy |

<a 0-href="contact-us.html">

Contact us

Disclaimer

About us

</td></tr></table></body></html>

employeelogin.jsp

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

  <head>

    <title>Employee Login Page</title>

 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" /> <%--<link href="themes/default/dream.css" rel="stylesheet" type="text/css"/>--%>

    <link href="css/style.css" rel="stylesheet" type="text/css" />

    <link href="css/ie6.css" rel="stylesheet" type="text/css" />


      <link href="css/dream.css" rel="stylesheet" type="text/css" />

    <style type="text/css">

    <!--

    .style1 {font-size: 36px}

    -->

    </style>

  </head>

    <div id="header">

<div class="row-1">

<div class="fleft"><a href="index.jsp">Cinema <span style="color:red">World</span></a></div>

      <ul>
```

```
        <li><a href="index.jsp"><img src="images/icon1-act.gif" alt="Home Page" /></a></li>
        <li><a href="contact-us.html"><img src="images/icon2-act.gif" alt="Gmail" /></a></li>
        <li><a href="about-us.html"><img src="images/icon3-act.gif" alt="About-us" /></a></li>
      </ul>
    </div>
  </div>
    <body id="page1">
<!-- START PAGE SOURCE -->
<div class="tail-top">
 <div class="tail-bottom">
  <div id="main">
   <div id="header">
    <div class="row-2">
     <ul>
      <li><a href="index.jsp">Home</a></li>
      <li><a href="about-us.html">About</a></li>
      <li><a href="articles.html">Articles</a></li>
      <li><a href="contact-us.html">Contacts</a></li>
     </ul>
    </div>
   </div>
    </div>
```

```
</div></div>

<%--<img src="Media/icon.png" alt="Cinema World" border="0" style="position:absolute; left: 35%;top:0px; "/>--%>

<table width="100%">
    <tr><td>
        <table><tr><td  valign="top">
            <table id="navigation"  width="170">
                <tr><td><a href="adminLogin.jsp">Admin</a></td></tr>
                <tr><td><a href="employeelogin.jsp">Employee</a></td></tr>
                <tr><td><a href="CustomerLogin.jsp">Customers</a></td></tr>
                <tr><td><a href="cinema.html">Cinemas</a></td></tr>
                <%--<tr><td><a href="#">Movies</a></td></tr>
                <tr><td><a href="schedule.jsp">Schedule</a></td></tr>
                <tr><td><a href="javascript:viewTimeslot()">Time Slot</a></td></tr>
                <tr><td><a href="javascript:viewTicket()">Ticket</a></td></tr>
                <%--<tr><td><a href="javascript:logout()">Log Out</a></td></tr>--%>
            </table></td>
            <td id="databar">
```

```
                <h4><span style="color:white">Employee Login :  </span></h4>
        <form action="employeeauth.jsp" method="post">
         <label for="name"><br />
          Username :<br /><br/>
         </label>
         <input name="userid" type="text" id="name"/>
         <label for="name"><br />
         <br />
          Password :<br /><br/>
         </label>
          <input type="password" id="pass" name="pass" />
            <br /><br/>
         <p>
           <input name="imageField" type="submit" class="LOGIN" id="imageField" value="Login"  />
           <input type="reset" name="imageField" id="imageField"  class="RESET" />
         </p>
         <p> </p>
        </form>
                <a href="NewEmployeeRegistration.jsp">Click here for New Registration!!</a>
               </td>
          </tr>
            </table></td></tr></table></body></html>
```

customerlogin.jsp

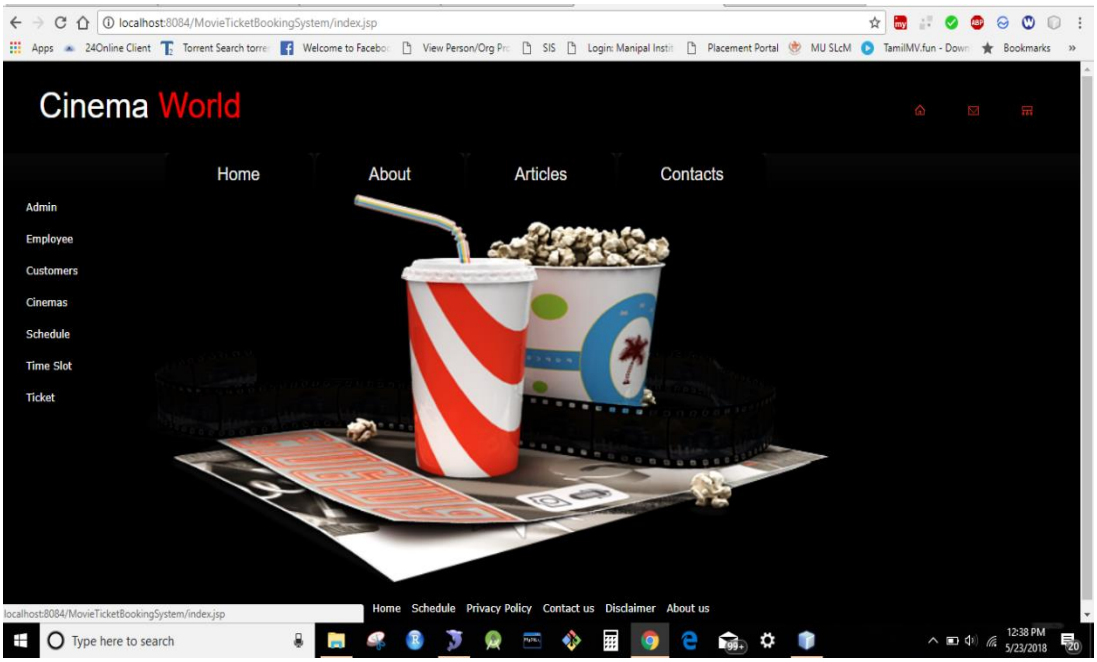
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
   <head>
      <title>Customer Login Page</title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
      <%--<link href="themes/default/dream.css" rel="stylesheet" type="text/css"/>--%>
      <link href="css/style.css" rel="stylesheet" type="text/css" />
      <link href="css/ie6.css" rel="stylesheet" type="text/css" />

        <link href="css/dream.css" rel="stylesheet" type="text/css" />
      <style type="text/css">
      <!--
      .style1 {font-size: 36px}
      -->
      </style>
   </head>
      <div id="header">
<div class="row-1">
<div class="fleft"><a href="index.jsp">Cinema <span style="color:red">World</span></a></div>
        <ul>
```

```
        <li><a href="index.jsp"><img src="images/icon1-act.gif" alt="Home Page" /></a></li>
        <li><a href="contact-us.html"><img src="images/icon2-act.gif" alt="Gmail" /></a></li>
        <li><a href="about-us.html"><img src="images/icon3-act.gif" alt="About-us" /></a></li>
      </ul>
    </div>
  </div>
    <body id="page1">
<!-- START PAGE SOURCE -->
<div class="tail-top">
 <div class="tail-bottom">
  <div id="main">
   <div id="header">
    <div class="row-2">
     <ul>
      <li><a href="index.jsp">Home</a></li>
      <li><a href="about-us.html">About</a></li>
      <li><a href="articles.html">Articles</a></li>
      <li><a href="contact-us.html">Contacts</a></li>
     </ul>
    </div>
   </div>
    </div>
```

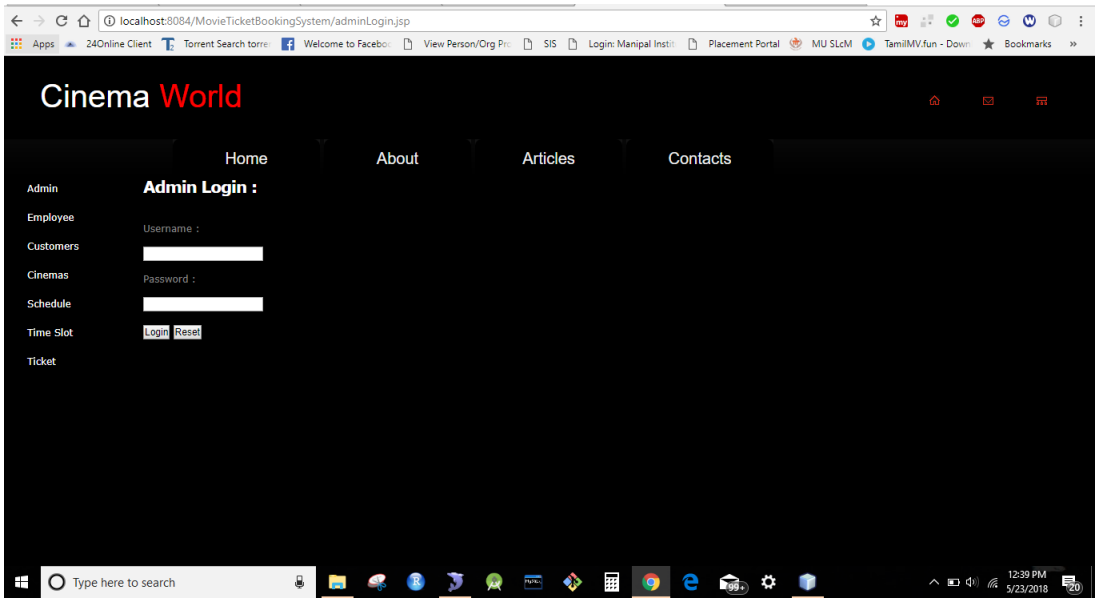
```
</div></div>

<%--<img src="Media/icon.png" alt="Cinema World" border="0" style="position:absolute; left: 35%;top:0px; "/>--%>

<table width="100%">
    <tr><td>
        <table><tr><td  valign="top">
            <table id="navigation"  width="170">
                <tr><td><a href="adminLogin.jsp">Admin</a></td></tr>
                <tr><td><a href="employeelogin.jsp">Employee</a></td></tr>
                <tr><td><a href="CustomerLogin.jsp">Customers</a></td></tr>
                <tr><td><a href="cinema.html">Cinemas</a></td></tr>
                <tr><td><a href="schedule.jsp">Schedule</a></td></tr>
                <%--<tr><td><a href="javascript:viewTimeslot()">Time Slot</a></td></tr>
                <tr><td><a href="javascript:viewTicket()">Ticket</a></td></tr>
                <tr><td><a href="javascript:logout()">Log Out</a></td></tr>--%>
            </table></td>
            <td id="databar">
            <h4><span style="color:white">Customer Login :  </span></h4>
```

```
<form action="customerauth.jsp" method="post">
 <label for="name"><br />
  Username :<br /><br/>
 </label>
 <input name="userid" type="text" id="name"/>
 <label for="name"><br />
 <br />
  Password :<br /><br/>
 </label>
  <input type="password" id="pass" name="pass" />
    <br /><br/>
 <p>
    <input name="imageField" type="submit" class="LOGIN" id="imageField" value="Login" />
    <input type="reset" name="imageField" id="imageField" class="RESET" />
 </p>
 <p> </p>
</form>
        <a href="customeregister.jsp">Click here to Register!!</a>
       </td>
  </tr>
    </table></td></tr></table>
</body>
</html>
```

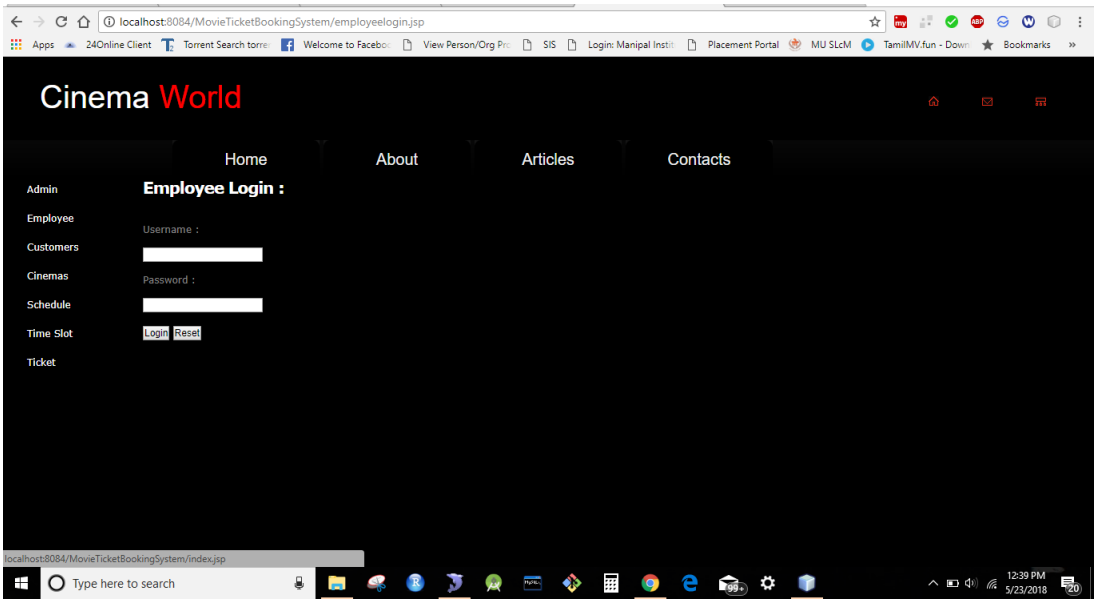
8.2 SYSTEM SNAPCHATS:



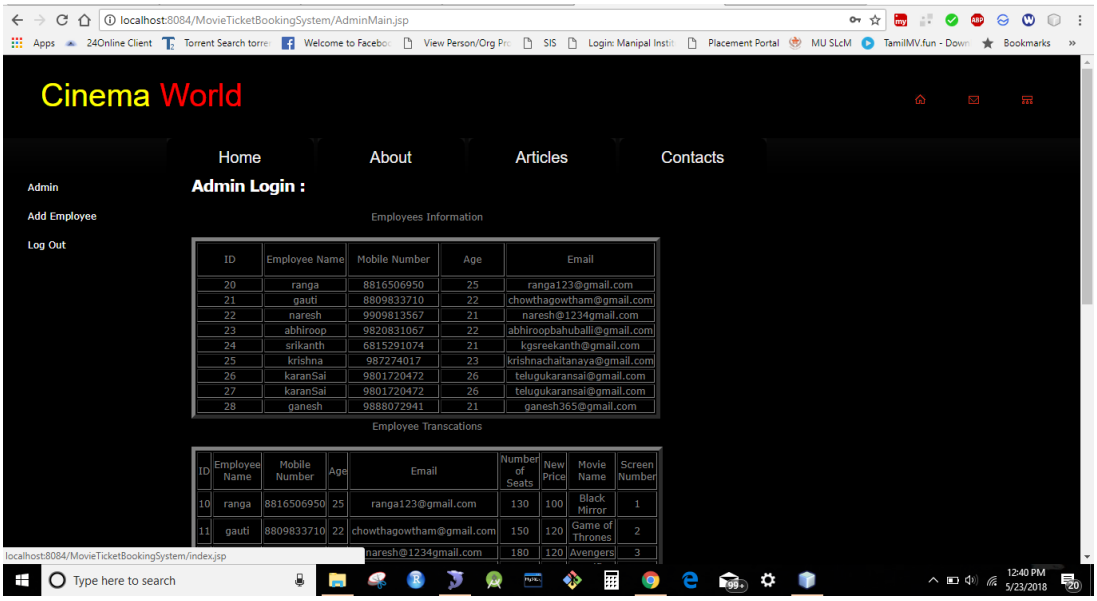
HOME PAGE



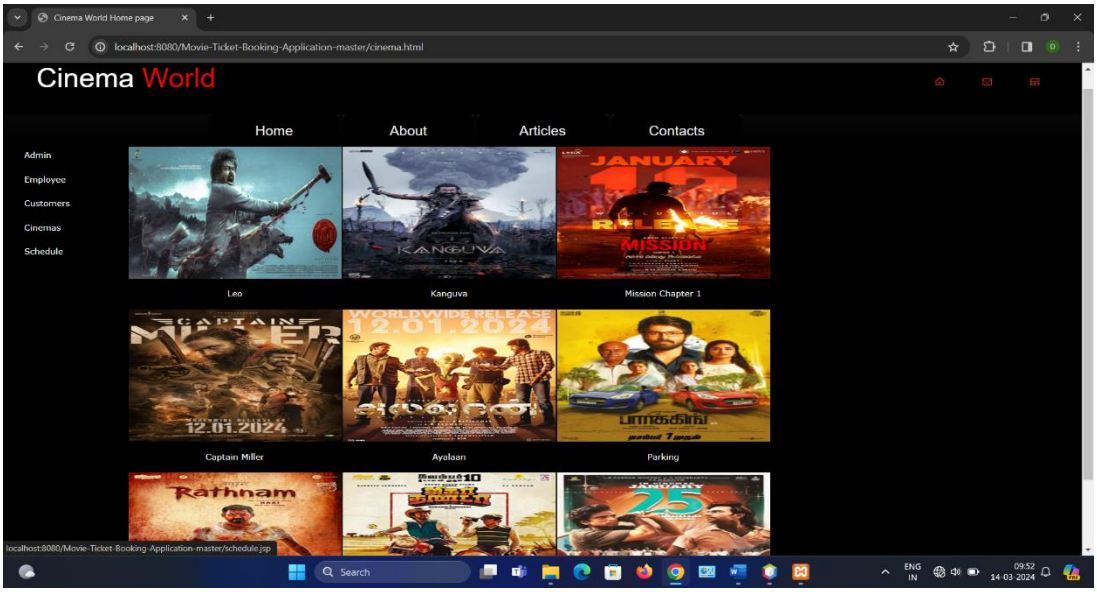
ADMIN LOGIN



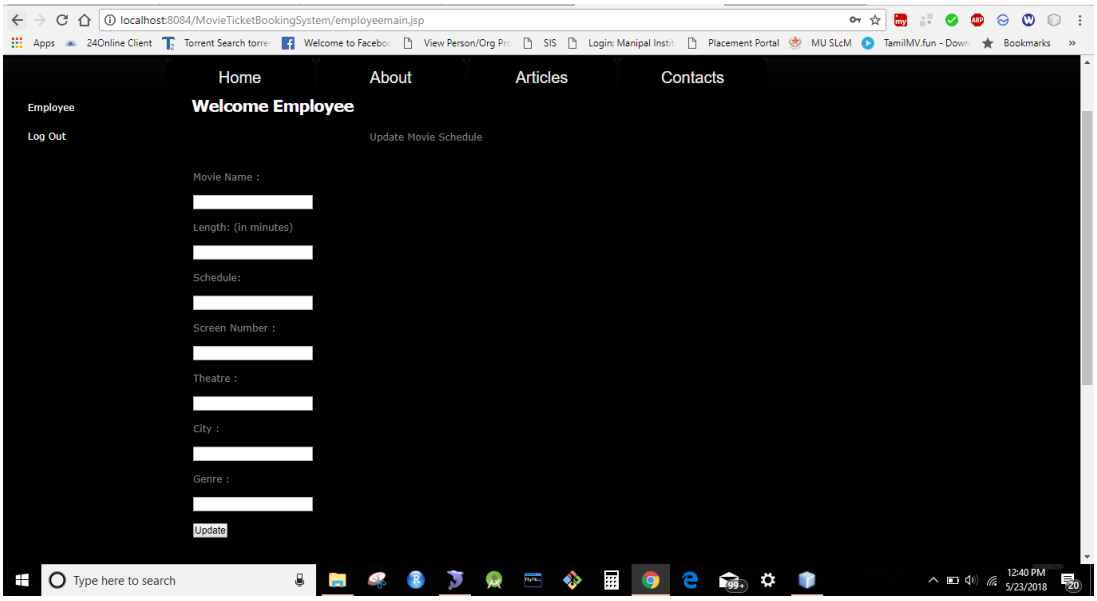
EMPLOYEE LOGIN



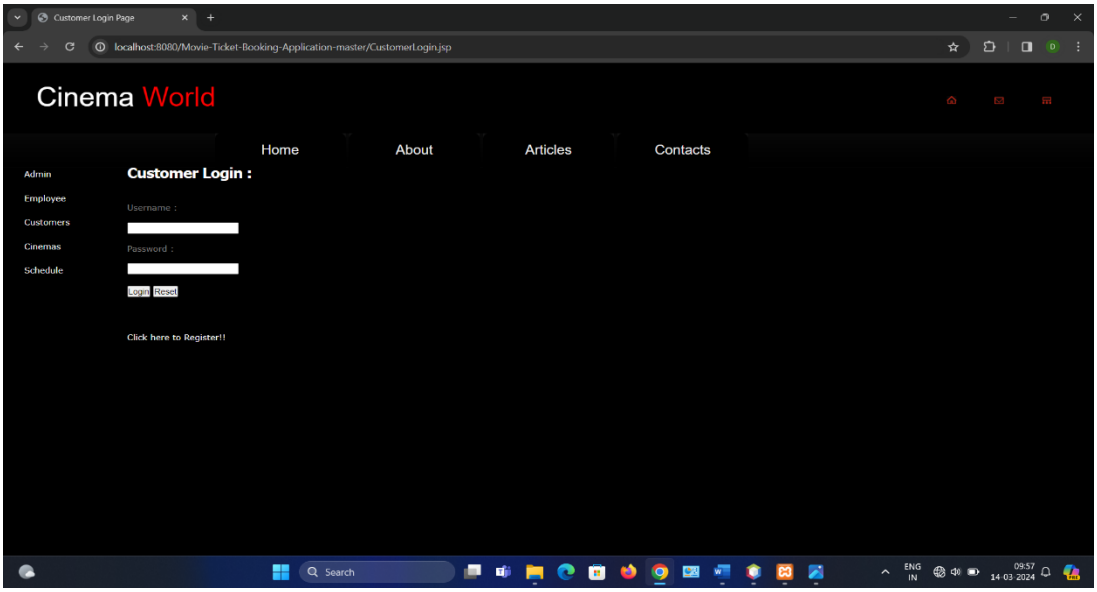
ADMIN PAGE



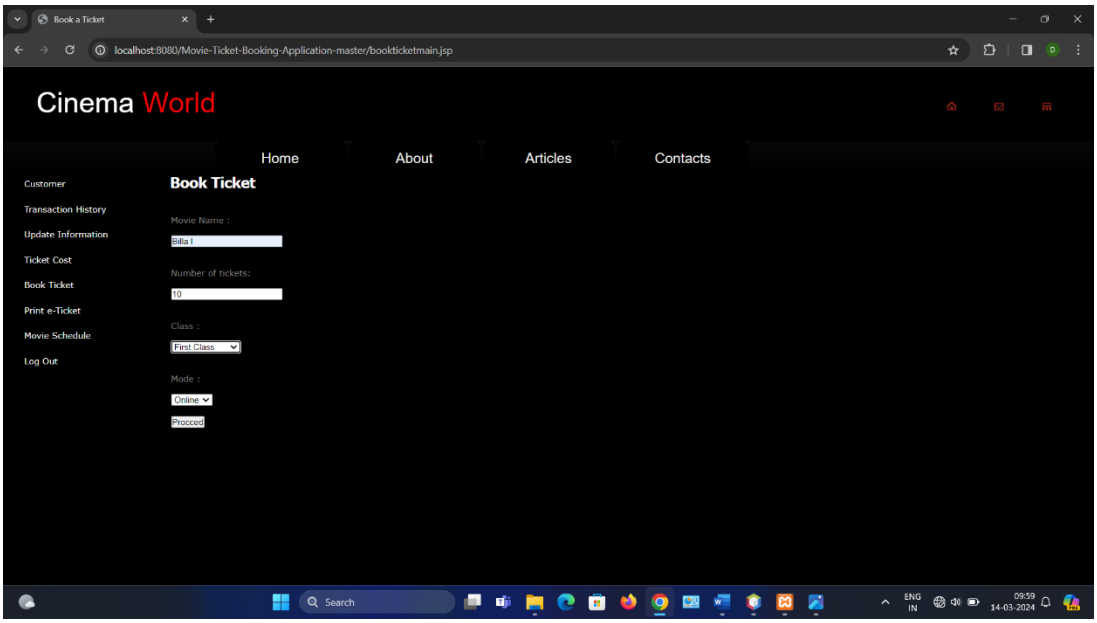
CINEMA



MOVIE UPDATION



CUSTOMER LOGIN



TICKET BOOKING

CHAPTER 9

BIBLIOGRAPHY

BIBLIOGRAPHY:

Java EE Documentation Oracle

URL: https://docs.oracle.com/javaee/7/index.html

NetBeans IDE Documentation Apache NetBeans

URL: https://netbeans.apache.org/documentation/index.html

HTML MDN Web Docs Mozilla Developer Network (MDN)

URL: https://developer.mozilla.org/en-US/docs/Web/HTML

CSS MDN Web Docs Mozilla Developer Network (MDN)

URL: https://developer.mozilla.org/en-US/docs/Web/CSS

Java Server Pages Oracle Tutorial Oracle

URL: https://docs.oracle.com/javaee/7/tutorial/jsf-intro.htm.

W3Schools Online Tutorials W3Schools

URL: https://www.w3schools.com/

Github Repository:

URL: https://github.com/chintamanand/Movie-Ticket-Booking-pplication.git