2004

# Movie theater ticket order system: (MTTOS)

Chun-Kai Chiu

# MOVIE THEATER TICKET ORDER SYSTEM

# (MTTOS)

---

A Project

Presented to the

Faculty of

California State University,

San Bernardino

---

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

---

by

Chun-Kai Chiu

June 2004

# MOVIE THEATER TICKET ORDER SYSTEM

# (MTTOS)

---

A Project

Presented to the

Faculty of

California State University,

San Bernardino

---

by

Chun-Kai Chiu

June 2004

Approved by:

[redacted]

Dr. David Turner, Chair, Computer Science

6/3/2004

Date

[redacted]

Dr. Richard Botting

[redacted]

Dr. Ernesto Gomez

ABSTRACT

Movie theater ticket order system (MTTOS) is an on-line system that sells movie theater tickets to the general public. The system allows visitors to the site to browse information about the available movies, including showing times, room number, movie summary, and an advertisement image. Visitors can purchase tickets online to attend any of the offered movies. If the user forgets the details of the ticket purchase, he or she can log in at any time to retrieve this information. Additionally, the system emails to the customer purchase information, including the date and time of the movie, name of movie, number and type of tickets, etc.

In addition to customer functions, the system also provides functions to an administrator that allows a movie theater employee to manage the contents of the site, including movie descriptions, showing times, prices, etc.

This project is based on a Model-View-Controller (MVC) architecture, which introduces a controller servlet to provide a single point of entry to the Web system and encourages more reuse and extensibility of the code. In this system, we use the PostgreSQL database server to maintain persistent data.

ACKNOWLEDGMENTS

I would like to express my deep appreciation to my graduate advisor, Dr. David Turner, for his important and valuable contributions to me and my project. Discussions with Dr. Turner always give me significant help to resolve my problems and give me positive directions to do my research. Dr. Turner offered me a lot of precious information and suggestions during all the period of doing my project. I also would like to thank my two project committee professors, Dr. Botting and Dr. Gomez, for their supportive help and important advice.

I would like to thank my parents, deeply from my heart, who provide me the chance to fulfill my every dream in life. Their love warmly supports me in every day and their encouragement gives me the power to accomplish every success in my life.

The support of the National Science Foundation under award 9810708 is gratefully acknowledged.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

# CHAPTER ONE

# INTRODUCTION

Movie theater ticket order system (MTTOS) is an on-line movie theater ticket order system. MTTOS allows customers to get movie information on-line and lets the movie theater sell tickets to customers. In MTTOS, information about movies is presented. The movie consumer can browse through all the movie and information. Since the information would be varies from time and room of movie, all possible information will be presented in an orderly fashion to account for all the different possibilities. The consumer can log on and check the order information from time to time. After the customers purchase tickets, this system emails the order details to the consumer automatically.

## 1.1 Purpose of this Project

This project is a movie theater ticket order system. This system allows people to get movie information and purchase tickets in the Internet. Visiting guests can apply to be club members of the movie theater. Members can login to the member club to check their order status any time. The administrator manages the whole information about movies and members.

In the system, security is very important, otherwise someone could get in the database and change the data or get sensitive information such as personal information of the customers. Also, private information can be intercepted on the Internet. This system will use firewalls and SSL (secure Sockets Layer) to solve these problems.

This is a software product being developed for a master's degree project. In its final form, this product will allow the consumer to order tickets and the details about past purchases in case they forget something, such as time of showing, date, number of tickets, etc. To join the theater's movie club, a consumer will be able to fill up the required forms online to join. All data is stored in a database (PostgreSQL).

## 1.2 Project Products

The MTTOS project led to the following products:

- Implementation of MTTOS: a working web-based online application system with JAVA programs, XML configuration, JSP and Postgres database, which achieves the specific needs of MTTOS. All the forms follow the original paper application

form in order to accomplish convenient and familiar processes for the user.

- System documentation: a project documentation, which is available with system design, specifications, project implementation and testing reports.

# CHAPTER TWO

## SYSTEM ARCHITECTURE

The MTTOS project aims to be friendly and convenient for the customers. The components of MTTOS are a web server, a database server, a client browser, and an email server. Customer browsers use the Internet (TCP/IP) to get movie information and purchase tickets through the web server under HTTP/HTTPS. The web server connects to the email server with TCP/IP in order to send or retrieve email through SMTP. The web application connects to the database with TCP, and accesses database functionality through JDBC.

In order to choose implementation components conforming to the criteria of the shareware standard, this project uses Tomcat as Web server, and PostgreSQL as database server. The other components, such as the web browsers, are dependant on which kind of browsers the customers use. The email server was provided by the ISP (Internet Service Provider).

The architecture of this project is shown below in Figure 1.

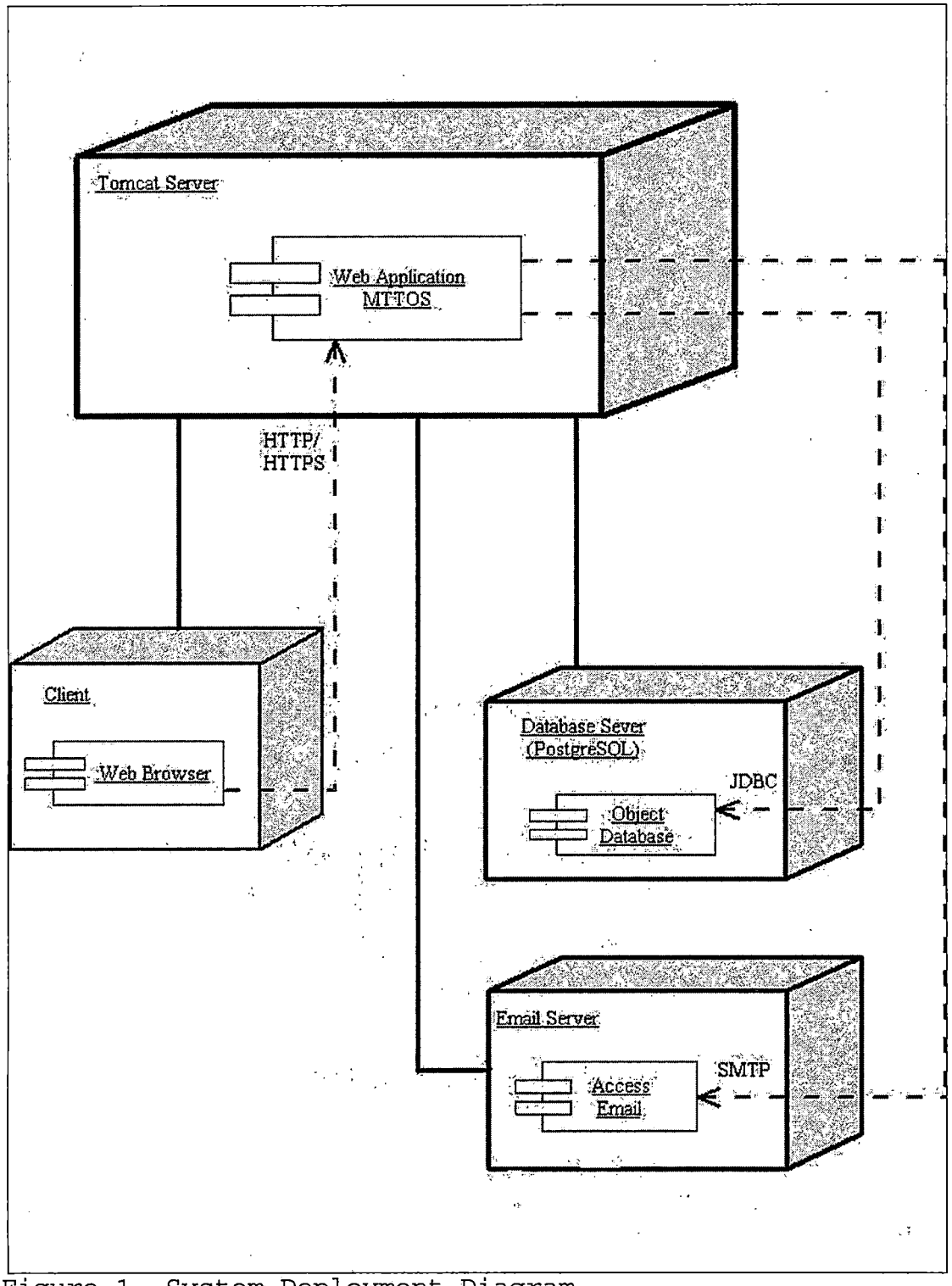


Figure 1. System Deployment Diagram

## 2.1 System Interfaces

The MTTOS is a 3-tier distributed architecture that displays the user interface in a Web browser using HTML. The middle tier is a Java Servlet (automatically created from JSP pages) that handles requests from the client browser and provides access to the third tier via JDBC, which is a PostgreSql database. The HTTP server is provided by Apache Tomcat, which also implements JSP and Java Servlet APIs.

## 2.2 Hardware Interfaces

MTTOS will not implement hardware interface directly. However, it will trust the underlying operating system (Windows, NT, ME, XP, Linux, UNIX, MAC) to handle the hardware interfaces.

## 2.3 Software Interfaces

As explained above, there will be two different software interfaces depending on the type of access that the user is demanding or the function that the user wants the software to perform. The reasons that are why I choose the software interfaces used in this project are summarized as following:

- Operating system (Linux/Unix Red Hat 9.0): RedHat is an open source and it is the most popular distribution for Linux.
- Web Server/Container (Jakarta Tomcat Server 4.1.29): Tomcat server is a Java based Web Application container that was created to run Servlets and JavaServer Pages (JSP) in Web applications.
- JAVA 2 Platform, Standard Edition (J2SE): A Java-based, runtime platform that provides many features for developing Web-based Java applications, including database access (JDBC API) interface technology, and security for both local network and Internet use and it's required in the Tomcat JAVA Container.
- Database Server (PostgreSQL Serve 7.4.1): PostgresSQL is open source database software, and is included in RedHat by default. PostgresSQL also provides a JDBC driver to easily connect from a JAVA program.
- Java Database Connector (JDBC): PostgreSQL connector.

- Build Tool (Apache Ant 1.6.0): Ant is a make-like utility.We endorse the use of Ant because it supports platform independence and it is easy to use.
- Email package: JavaMail 1.3.
- Languages: HTML/JAVA/JavaScript/JSP/XML.

# CHAPTER THREE

# DATABASE DESIGN

The database that will be required by MTTOS will be written in PostgreSQL. Once the database has been created using PostgreSQL, all interfacing with the database will be done from Java. One must note that all interfaces as seen by the users of the system will be through pages with HTML forms generated from JSP and JavaServlets.

## 3.1 Data Analysis

The data for designing and implementing the schema of the database depends on eight entities: Books table, Member table, Movie table, Movie_Time table, Room table, Seat table, Password table and Ticket table. All the input data will be checked by using JavaServlet or JavaScript when the data is processed. The tables of Movie, Movie_Time, Room and Seat are connected by the relation of movie number. The tables of Books and ticket are connected by the relation of ticket number. To get the customer purchase information, we use member_ID to search it.

## 3.2 Database Schema Conceptual Model- ER Diagram

All the entities and relations used in MTTOS are described in Figure 2 E-R Diagram,

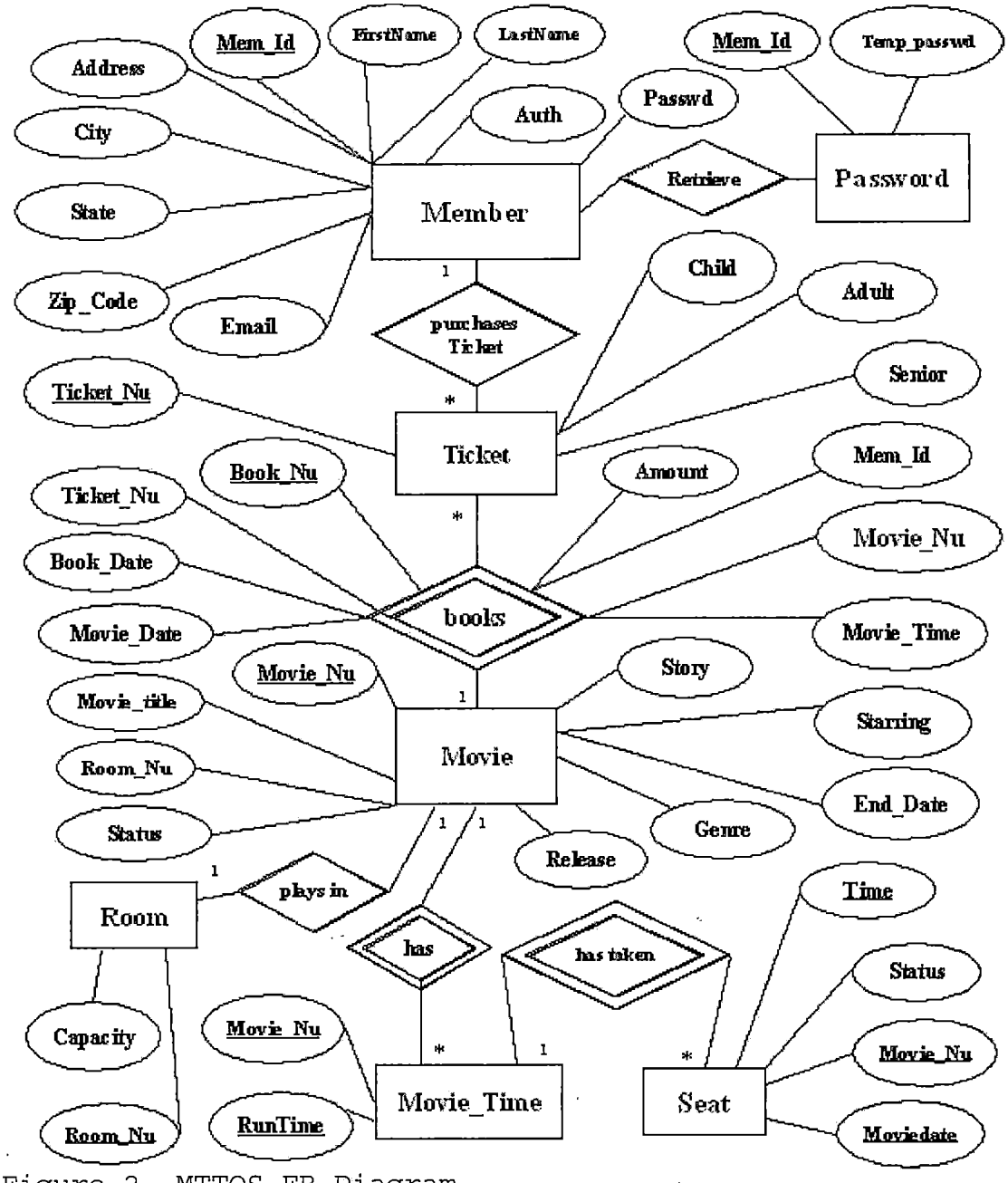


Figure 2. MTTOS ER Diagram

## 3.3 Database Schema Logical Model - Relational Schema

The conceptual model ER diagram maps into the following relational table design. In the following tables (underlined fields indicate the primary key).

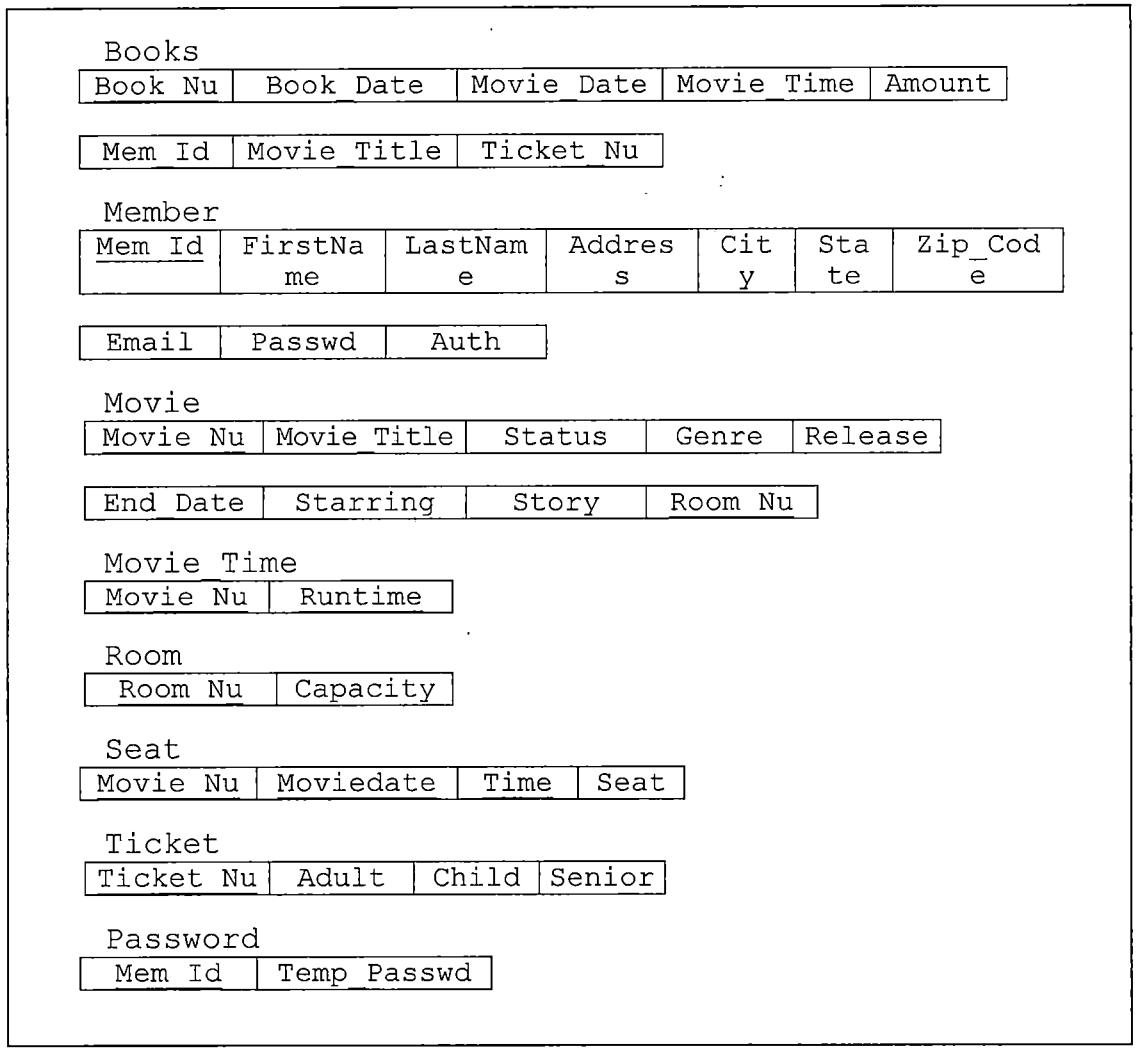


Figure 3. MTTOS Database Relational Schema

## 3.4 Data Type and Detail

The logical model establishes the following detailed design in PostgreSQL database. The following tables describe data type, length, primary key, null or non-null keys, and extra information, such as auto_increment.

Table 1. Structure of Table Books

| field | type | null | key | default | extra |
|---|---|---|---|---|---|
| Book_Nu | varchar(10) | | PRI | | |
| Book_Date | Date | | | | |
| Movie_Date | varchar(10) | | | | |
| Movie_Time | varchar(6) | | | | |
| Amount | varchar(10) | | | | |
| Mem_Id | varchar(20) | Yes | FOR | | |
| Movie_Title | varchar(36) | | FOR | | |
| Ticket_Nu | varchar(10) | | FOR | | |

Table 2. Structure of Table Member

| field | Type | null | key | default | extra |
|---|---|---|---|---|---|
| Mem_Id | varchar(20) | | PRI | | |
| FirstName | varchar(20) | | | | |
| LastName | varchar(20) | | | | |
| Address | varchar(40) | Yes | | | |
| City | varchar(20) | Yes | | | |
| State | varchar(2) | Yes | | | |
| Zip_Code | varchar(5) | Yes | | | |
| Email | varchar(30) | | | | |
| Passwd | varchar(20) | | | | |
| Auth | varchar(1) | | | | |

Table 3. Structure of Table Movie

| field | Type | null | key | default | extra |
|---|---|---|---|---|---|
| Movie_NU | varchar(10) | | PRI | | |
| Movie_Title | varchar(36) | | | | |
| Status | varchar(1) | | | | |
| Room_Nu | varchar(2) | | FOR | | |
| Genre | varchar(2) | | | | |
| Release | varchar(2) | Yes | | | |
| End_date | varchar(2) | Yes | | | |
| Starring | varchar(2) | Yes | | | |
| Story | Text | | | | |

Table 4. Structure of Table Movie_Time

| field | Type | null | key | default | extra |
|---|---|---|---|---|---|
| Movie_NU | varchar(10) | | PRI | | |
| Runtime | varchar(6) | | PRI | | |

Table 5. Structure of Table Room

| field | Type | null | key | default | extra |
|---|---|---|---|---|---|
| Room_Nu | varchar(2) | | PRI | | |
| Capacity | Int | | | | |

Table 6. Structure of Table Seat

| field | Type | null | key | default | extra |
|---|---|---|---|---|---|
| Movie_NU | varchar(10) | | PRI | | |
| Moviedate | varchar(10) | | PRI | | |
| Time | varchar(6) | | PRI | | |
| Seat | Int | Yes | | | |

Table 7. Structure of Table Ticket

| field | Type | null | key | default | extra |
|---|---|---|---|---|---|
| Ticket_Nu | varchar(10) | | PRI | | |
| Adult | varchar(2) | Yes | | | |
| Child | varchar(2) | Yes | | | |
| Senior | varchar(2) | Yes | | | |

Table 8. Structure of Table Password

| field | Type | null | key | default | extra |
|---|---|---|---|---|---|
| Mem_Id | varchar(20) | | PRI | | |
| Temp_Passwd | varchar(10) | | | | |

CHAPTER FOUR

PROJECT IMPLEMENTATION

The design of MTTOS aims to perform 12 main functions for 3 different prospected users. The following figure is the Use Case Diagram of this project.

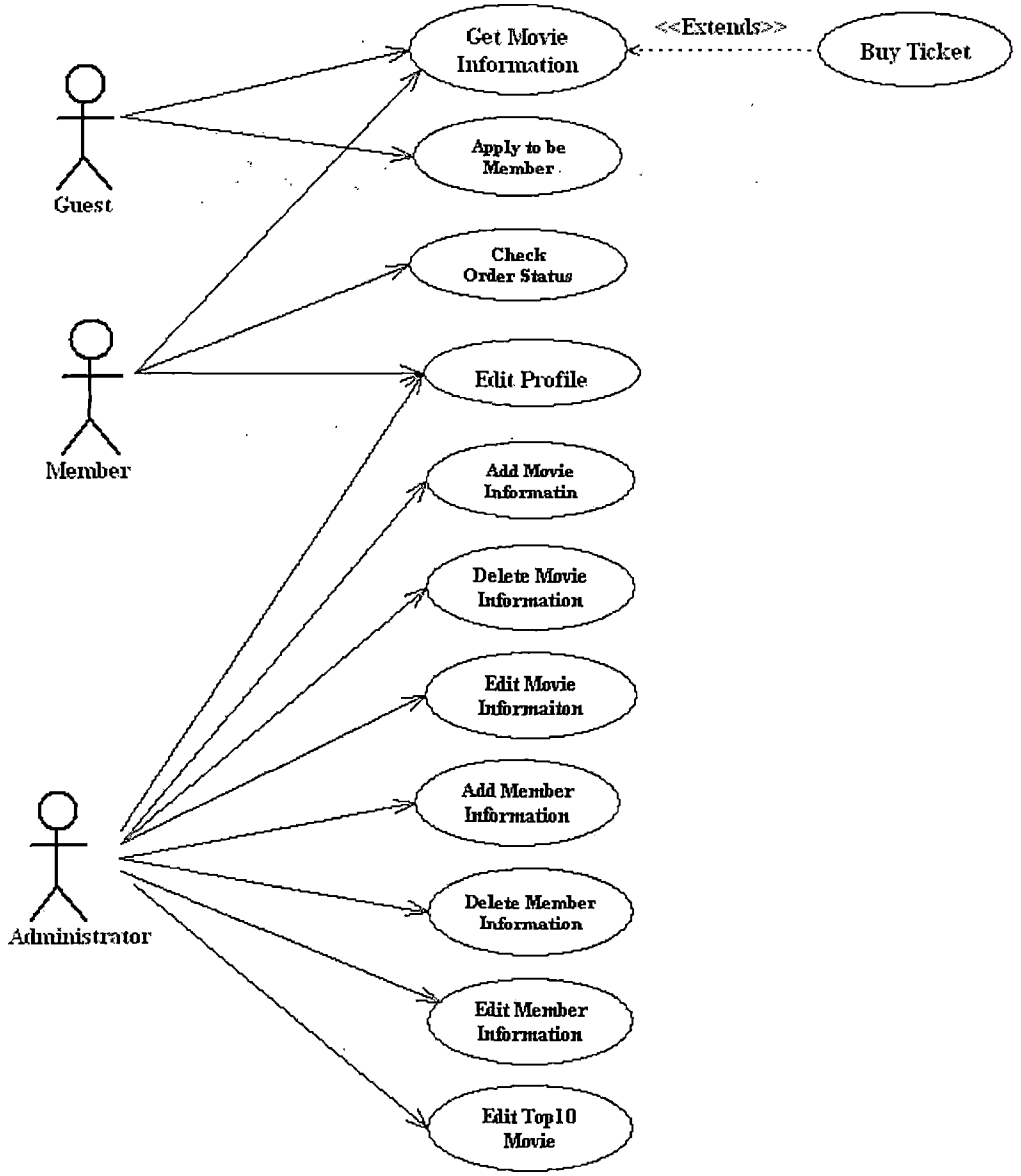


Figure 4. Use Case Diagram

## 4.1 User Interface Design

User interfaces will be established on the web and therefore it can use all user interfaces provided by the web browser including all plug-ins and any added functionality that the browser may possess. Any standard web browser, such as Microsoft Internet Explorer or Netscape Navigator/Communicator can provide the user interface to the system. There are two types of user interfaces. 1) Static user interfaces: the interface will be static for all people browsing the site, regardless of the access rights of the person. They will be in the form of HTML forms or pages that have HTML Data and pictures on them. 2) Dynamic user interface: the interfaces will be generated dynamically on the server side using JSP and JavaServlets. These user interfaces will provide information tailored to the user who is logged in.

## 4.2 Graphical User Interface and Description

### 4.2.1 MTTOS Home Page

This page is the starting page for all people (Guest, Member and Administrator) who are going to be using this software product. It shows all required logos. To reduce the download time for graphics, we write an effective caching down by browser. All the users can click on the

movie title link in the page; then the Servlet will forward to the movie information page. If one comes to this page, one will be able to find information about the club that exists at the movie theater. Membership and contact information will be provided for this organization. If guests become members, they will be able to learn what they can expect from the club and what would be expected from them. If the members forget their password, they can click on the forget password link and provide their user id to retrieve their password from the database by automatically generated mail. The member or administrator can log in by providing a user id and a password on this page. The login servlet will verify the user id and password. If it is correct, it forwards to a JSP page, which will show the main page of the member or administrator.

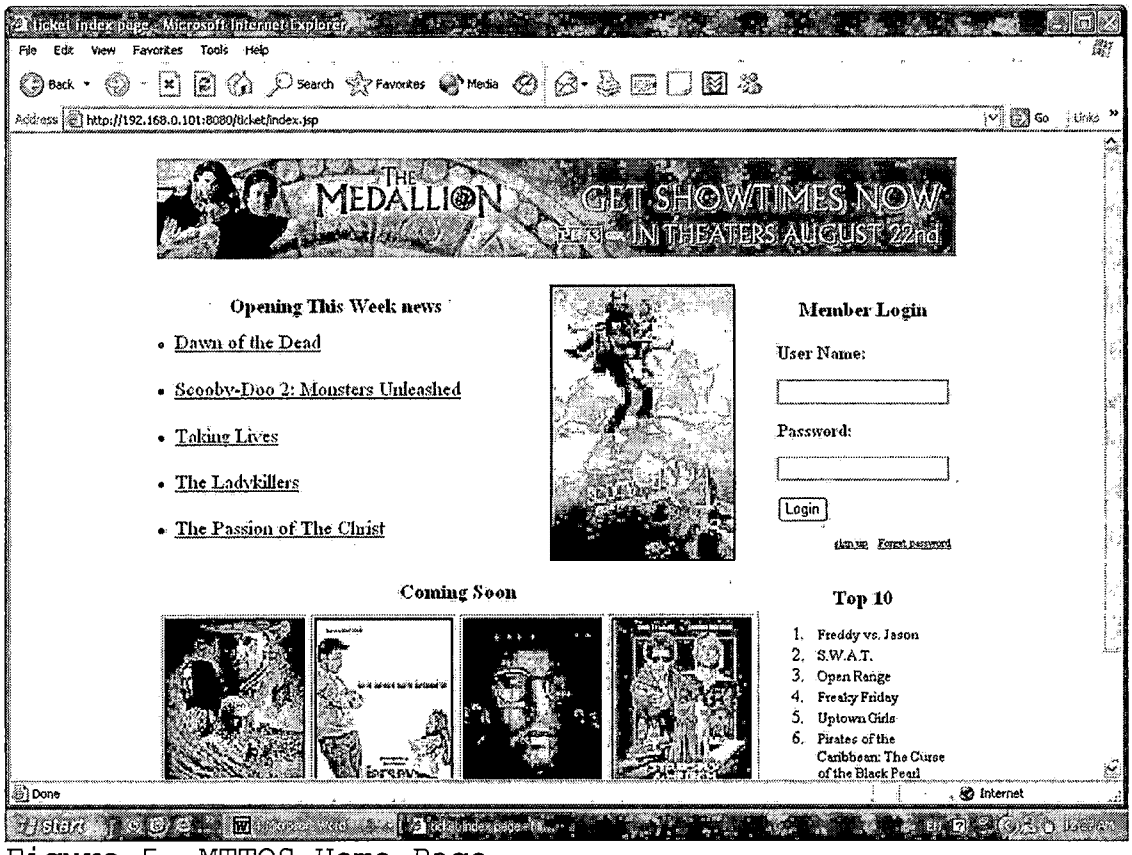

Figure 5. MTTOS Home Page

### 4.2.2 Movie Information Page

This page gives information about the movie. This page is essentially supposed to serve as an infomercial page to encourage the guest to order the ticket. All users can click on the movie time to buy the movie ticket.

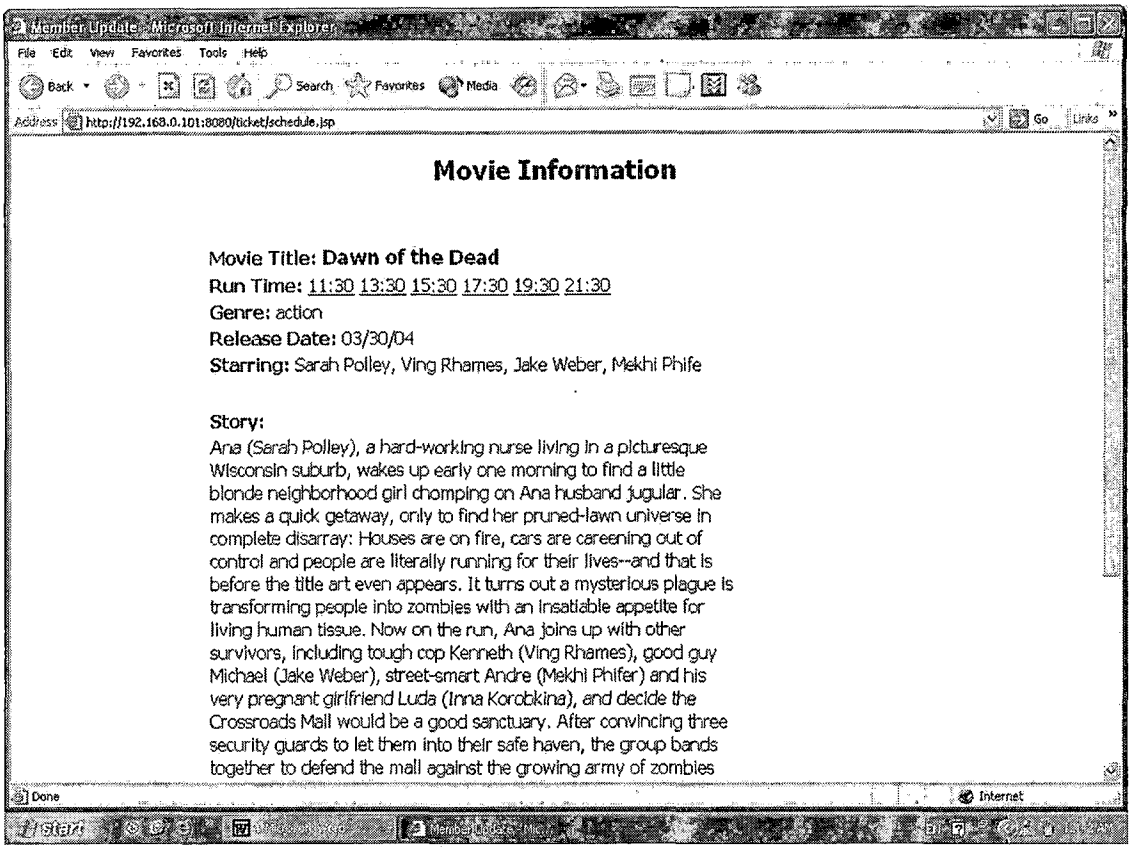

Figure 6. Movie Information Page

### 4.2.3 Purchase Movie Ticket Page

On this page, users select on movie's viewing date and quantity of each type of ticket.

Then users click on the Buy button to go forward to submit payment page, or they click on the Cancel button to go back to MTTOS home page. After the users click on the Buy button the server will check the information that the user selected. If it is incorrect, the browser will display an error message to this page. If it is correct, it will be forwarded to the destination page.

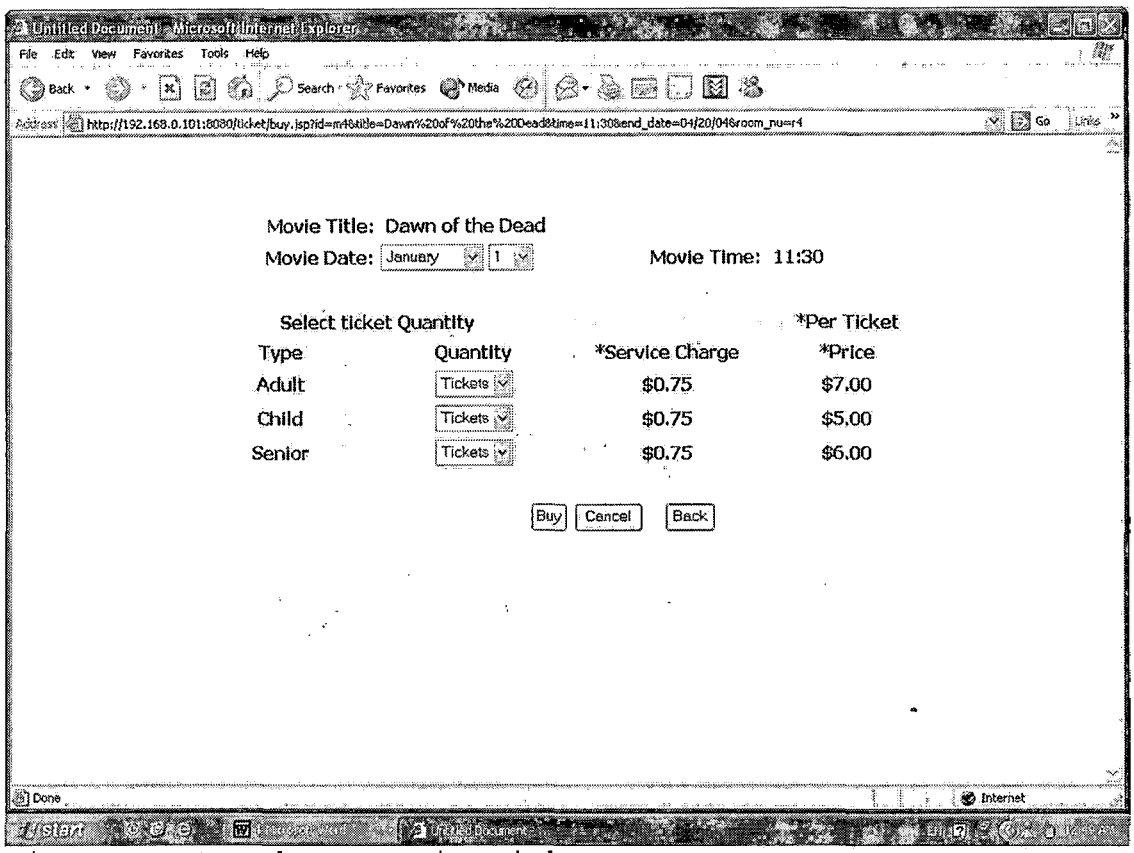

Figure 7. Purchase Movie Ticket Page

### 4.2.4 Submit Payment Page

This page shows the user's order information. A user can double-check to see if their order status is correct or not. If it is incorrect, users can click on the back button to go back to the purchase movie ticket page. If it is correct, users need to input their payment information and the server will validate user's credit card information. If it is invalidated, the browser will show an error message in this page. If it is validated, the

user will be charged the amount shown in the order information, and be sent to an order confirmation page.

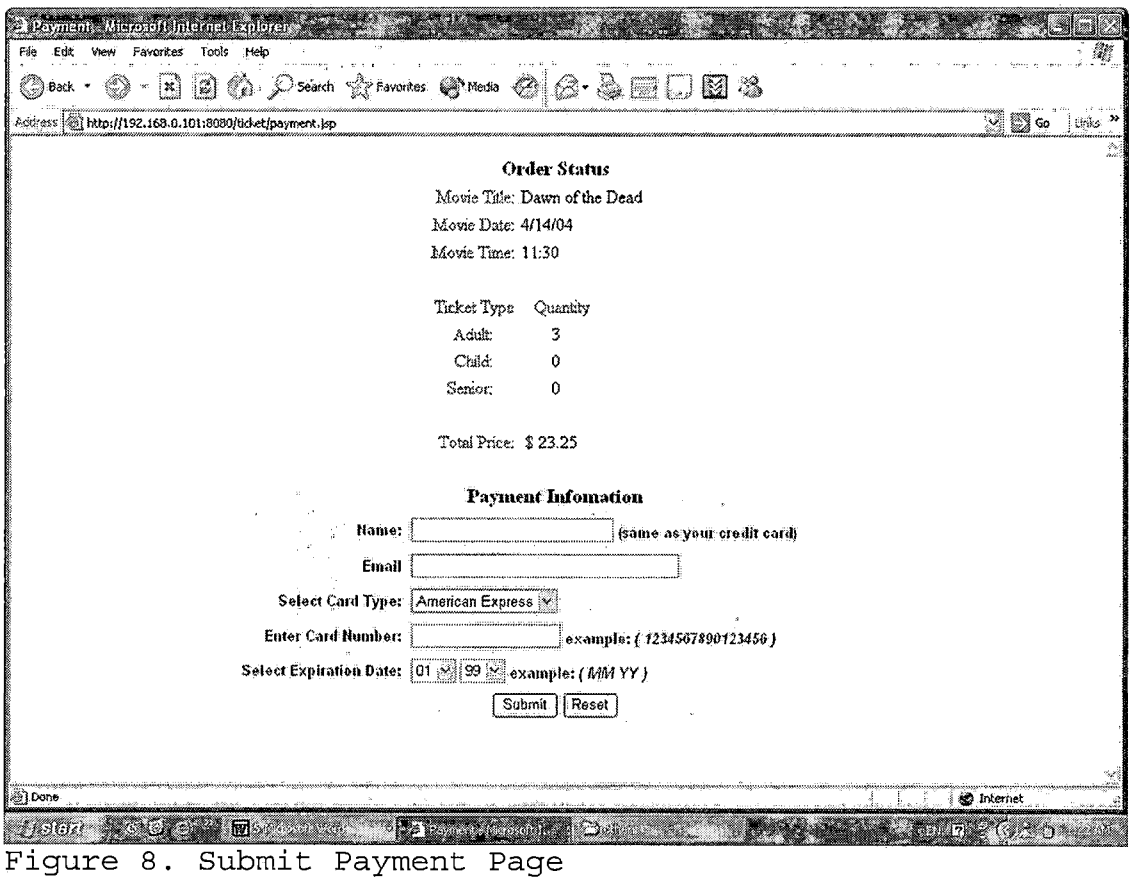

Figure 8. Submit Payment Page

### 4.2.5 Payment Confirmation Page

This page shows the user's credit card information. A user is availed to double-check his/her payment data. If it is incorrect, users can either click on the edit payment button to go back to the submit payment page, or click on the cancel button to go back to MTTOS home page. If it is correct, the user can click on the submit payment button to process the order. At the same, all the purchase

information will be saved in database. The user will be sent to an order confirmation page.

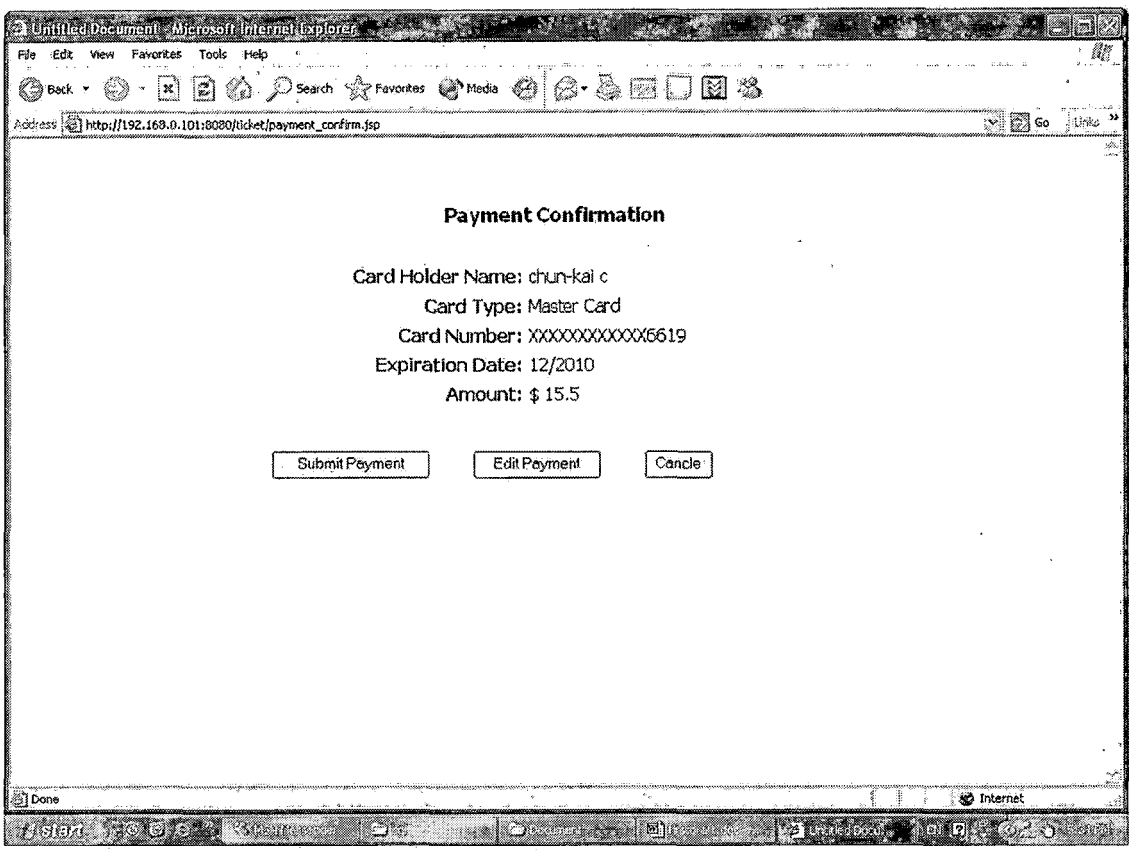


Figure 9. Payment Confirmation Page

## 4.2.6 Order Confirmation Page

This page shows the customer's booking number and order information. Users can click on the print page button to print their order information or click on the home button to go back to MTTOS home page.

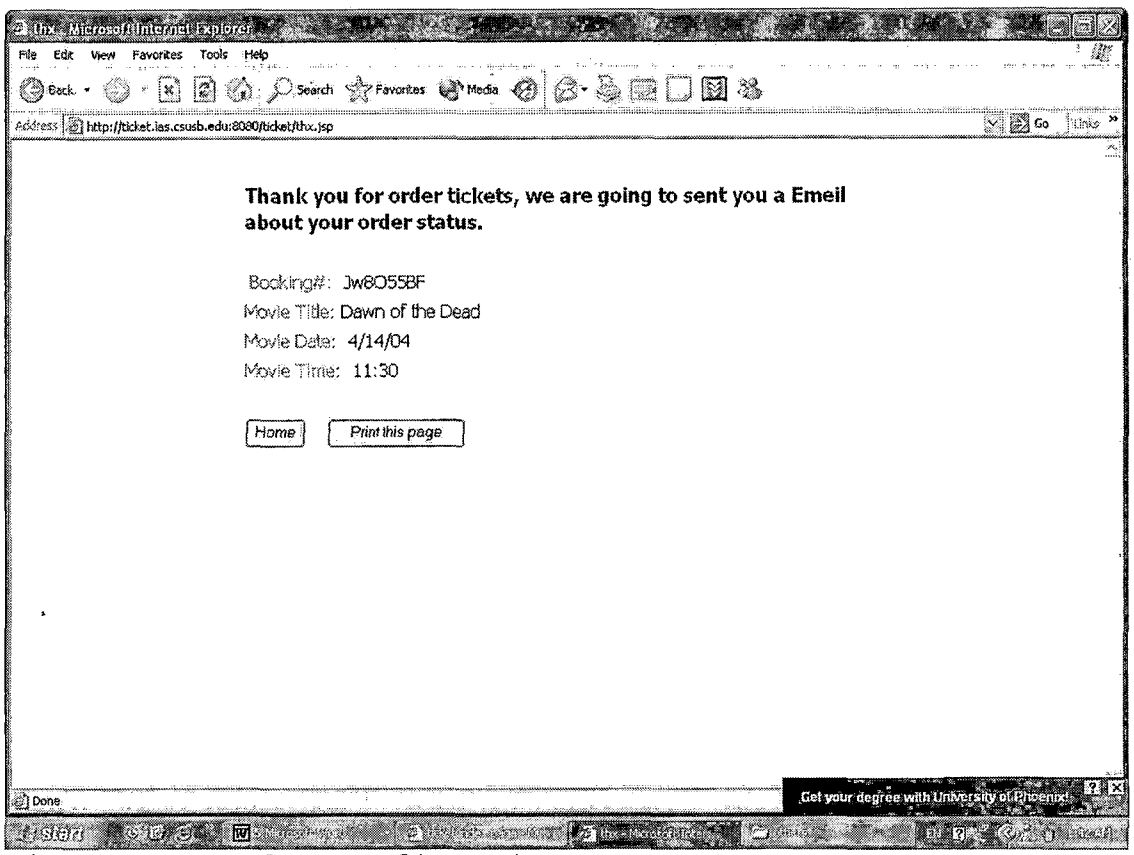

Figure 10. Order Confirmation Page

### 4.2.7 Member Register Page

A user needs to fill up this form to be a member of the movie theater. If the input data is incorrect, for example, if passwords don't match, then the browser will display an appropriate error message. Otherwise, the MTTOS will generate an account for this user and forward it to the welcome sign in page.

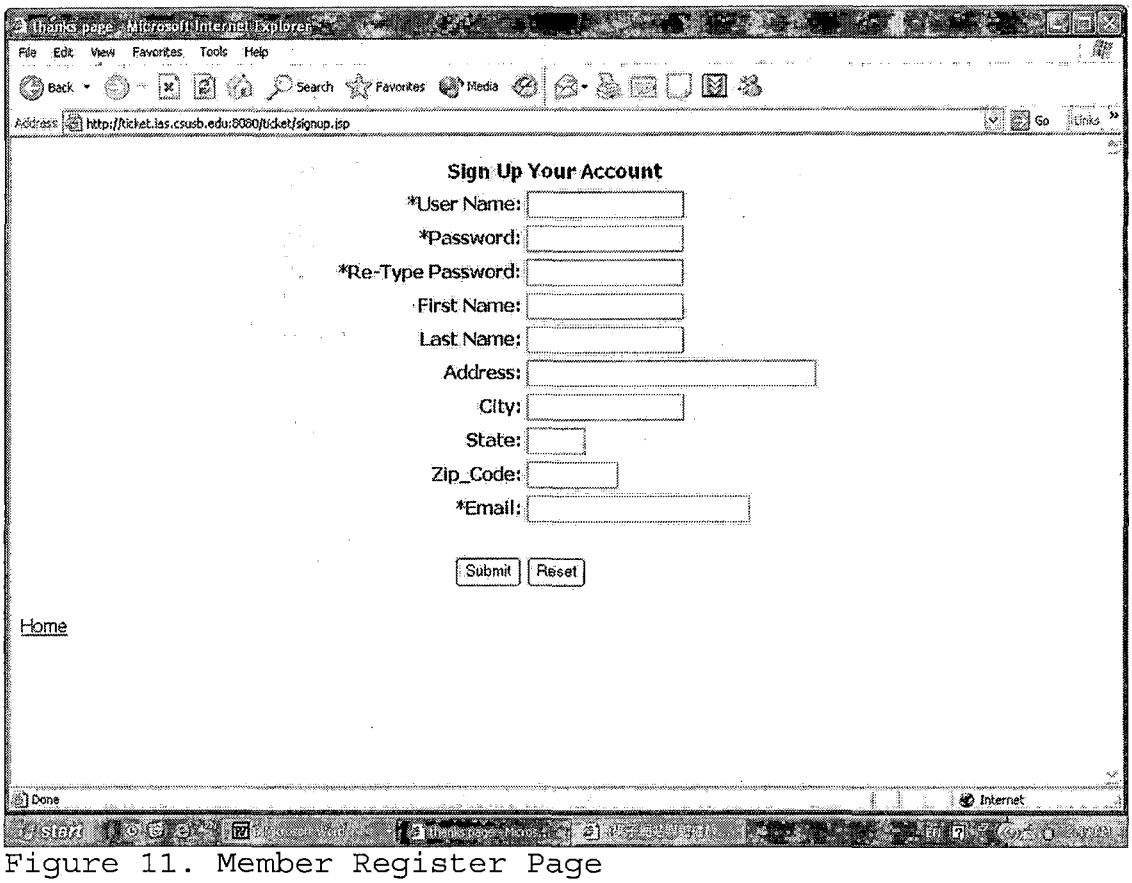


Figure 11. Member Register Page

### 4.2.8 Welcome New Member Page

New members click on the sign in link to go back MTTOS home page and log in to member's main page.

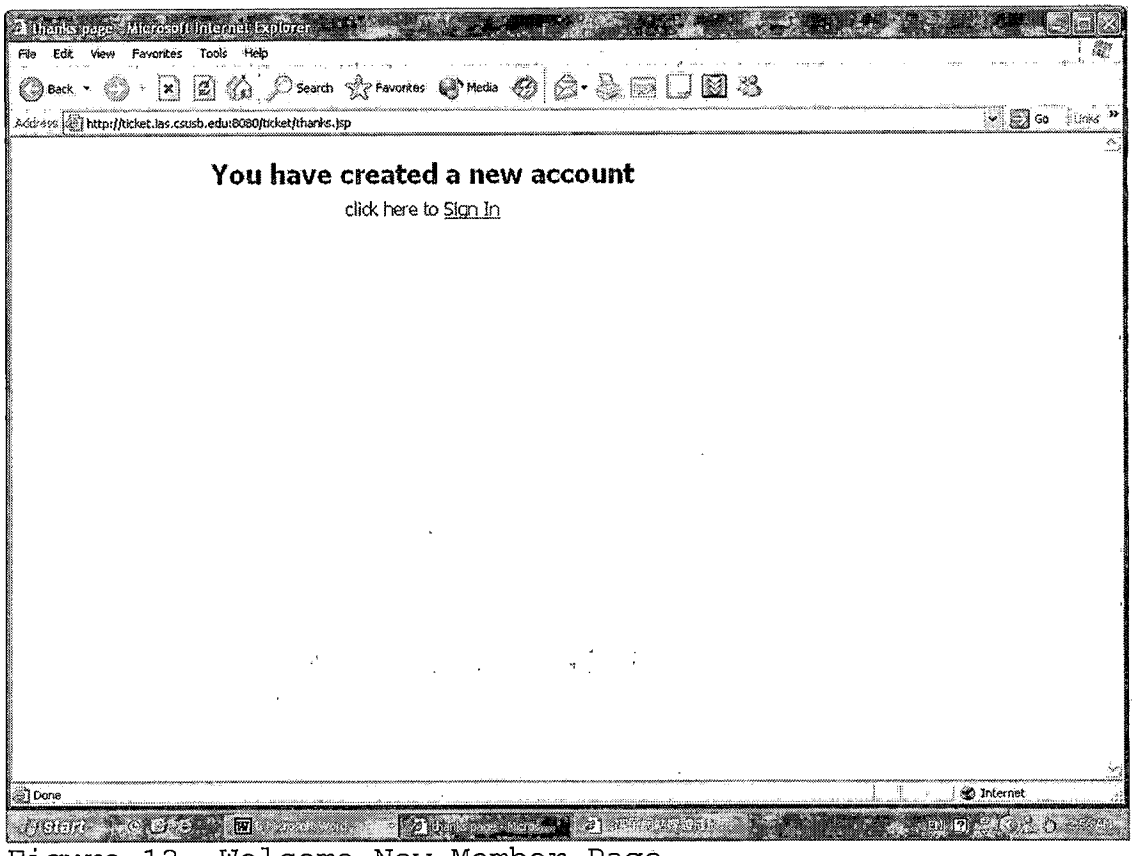

Figure 12. Welcome New Member Page

### 4.2.9 Forget Password Page

On this page, members provide their user ID to modify their password. If the user ID is correct, the Servlet will generate a random number as member's temporary password and sent a url link by email. Members can click on the link to login to system to modify their password. After members modify their password, the temporary password will be deleted from database automatically.

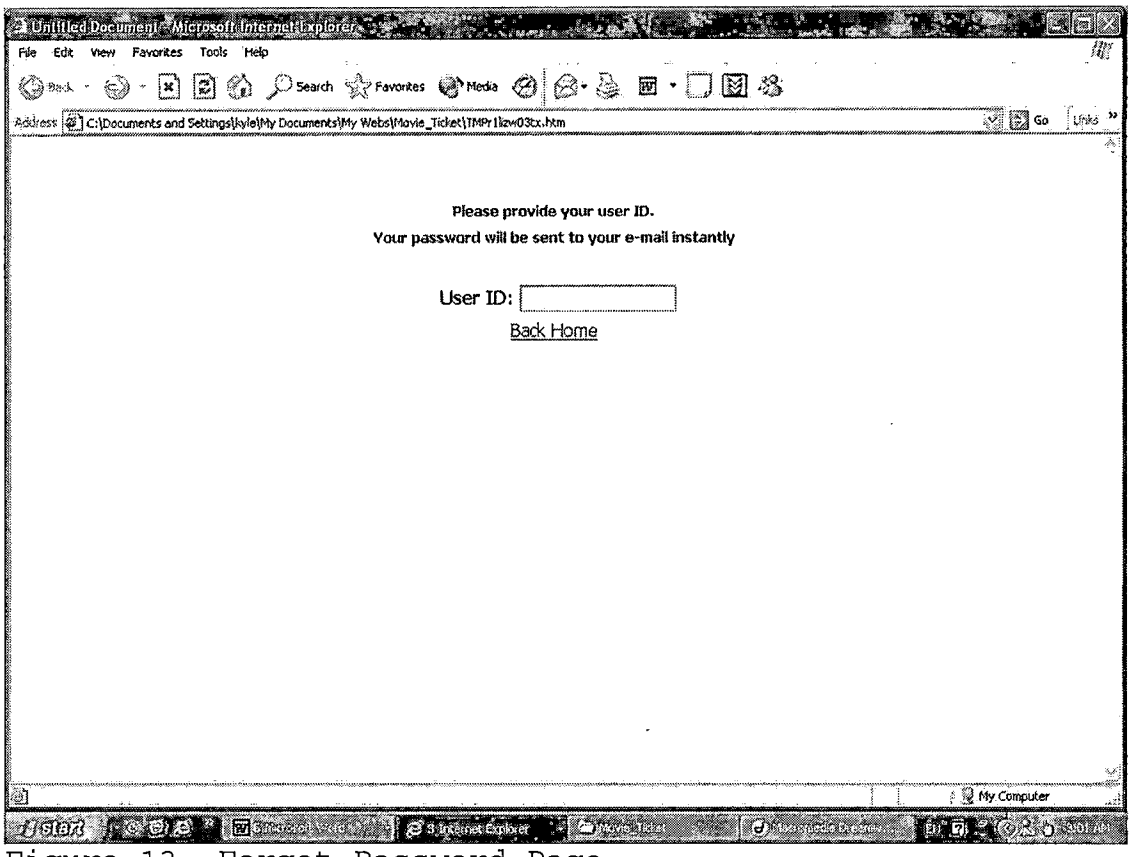

Figure 13. Forget Password Page

4.2.10 Member's Main Page

This page (Fig. 15) is similar to the MTTOS home page. The difference is that the login function is replaced by a menu of member options. Members can click on the options of menu bar to check their order status, manage their personal profile or log out the system.

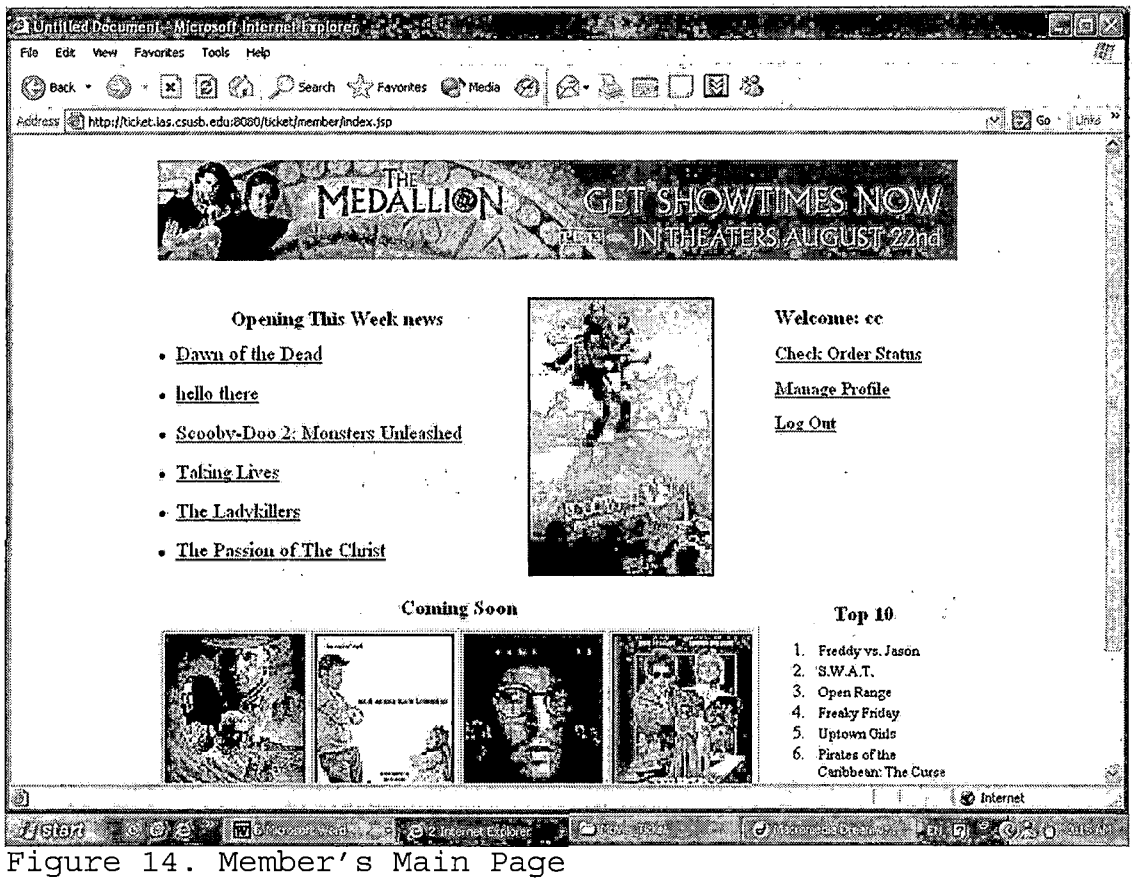

Figure 14. Member's Main Page

## 4.2.11 Member Check Order Status Page

This page shows customer's current order information.

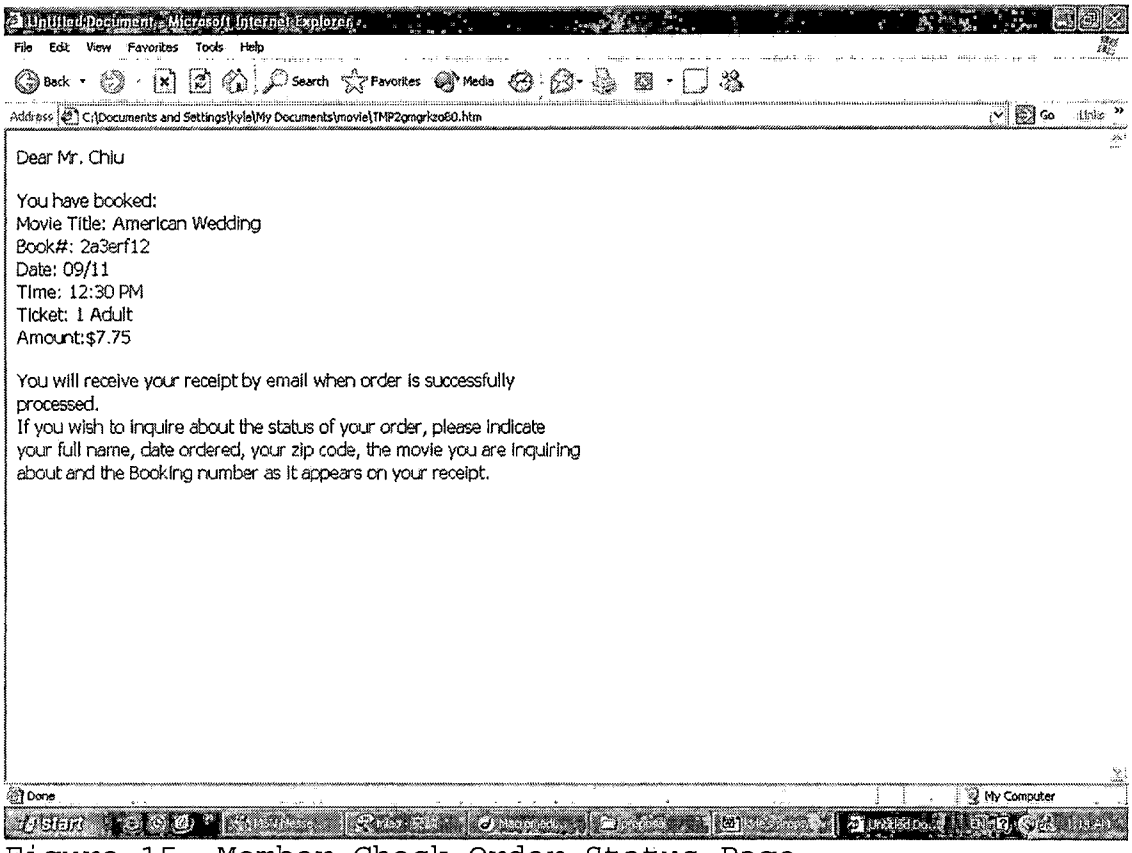

Figure 15. Member Check Order Status Page

## 4.2.12 Member Manage Profile Page

Members can update their information in this page. On this update page, it will not show the member's password in the input text area because of security problem. If users leave them empty, the system won't update their password. After they update their information the link will send back the same page with their new information.

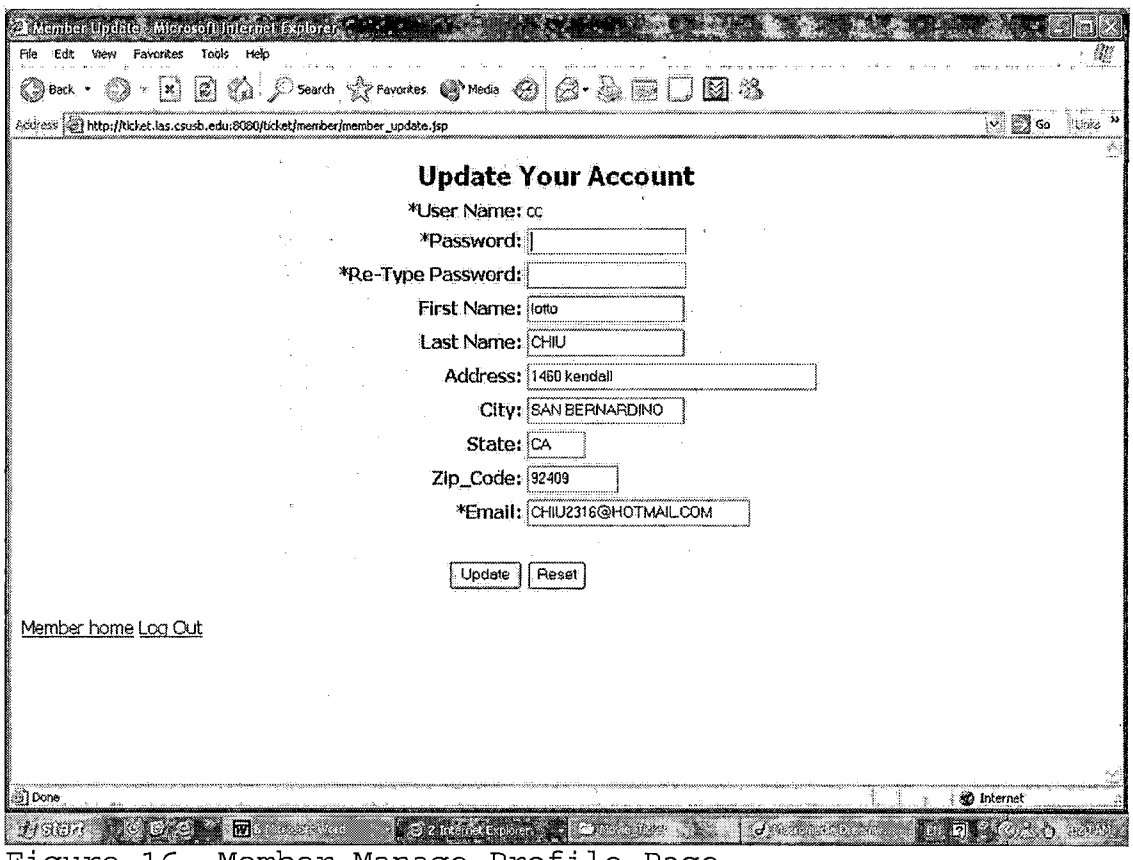

Figure 16. Member Manage Profile Page

### 4.2.13 Administrator Main Page

This page will be the same as MTTOS home page, and users can invoke the same functions as MTTOS home page. The only difference is that the login function becomes the administrator's menu bar. The administrator can click on the options of the menu bar to add, delete or edit a movie's information; add, delete or edit member's information; edit the top10 movie; manage their personal profile; or log out of the system.

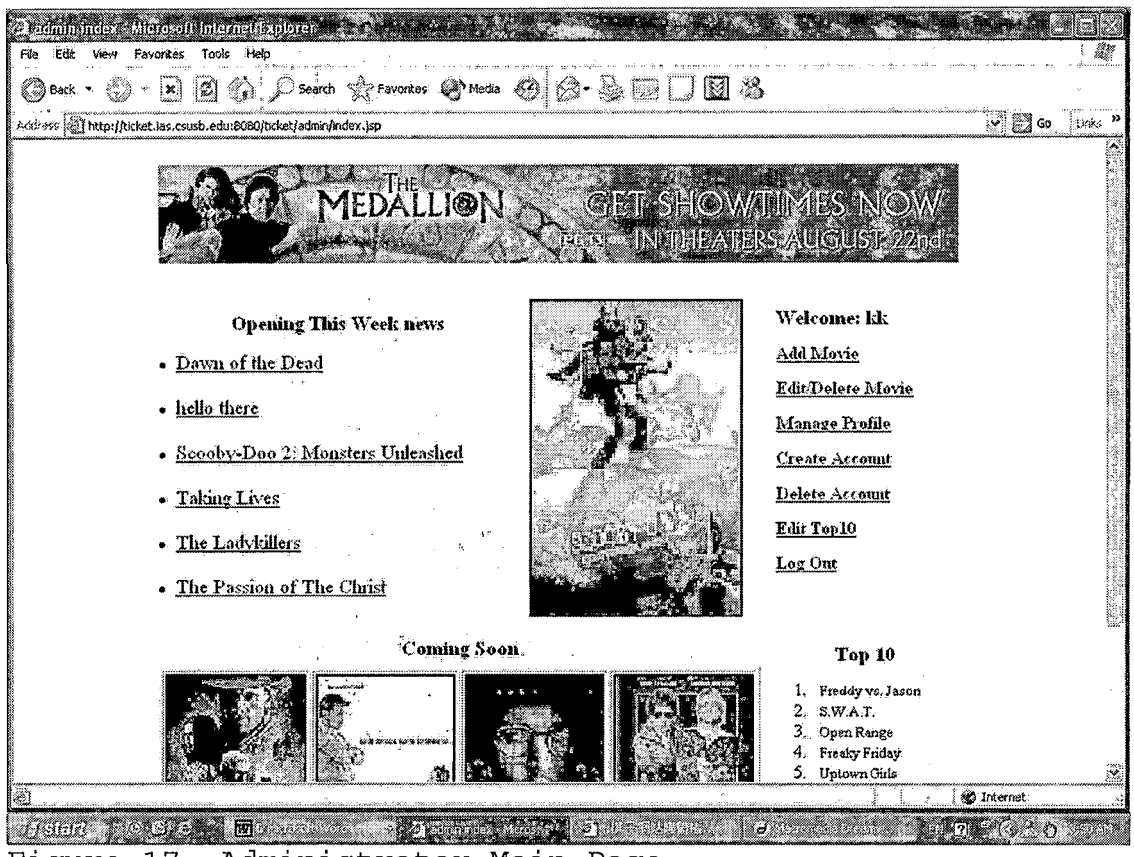

Figure 17. Administrator Main Page

### 4.2.14 Add Movie Information

The administrator creates new movie offerings in this page. The movie number can't be null. There are three options of movie status, which are: "opening this week," "coming soon," and "preview." If you chose the movie status to be either "opening this week" or "coming soon," the movie title will be shown in the opening this week box or the coming soon box directly. The movie status of preview saves the movie data in the database. Once the

system completes the process, it forwards the administrator to the Delete/Edit Movie Page.

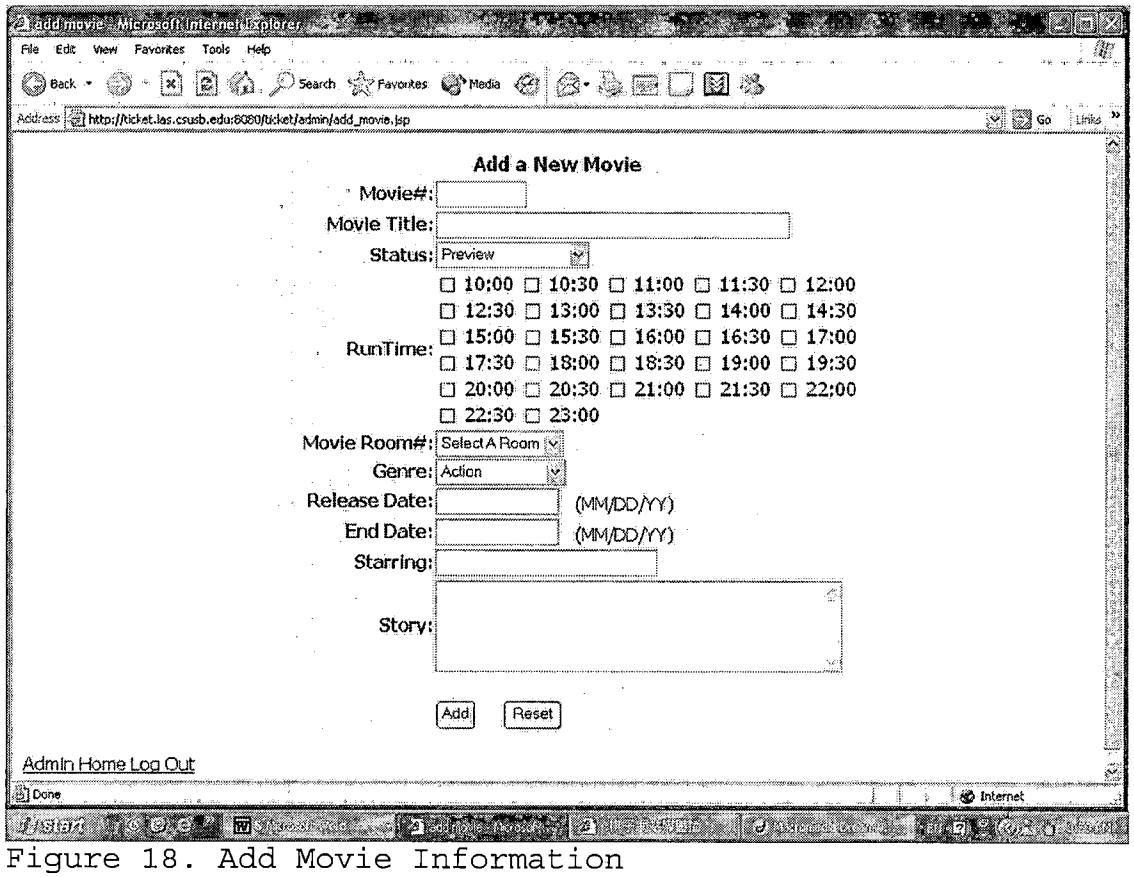

Figure 18. Add Movie Information

## 4.2.15 Delete/Edit Movie Page

This page lists all of the movies saved in the database. If you click on the edit link, the page will be forwarded to the Edit Movie Information Page. Click on the movie delete box for each movie you want to delete, and then click on the delete button; the movie will be automatically deleted from the database. Once the system

completes the process, it forwards the administrator to this page again.

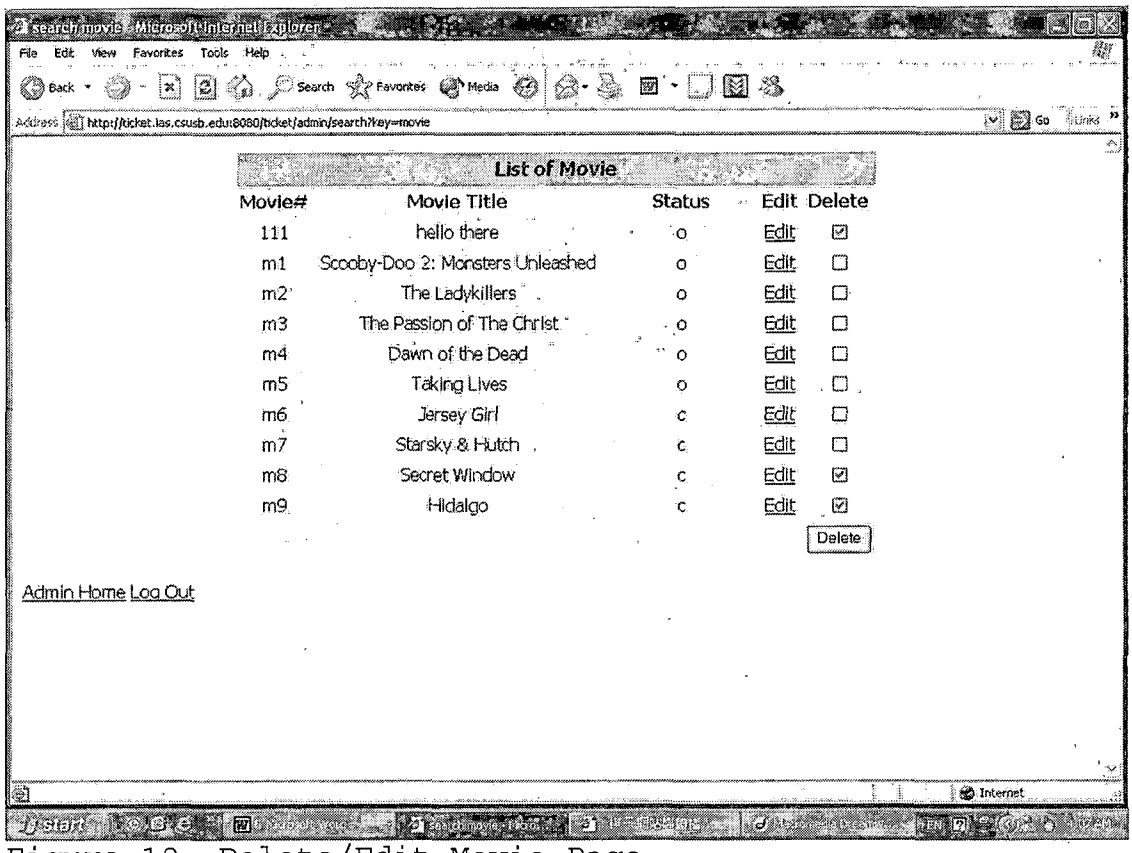

Figure 19. Delete/Edit Movie Page

## 4.2.16 Edit Movie Information Page

The administrator can update the movie information in this page. After the administrator updates the movie information, the system will forward it back to the same page with its new movie information.

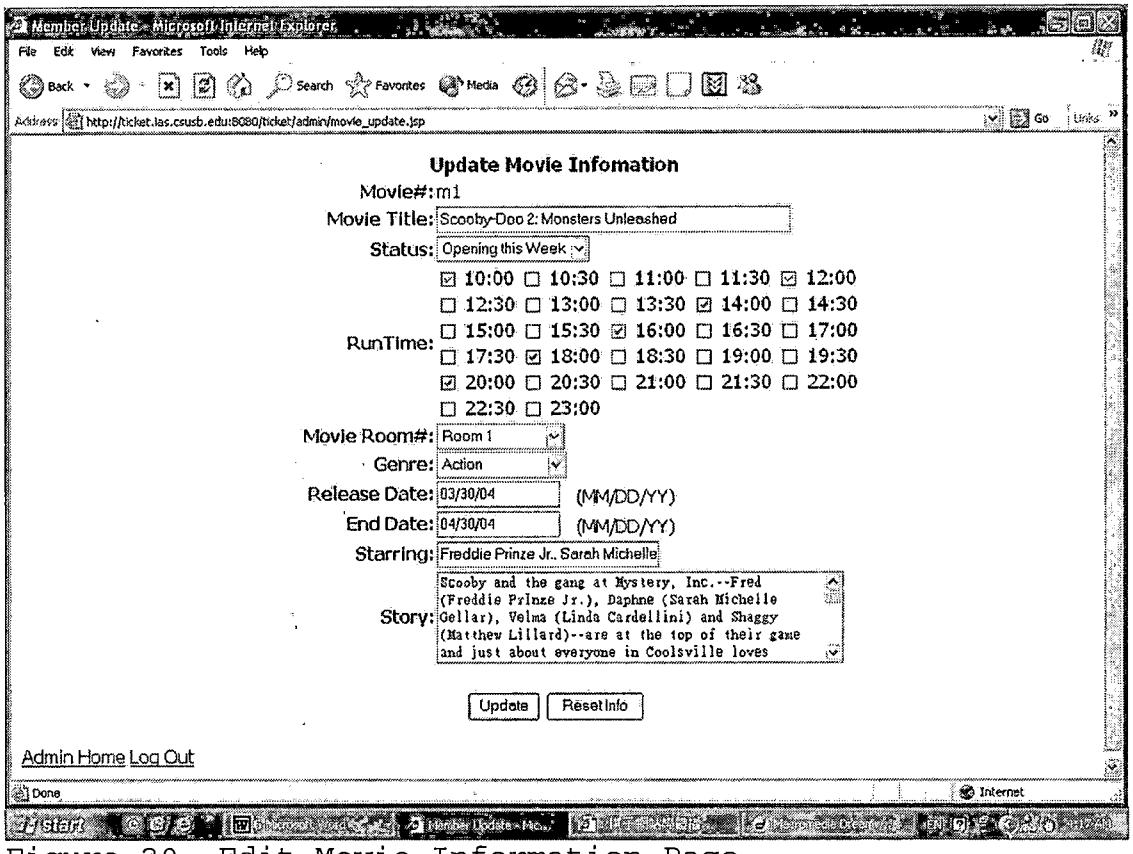

Figure 20. Edit Movie Information Page

### 4.2.17 Add Account Page

The administrator creates either member or administrator's new account in this page. If the input data is incorrect, for example, the userID or email is null or password and re-type password are not matched, then the browser will display an appropriate error message. Otherwise, the MTTOS will generate a new account and forward to the Delete/Edit Account Page.

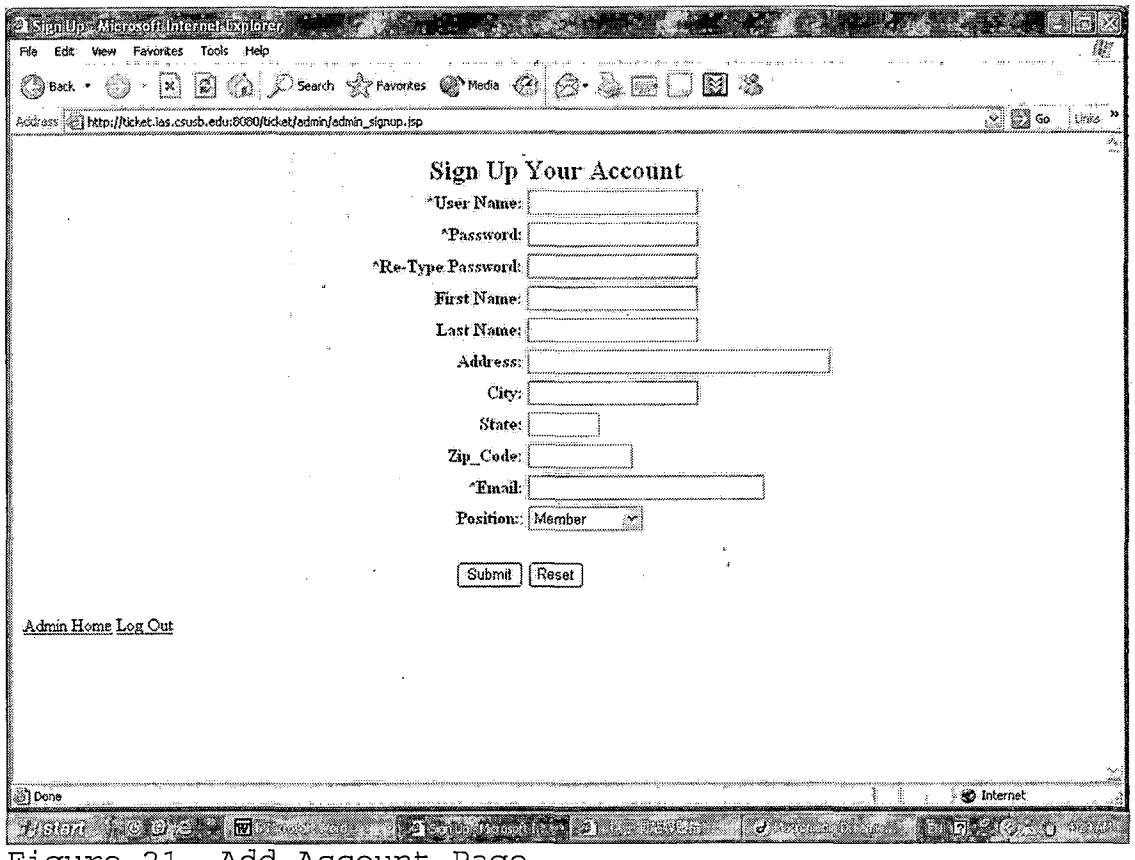

Figure 21. Add Account Page

## 4.2.18 Delete/Edit Account Page

This page will list all of the user ID and their names which are saved in the database. If you click on the edit link, the page will be forwarded to Edit Account Information Page. Click on the account delete box which you want to delete, and then click on the delete button; the movie will be automatically deleted from the database. Once the system completes the process, it forwards the administrator to this page again.

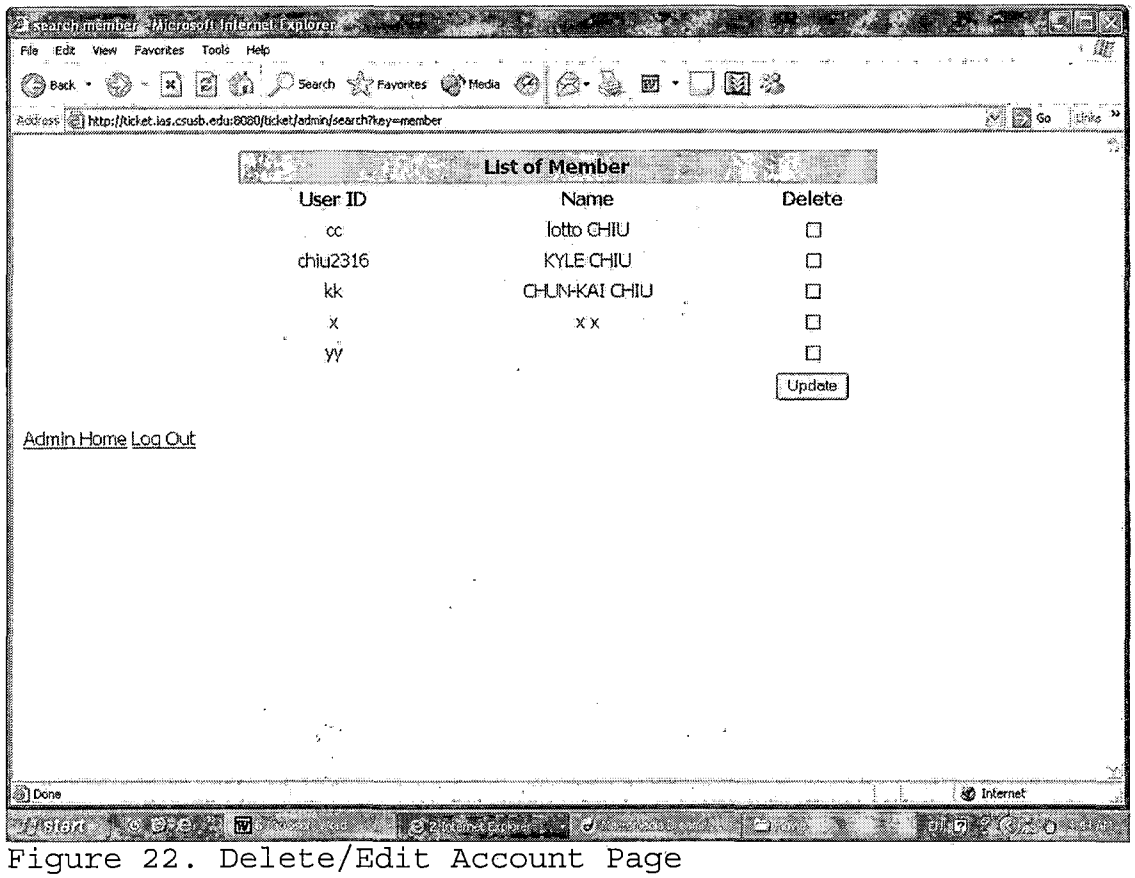

Figure 22. Delete/Edit Account Page

## 4.2.19 Edit Account Page

The administrator can edit the entire user's information in this page. Once the system completes the process, it forwards the administrator to the Delete/Edit Account Page with its new account information.

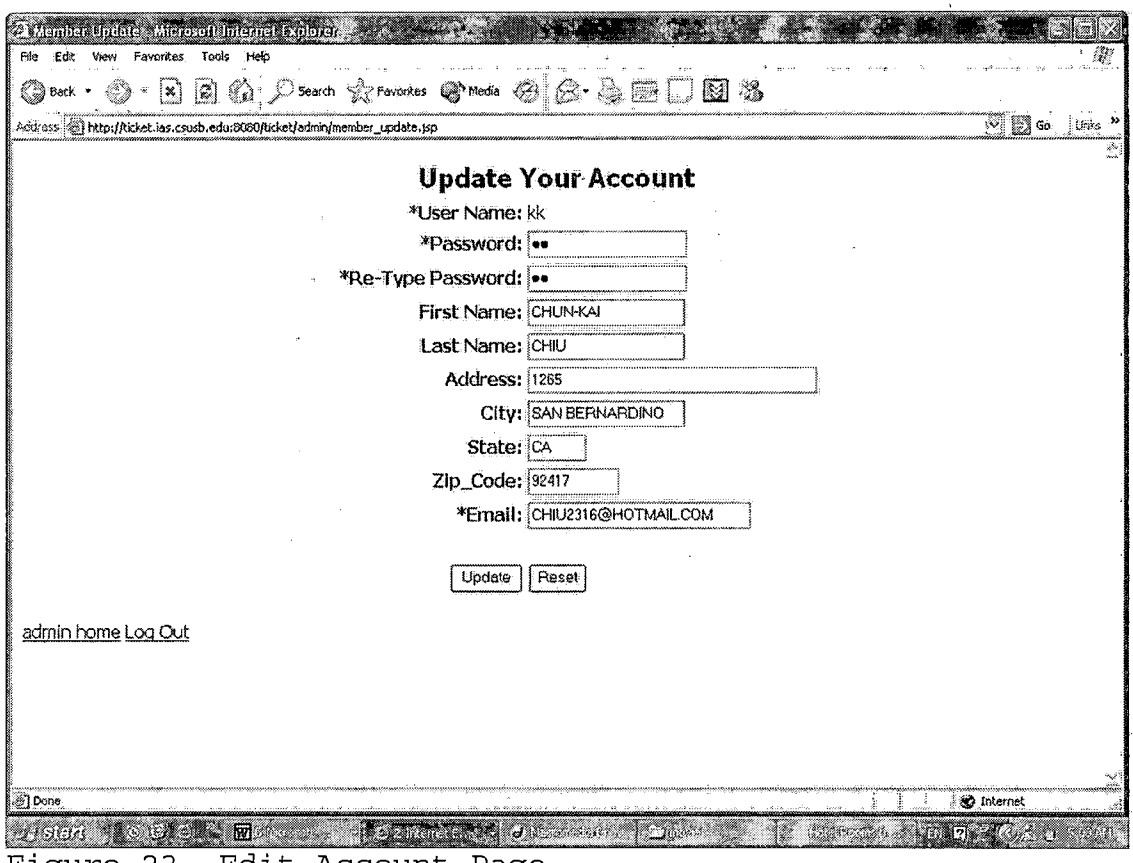

Figure 23. Edit Account Page

## 4.2.20 Administrator Manage Profile Page

The administrator can update his/her information in this page. In this update page, it will not show the password and re-type password in the input text area because of security problem. If the administrator leaves them empty, the system won't update his/her password. After administrator updates his/her information the link will send back to the same page with his/her new information.

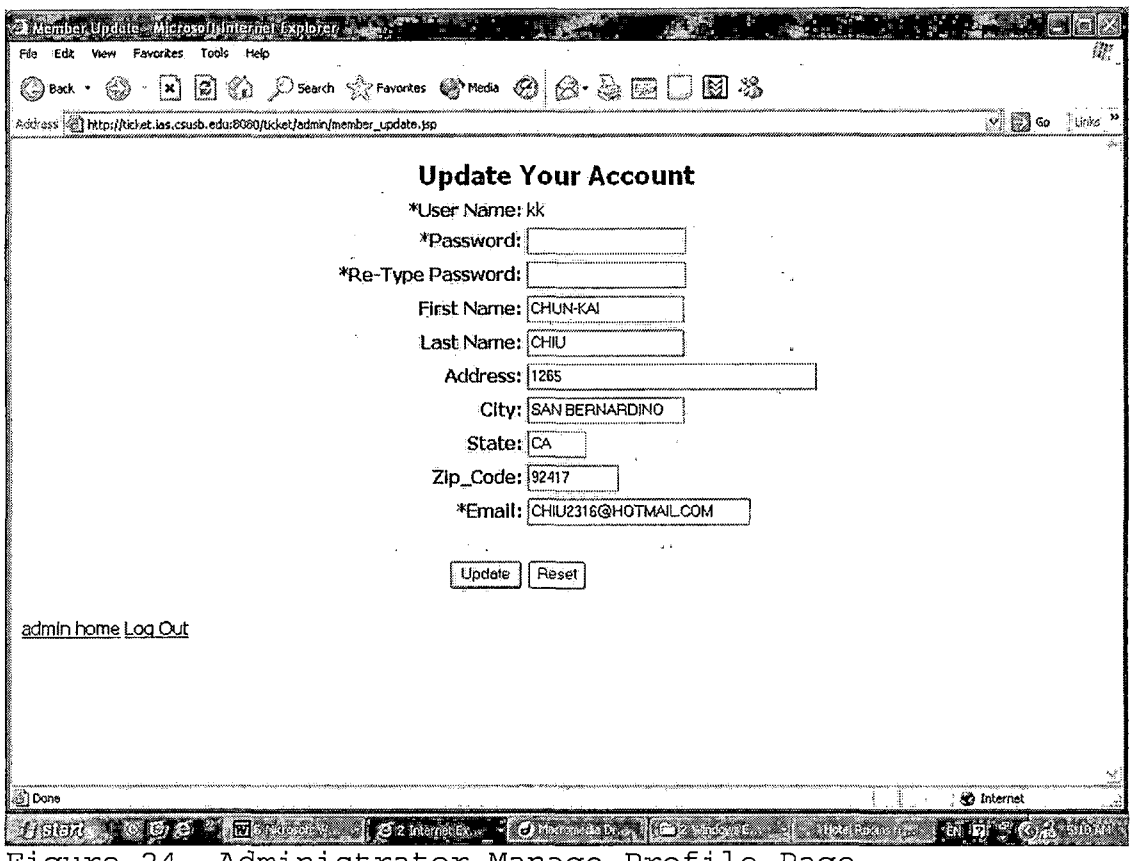

Figure 24. Administrator Manage Profile Page

### 4.2.21 Update Top10 Movie Information Page

The administrator can update Top10 Movie Information in this page.

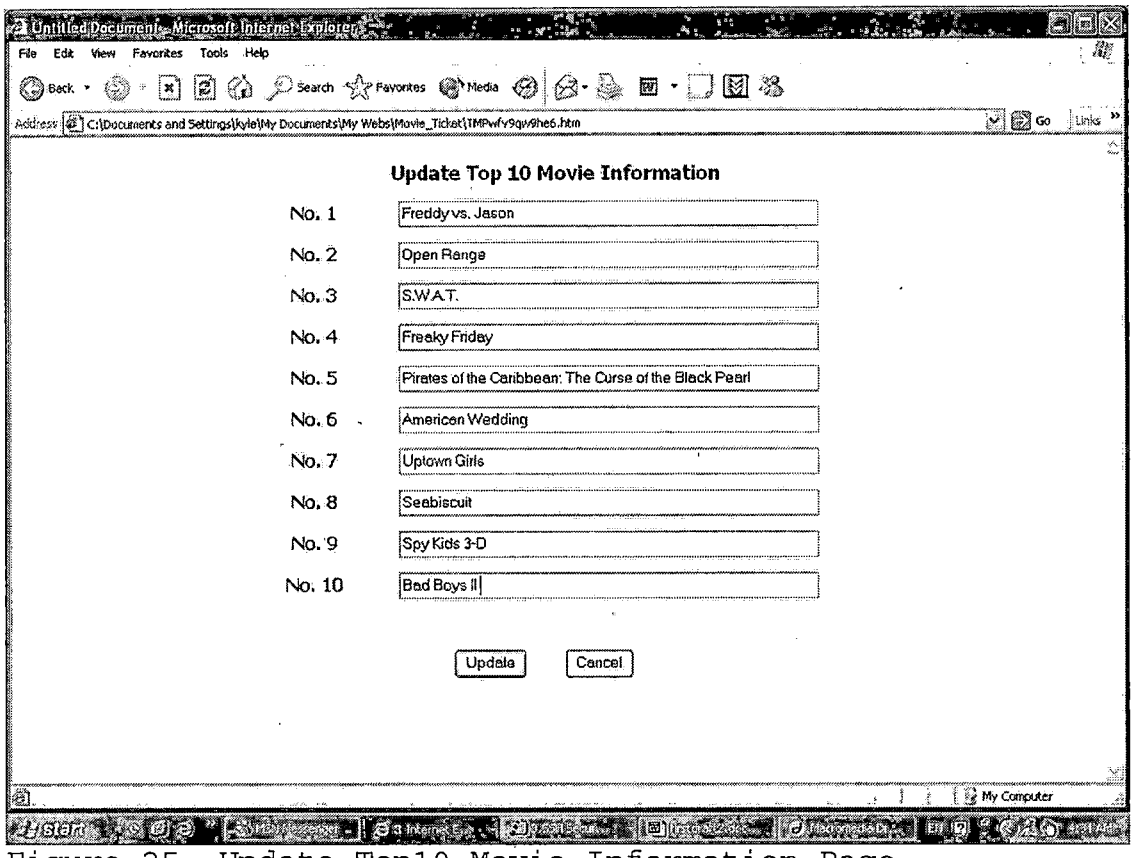

Figure 25. Update Top10 Movie Information Page

# CHAPTER FIVE

# SYSTEM VALIDATION

To validate a system, just having an understanding of computers and software systems is not adequate. It is essential to fully understand the process and the equipment that is being validated. Validation testing is a concern that overlaps with integration testing. Ensuring that the application fulfils its specification is a major criterion for the construction of an integration test. Validation testing also overlaps to a large extent with system testing, where the application is tested with respect to its typical working environment.

## 5.1 Unit Test

The Unit test presents the basic level of testing on all the individual components. The individual components include the object, the class, and the program in the system. The following table shows the results of the unit test for the MTTOS.

Table 9. The Unit Test Results

| Unit Test | Tests Performed | Results |
|---|---|---|
| MTTOS:<br>Main page | • Check the correctness of the displayed data in JSP included pages components.<br>• Verify the login function working properly and get the error message properly.<br>• Check the entire movie title link working properly | OK |
| Movie Information Page | • Check the correctness of the displayed data.<br>• Check the button and links working properly.<br>• Verify the movie information save in session. | OK |
| Purchase Movie Ticket Page | • Check the correctness of the displayed data.<br>• Check the entire selection list, buttons, and links working properly.<br>• Test the correctness of the validation for all input data.<br>• Verify the page can get the error message and work properly by the message. | OK |
| Submit Payment Page | • Check the correctness of the displayed data.<br>• Check the entire buttons and links working properly.<br>• Check JavaScript function working properly.<br>• Test the correctness of the validation for all input data.<br>• Test the SSL function working as expected.<br>• Verify the page can get the error message and working properly by the message. | OK |
| Order Confirmation Page | • Check the correctness of the displayed data.<br>• Check the entire buttons and links working properly.<br>• Test the email response from JavaMail working as expected | OK |
| Member Register Page | • Check the entire buttons and links working properly.<br>• Test the correctness of the validation for all input data.<br>• Verify the page can get the error message and working properly by the message. | OK |

| Unit Test | Tests Performed | Results |
|---|---|---|
| Welcome New Member Page | • Check the correctness of the displayed data.<br>• Test the link working properly. | OK |
| Forget Password Page | • Check the correctness of the displayed data.<br>• Check all the buttons and links working properly. | OK |
| Member's Main Page | • Check the correctness of the displayed data in JSP included pages components.<br>• Check the buttons and links working properly.<br>• Verify the user save in session after login. | OK |
| Member Check Order Status Page | • Check the correctness of the displayed data.<br>• Test all the link working properly. | OK |
| Member Manage Profile Page | • Verify the page get the correct user information.<br>• Verify all links and buttons working as expected.<br>• Test the correctness of the validation for all input data<br>• Verify the page can get the error message and working properly by the message. | OK |
| Administrator Main Page | • Check the correctness of the displayed data in JSP included pages components.<br>• Check all the links and buttons working properly.<br>• Verify the user save in session after login. | OK |
| Add Movie Information Page | • Check the entire buttons, link and selection list working properly.<br>• Test the correctness of the validation for all input data.<br>• Verify the page can get the error message and working properly by the message. | OK |
| Delete/Edit Movie Page | • Check the correctness of the displayed data.<br>• Check button, links and check boxes working properly. | OK |
| Edit Movie Information page | • Verify the page get the correct movie information.<br>• Verify all links and buttons working as expected.<br>• Test the correctness of the validation for all input data.<br>• Verify the page can get the error message and working properly by the message. | OK |

| Unit Test | Tests Performed | Results |
|---|---|---|
| Add Account Page | • Check the entire buttons, link and selection list working properly.<br>• Test the correctness of the validation for all input data.<br>• Verify the page can get the error message and working properly by the message. | OK |
| Delete/Edit Account Page | • Check the correctness of the displayed data.<br>• Check button, links and check boxes working properly. | OK |
| Edit Account Information page | • Verify the page get the correct movie information.<br>• Verify all links and buttons working as expected.<br>• Test the correctness of the validation for all input data.<br>• Verify the page can get the error message and working properly by the message. | OK |
| Administrator Manage Profile Page | • Verify the page get the correct user information.<br>• Verify all links and buttons working as expected.<br>• Test the correctness of the validation for all input data<br>• Verify the page can get the error message and working properly by the message. | OK |

## 5.2 Subsystem Testing

Subsystem testing is the next step up in the testing process where all related units from a subsystem do a certain task. Thus, the subsystem test process is useful for detecting interface errors and specific functions. Table 24 show subsystem test results in detail.

Table 10. Subsystem Test Results

| Subsystem | Tests Performed | Results |
|---|---|---|
| Purchase Ticket Subsystem | • Make sure all the movie titles are shown on the list.<br>• Make sure all the movie information are displayed properly.<br>• Verify the subsystem check the guest select movie information before forwarding to submit payment page.<br>• Verify the subsystem check the credit card information before insert information to database and charge money from it. | OK |
| Join Member Subsystem | • Make sure the subsystem checks the guest information before create a new account for him/her.<br>• Check if the subsystem can detect the error of creating of the user that exists in the subsystem | OK |
| Authorize Subsystem | • Test if it can get the error message.<br>• Make sure the result of authorizing user is correct.<br>• Verify the login user information is store in session properly.<br>• Verify the login page redirect to the correct browsing or editing page after the user logins in. | OK |
| Accounts Management Subsystem | • Make sure all the existing users are list in the user list.<br>• Check if the subsystem can detect the error of creating of the user that exists in the subsystem.<br>• Check if the user can update his/her personal account properly.<br>• Verify the created user information is the same as the information provided.<br>• Verify the subsystem can delete a user account properly.<br>• Make sure the password query function can working properly. | OK |
| Editing Account Subsystem | • Make sure the subsystem checks the user privilege before forwarding to edit page.<br>• Verify the subsystem check the user privilege before update the page information.<br>• Verify if the subsystem shows the page properties is the users are the owner or the administrator. | OK |

| Subsystem | Tests Performed | Results |
|---|---|---|
| Movie Management Subsystem | • Make sure all the existing movies are list in the movie list.<br>• Check if the subsystem can detect the error of creating of the movie that exists in the subsystem.<br>• Check if the administrator can update movie information properly.<br>• Verify the created movie information is the same as the information provided.<br>• Verify the subsystem can delete a movie properly. | OK |
| Editing Movie Subsystem | • Make sure the subsystem checks the movie information before forwarding to edit page.<br>• Verify the subsystem check the movie information before update the page information.<br>• Verify if the subsystem shows the movie information page properties. | OK |
| Edit Top10 Movie Subsystem | • Verify the subsystem check the top10 movie information before update the page information.<br>• Make sure all the top10 movie titles are list in the top10 movie list. | OK |
| Browsing Subsystem | • Check if the subsystem checks for user privilege before showing pages.<br>• Verify the page is showing properly after the user click on the page link. | OK |

## 5.3 System Test Plan

System test plan is a test process that uses real data, which the system is intended to manipulate, to test the system. First of all the subsystem will be integrated into one system. Then test the system by using a variety of data to see the overall results.

The steps for the system test plan are showed in the following table:

Table 11. System Test Results

| System Test | Results |
|---|---|
| 1. Install the MTTOS into server. | OK |
| 2. Start up all services such as JSP engine, PostgreSQL database engine. | OK |
| 3. Running testing by using real data on all forms and reports. | OK |

# CHAPTER SIX

# MAINTENANCE MANUAL

The Maintenance Manual provides maintenance personnel with the information necessary to maintain the system effectively. The manual provides the definition of the software support environment, the roles and responsibilities of maintenance personnel, and the regular activities essential to the support and maintenance of program modules, job streams, and database structures. In MTTOS, there are 3 major issues: Software Installation, Variable Installation, and MTTOS Installation.

## 6.1 Software Installation

In MTTOS, it requires RedHat, PostgreSQL, JSDK, Ant, TOMCAT, and JDBC to run the programs. Following will detail the installation of those five software systems.

### 6.1.1 RedHat Installation

RedHat is the most popular distribution for Linux. Following are the steps to install RedHat 9.0 onto your machine.

1. Download a latest version of the RedHat operating systems from http://ftp.redhat.com/pub/redhat/linux/9/en/iso/i386/ and burn the iso files into CDs.

2. RedHat Linux 9.0 is distributed on 3 CDs. Install the operating system by inserting CD 1 into the CD-ROM and make sure the bios is set to boot from CD.

3. The machine will be startup via CD-ROM and start to install RedHat.

4. Follow the install wizard and sets up the required information such as network setting and the hardware environment.

5. After all the necessary files are copied into the computer and install it, the machine will restart and RedHat is installed.

### 6.1.2 Install JAVA 2 Platform, Standard Edition (J2SE)

Following are the steps to install J2SE SDK onto your machine.

1. Create a directory under /usr called /java. Go to http://java.sun.com/j2se/1.4.1/download.html to download SDK Linux to this directory. then execute the following commands:

```
chmod +x j2sdk-1_4_2_01-linux-i586-rpm.bin
./j2sdk-1_4_2_01-linux-i586-rpm.bin
rpm -ivh j2sdk-1_4_2_01-linux-i586.rpm
```

2. Set the environment variables in the file /etc/profile.d/myenv.sh, and adding the following lines to /etc/profile.d/myenv.sh:

```
JAVA_HOME=/usr/java/j2sdk1.4.2_01
PATH=${PATH}:${JAVA_HOME}/bin
Export JAVA_HOME
Export PATH
```

### 6.1.3 Install Ant

Following are the steps to install Ant under the directory /usr/java onto your machine.

1. Go to http://apache.oregonstate.edu/ant/binaries to obtain a more recent url, and then type tar -zxvf apache-ant-1.6.0-bin.tar.gz.
2. Set the environment variable and adding the line to /etc/profile.d/myenv.sh.

```
ANT_HOME=/usr/java/ apache-ant-1.6.0
PATH=${PATH}:${ANT_HOME}/bin
Export ANT_HOME
```

### 6.1.4 Install Tomcat

Following are the steps to install Tomcat under the directory /usr/java onto your machine

1. http://mirrors.midco.net/pub/apache.org/jakata/tomcat-4/tomcat-4.1.29.tar.gz -O, and then type tar -zxvf Jakarta-tomcat-4.1.29.tar.gz.

2. Set the environment variable and adding the line to /etc/profile.d/myenv.sh.

```
CATALINA_HOME=/usr/java/ jakata-tomcat-4.1.29
PATH=${PATH}:${CATALINA_HOME}/bin
Export CATALINA_HOME
```

3. The root user will run an instance of tomcat. Modify the file /usr/java/jarkata-tomcat-4.1.29/conf/ server.xml by add the following after ROOT in the file to run ticket web application:

```
<Context path="/ticket"
docBase="/home/ticket/web"
debug="0"
reloadable="true"
swallowOutput="true"
useNaming="true"/>
```

4. Restart Tomcat and type http://localhost:8080/ticket to test that your system works OK.

### 6.1.5 PostgreSQL Installation

The database system in MTTOS is PostgreSQL. To install PostgreSQL, follow the following steps:

1. PostgreSQL is preconfigured under RedHat 9.0 to run as user postgres. This user's home directory is /var/lib/pgsql. If PostgreSQL is not installed in the

RedHat 9.0, then you can go to

http://www.postgresql.org to download it.

2. Initialize a data directory by running he follow command as postgres user:

   su postgres

   initdb -D /var/lib/pgsql/data

3. In the user's environment setup file /etc/profile.d/*.sh, add the following line:

   PGDATA=/var/lib/pgsql/data

   export PGDATA

4. Uncomment the line (tcpip_socket = ture) in the file /var/lib/pgsql/data/postgresql.conf.

5. As root, type /sbin/chkconfig --level 3 postgresql on to run the server at boot time and type /sbin/service postgresql start to start server without rebooting.

6. Create database users. As postgres, run the following command:

   createuser -P ticket

7. Create a database called ticket that is to be owned by ticket. Do the following as postgres:

   createdb -O ticket ticket

## 6.2 Variable Modifications

In MTTOS, we have to change some environment variables in the Linux system and server.xml in Tomcat server configuration directory.

### 6.2.1 System Variables

1. Open the file "server.xml" in the directory "/usr/java/jakarta-tomcat-4.1.29/conf" via "vi" or any other editor.
2. Scroll down until you see the context area we added in at chapter 5.4.1.
3. The variable "path" in Context indicates the context path of the web application. The default value would be "/ticket."
4. The variable "docBase" in Context is the files directory for the web application. The default value would be "/home/ticket/web."
5. The variable "variable" in Logger is the absolute or relative pathname of a directory in which log files created by this logger will be placed. The default value would be "/home/ticket."
6. Now, let's look down at the parameter setting for MTTOS.

7. The parameter "contextPath" indicate the context path for MTTOS which would be the same as the value of path.
8. The parameter "username" is the user name who can access the database system. Usually, this value would be the administrator of MTTOS.
9. The parameter "password" is the password corresponding to the user name at 8. If there is no special setting in database system, leave the value to be empty.

## 6.3 Movie Theater Ticket Order System (MTTOS) Installation/Migration

1. All the JSP programs and HTML programs are stored in
   /home/ticket/web
2. All the *.java all stored in
   /home/ticket/src/ticket
3. All the classes are stored in
   /home/ticket /web/WEB-INF/classes
4. All the pictures all stored in
   /home/ticket/web/pics
5. Place the web.xml for MTTOS in
   /home/ticket/web/WEB-INF

## 6.4 Backup and Restore

Protecting system information is one of the system administrator's most import tasks. Backups allow the administrator to restore a file system to the condition it was in at the time of the last backup. Backups must be done carefully and on a strict schedule. The backup system and backup media must also be tested regularly to verify that they are working correctly. There are two steps to back up MTTOS. One is to backup the system files. The other step is to backup the database which is used by MTTOS.

### 6.4.1 System Backup

All the MTTOS system files are stored in the directory "/home/ticket" and the subdirectory of its. Thus, in order to backup the system files, we can compress this directory by using the compress program "tar" to backup the system files:

```
tar -cf MTTOS.tar /home/ticket
```

### 6.4.2 Database Backup

To backup the database system, we use pg_dump command to backup all the database used by MTTOS. The following command is used to backup the database:

1. Create a subdirectory named backup under the directory /usr/local/pgsql/

2. Using command:

pg_dump ticket(database name) | gzip > MTTOS.gz

(backupfilename)

After executing the backup command above, the file MTTOS.gz would be the backup file of the database.

### 6.4.3 System Restore

To restore the system file, simply extract the backup file by using the following command:

tar -xzvf MTTOS.tar /

### 6.4.4 Database Restore

To restore the database needed for the MTTOS, go to the directory /usr/local/pgsql/backup, and execute the following commands:

createdb ticket

cat MTTOS.gz (backupfilename) | gunzip | psql ticket

(database name)

# CHAPTER SEVEN

# CONCLUSION AND FUTURE DIRECTIONS

## 7.1 Conclusion

All the functions of MTTOS are descripted as follows: 1) Guests can check any movie schedule in website; information regarding the movies are easily obtained. 2) The consumer can order tickets online; they don't need to go to Movie Theater to save time. 3) The movie theater needs fewer workers to take care of orders. 4) Movie theater club members get information form the club through the website or e-mail. 5) Members, in turn, can log on and check the details of purchased tickets if they need to.

The objective of the MTTOS project is to modernize a movie theater's administration, with a focus on customer service, and to link this effort with a boost for the movie theater's business. This is to be achieved by developing a new delivery channel for customer services and Movie Theater's business: the integrated electronic single-window service.

Taking advantage of this system, for example, a movie theatre can significantly save cost of sales, offer a non-stop service of 24-hours per day, 7-days a week, extend the dynamical advertising channels, and process the

data regarding sales of the movies in real time by which the management can be facilitated intelligently: all of these features can undoubtedly boost the competitive ability of a movie theater for grabbing more market share.

## 7.2 Future Direction

To build more friendly graphical interfaces, improve the member club such as provide more functions for member and improve the performance of the system is another goal for a further project.

# APPENDIX A

# XML FILES

# XML FILES

The build.xml

```
<!DOCTYPE project [
 <!ENTITY path_names SYSTEM "/home/global/path_names.inc">
 <!ENTITY manager_targets SYSTEM "/home/global/manager_targets.inc">
]>
<project basedir="." default="build">
  &path_names;
  <property name="context.path" value="/ticket" />
  <property name="context.file" value="context.xml" />
  &manager_targets;
  <target name="build">
    <!-- compile -->
    <javac srcdir="src" destdir="web/WEB-INF/classes">
      <classpath>
        <pathelement location="${servlet.jar}" />
        <pathelement location="${catalina.home}/common/lib/jdbc2_0-stdext.jar"/>
      </classpath>
    </javac>
  </target>
  <target name="clean">
    <delete dir="web/WEB-INF/classes" />
    <mkdir dir="web/WEB-INF/classes" />
  </target>
  <target name="rmlogs">
    <delete>
      <fileset dir="."">
        <include name="*.log" />
      </fileset>
    </delete>
  </target>
  <target name="restart">
    <antcall target="stop" />
    <antcall target="start" />
  </target>

  <target name="stop">
    <antcall target="remove" />
    <antcall target="rmlogs" />
  </target>
  <target name="start">
    <antcall target="install" />
  </target>
</project>
```

The web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
   PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
   "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
  <!-- filters -->
   <filter>
    <filter-name>accessControlFilter</filter-name>
    <filter-class>ticket.AccessControlFilter</filter-class>
  </filter>
  <filter>
    <filter-name>hidingFilter</filter-name>
    <filter-class>ticket.HidingFilter</filter-class>
  </filter>
```

```
 <!-- filter mappings -->
<filter-mapping>
  <filter-name>accessControlFilter</filter-name>
  <url-pattern>/admin/*</url-pattern>
</filter-mapping>
 <filter-mapping>
  <filter-name>hidingFilter</filter-name>
  <url-pattern>/hidden/*</url-pattern>
</filter-mapping>

<!-- servlets -->
 <servlet>
  <servlet-name>account</servlet-name>
  <servlet-class>ticket.AccountServlet</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet>
  <servlet-name>addmovie</servlet-name>
  <servlet-class>ticket.AddMovieServlet</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet>
  <servlet-name>checkseat</servlet-name>
  <servlet-class>ticket.CheckSeatServlet</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet>
  <servlet-name>delete</servlet-name>
  <servlet-class>ticket.DeleteServlet</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet>
  <servlet-name>display</servlet-name>
  <servlet-class>ticket.DisplayServlet</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet>
  <servlet-name>edit</servlet-name>
  <servlet-class>ticket.EditServlet</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet>
  <servlet-name>login</servlet-name>
  <servlet-class>ticket.LoginServlet</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet>
  <servlet-name>logout</servlet-name>
  <servlet-class>ticket.LogoutServlet</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet>
  <servlet-name>mail</servlet-name>
  <servlet-class>ticket.MailServlet</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet>
  <servlet-name>movieupdate</servlet-name>
  <servlet-class>ticket.MovieupdateServlet</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet>
  <servlet-name>orderticket</servlet-name>
  <servlet-class>ticket.BuyTicketServlet</servlet-class>
```

```
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet>
    <servlet-name>orderinfo</servlet-name>
    <servlet-class>ticket.OrderInfoServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet>
    <servlet-name>schedule</servlet-name>
    <servlet-class>ticket.ScheduleServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet>
    <servlet-name>search</servlet-name>
    <servlet-class>ticket.SearchServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet>
    <servlet-name>submitpayment</servlet-name>
    <servlet-class>ticket.SubmitPaymentServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet>
    <servlet-name>update</servlet-name>
    <servlet-class>ticket.UpdateServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet>
    <servlet-name>updatetop</servlet-name>
    <servlet-class>ticket.UpdateTopServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <!-- servlet mapping -->
  <servlet-mapping>
    <servlet-name>account</servlet-name>
    <url-pattern>/admin/account</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>account</servlet-name>
    <url-pattern>/account</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>addmovie</servlet-name>
    <url-pattern>/admin/addmovie</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>checkseat</servlet-name>
    <url-pattern>/checkseat</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>delete</servlet-name>
    <url-pattern>/admin/delete</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>display</servlet-name>
    <url-pattern>/display</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>display</servlet-name>
    <url-pattern>/admin/display</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>display</servlet-name>
    <url-pattern>/member/display</url-pattern>
```

```
</servlet-mapping>
<servlet-mapping>
  <servlet-name>edit</servlet-name>
  <url-pattern>/edit</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>edit</servlet-name>
  <url-pattern>/admin/edit</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>edit</servlet-name>
  <url-pattern>/member/edit</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>login</servlet-name>
  <url-pattern>/login</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>logout</servlet-name>
  <url-pattern>/admin/logout</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>logout</servlet-name>
  <url-pattern>/member/logout</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>mail</servlet-name>
  <url-pattern>/mail</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>movieupdate</servlet-name>
  <url-pattern>/admin/movieupdate</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>orderticket</servlet-name>
  <url-pattern>/orderticket</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>orderinfo</servlet-name>
  <url-pattern>/member/orderinfo</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>schedule</servlet-name>
  <url-pattern>/schedule</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>schedule</servlet-name>
  <url-pattern>/admin/schedule</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>schedule</servlet-name>
  <url-pattern>/member/schedule</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>search</servlet-name>
  <url-pattern>/admin/search</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>submitpayment</servlet-name>
  <url-pattern>/submitpayment</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>update</servlet-name>
  <url-pattern>/admin/update</url-pattern>
</servlet-mapping>
```

```
  <servlet-mapping>
    <servlet-name>update</servlet-name>
    <url-pattern>/member/update</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>updatetop</servlet-name>
    <url-pattern>/admin/updatetop</url-pattern>
  </servlet-mapping>
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
  <!-- tag library descriptor -->
  <taglib>
    <taglib-uri>turner</taglib-uri>
    <taglib-location>tlds/turner.tld</taglib-location>
  </taglib>
  <resource-ref>
      <res-ref-name>database</res-ref-name>
      <res-type>javax.sql.DataSource</res-type>
      <res-auth>Container</res-auth>
   </resource-ref>
</web-app>
```

# APPENDIX B

# JAVA SOURCE CODE

# JAVA SOURCE CODE

**The AccessControlFilter.java**

```
package ticket;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.IOException;

// Requires users to login successfully in order to access protected resources.
public class AccessControlFilter implements Filter
{
  public void init(FilterConfig filterConfig) throws ServletException { }
  public void destroy() { }

  // User is logged in if a user object is stored in session.
  public void doFilter( ServletRequest req, ServletResponse resp, FilterChain chain) throws IOException, ServletException
  {
    HttpServletRequest request = (HttpServletRequest) req;
    HttpServletResponse response = (HttpServletResponse) resp;
    HttpSession session = request.getSession();
    User user = (User) session.getAttribute("user");
    if (user == null) {
      String requestedResource = request.getRequestURL().toString();
      session.setAttribute("requestedResource", requestedResource);
      response.sendRedirect(request.getContextPath() + Constants.loginPagePath);
      return;
    }
    chain.doFilter(request, response);
  }
```

**The AccountServlet.java**

```
package ticket;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;
import java.io.*;

public class AccountServlet extends HttpServlet
{
  RequestDispatcher adminaccountPage;
  RequestDispatcher accountPage;
  RequestDispatcher homePage;
  RequestDispatcher thanksPage;

  public void init() throws ServletException
  {
    ServletContext context = getServletContext();
    accountPage = context.getRequestDispatcher(Constants.accountPagePath);
    if (accountPage == null) {
      throw new ServletException(Constants.accountPagePath + " not found");
    }
    adminaccountPage = context.getRequestDispatcher(Constants.adminaccountPagePath);
    if (adminaccountPage == null) {
      throw new ServletException(Constants.adminaccountPagePath + " not found");
    }
    homePage = context.getRequestDispatcher(Constants.homePagePath);
    if (homePage == null) {
      throw new ServletException(Constants.homePagePath + " not found");
    }

    thanksPage = context.getRequestDispatcher(Constants.thanksPagePath);
    if (thanksPage == null) {
```

```
    throw new ServletException(Constants.thanksPagePath + " not found");
  }
}

public void doPost (HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
{
   doGet(request, response);
}

public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
{
  String userid = request.getParameter("id");
  String passwd = request.getParameter("passwd");
  String repasswd = request.getParameter("repasswd");
  String fname = request.getParameter("fname");
  String lname = request.getParameter("lname");
  String addr = request.getParameter("addr");
  String city = request.getParameter("city");
  String state = request.getParameter("state");
  String zip = request.getParameter("zip");
  String email = request.getParameter("email");
  String auth = request.getParameter("auth");
  String host = request.getParameter("host");

  if(host.equals("member"))
  {
  if(userid.equals(""))
  { request.setAttribute("error", "User ID is required.");
    accountPage.forward(request, response);
    return;
  }
  else if(passwd.equals(""))
  { request.setAttribute("error", "Password is required.");
    accountPage.forward(request, response);
    return;
  }
  else if(repasswd.equals(""))
  { request.setAttribute("error", "Re-type Password is required.");
    accountPage.forward(request, response);
    return;
  }
  else if(!passwd.equals(repasswd))
  { request.setAttribute("error", "Password and Re-type Password are not matched.");
    accountPage.forward(request, response);
    return;
  }
  else if(email.equals(""))
  { request.setAttribute("error", "E-Mail is required.");
    accountPage.forward(request, response);
    return;
  }
}

if(host.equals("admin"))
{
 if(userid.equals(""))
  { request.setAttribute("error", "User ID is required.");
    adminaccountPage.forward(request, response);
   return;
  }
  else if(passwd.equals(""))
  { request.setAttribute("error", "Password is required.");
    adminaccountPage.forward(request, response);
```

```
  return;
 }
 else if(repasswd.equals(""))
 { request.setAttribute("error", "Re-type Password is required.");
  adminaccountPage.forward(request, response);
  return;
 }
 else if(!passwd.equals(repasswd))
 { request.setAttribute("error", "Password and Re-type Password are not matched.");
  adminaccountPage.forward(request, response);
  return;
 }
 else if (email.equals(""))
 { request.setAttribute("error", "E-Mail is required.");
  adminaccountPage.forward(request, response);
  return;
 }
}

 Connection con= null;
 String sql= null;
 Statement stmt= null;
 ResultSet rs= null;

 try
 {
        con = DriverManager.getConnection("jdbc:postgresql://127.0.0.1/ticket?user=ticket&password=762316");
        sql = "select * from member where mem_id ="+ userid +";";
        stmt = con.createStatement();
        rs = stmt.executeQuery(sql);

        if(rs.next())
        {
          if(auth =="1" || auth.equals("1"))
          {
              request.setAttribute("error", "User Name already exists, please try another one.");
              accountPage.forward(request, response);
              return;
          }
          if(auth =="2" || auth.equals("2"))
          {
              request.setAttribute("error", "User Name already exists, please try another one.");
              adminaccountPage.forward(request, response);
              return;
          }
        }
 }catch (SQLException e)
 { String error = " error !";
        throw new ServletException(error);
    }finally{
        try
        { stmt.close();
   rs.close();
   con.close();
        }catch(SQLException ingored){}
   }

try
{
 Class.forName("org.postgresql.Driver").newInstance();
 con = DriverManager.getConnection("jdbc:postgresql://127.0.0.1/ticket?user=ticket&password=762316");
 sql="insert into member values (?,?,?,?,?,?,?,?,?,?);";
 PreparedStatement pstmt = con.prepareStatement(sql);
 pstmt.setString(1, userid);
 pstmt.setString(2, fname);
```

```
      pstmt.setString(3, lname);
      pstmt.setString(4, addr);
      pstmt.setString(5, city);
      pstmt.setString(6, state);
      pstmt.setString(7, zip);
      pstmt.setString(8, email);
      pstmt.setString(9, passwd);
      pstmt.setString(10, auth);
      pstmt.executeUpdate();
      pstmt.close();
      con.close();
    }catch (Exception e)
    { String error = "create account error !";
        throw new IOException(error);
    }finally{
        try
        { pstmt.close();
            con.close();
        }catch(SQLException ingored){}
    }
    if(host.equals("admin"))
    { response.sendRedirect("index.jsp");
    }else
    { response.sendRedirect(request.getContextPath()+Constants.thanksPagePath);
    }
  }
}
```

*The AddMovieServlet.java*

```
package ticket;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;
import java.io.*;

public class AddMovieServlet extends HttpServlet
{
  RequestDispatcher addmoviePage;
  RequestDispatcher addmoviecomfirmPage;

  public void init() throws ServletException
  {
    ServletContext context = getServletContext();
    addmoviePage = context.getRequestDispatcher(Constants.addmoviePagePath);
    if (addmoviePage == null) {
      throw new ServletException(Constants.addmoviePagePath + " not found");
    }
    addmoviecomfirmPage = context.getRequestDispatcher(Constants.addmoviecomfirmPagePath);
    if (addmoviecomfirmPage == null) {
      throw new ServletException(Constants.addmoviecomfirmPagePath + " not found");
    }
  }

  public void doPost (HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException
  {
    HttpSession session = request.getSession();
    String id = request.getParameter("movie_nu");
    String title = request.getParameter("title");
    String status = request.getParameter("status");
    String room_nu = request.getParameter("room_nu");
    String genre = request.getParameter("genre");
    String release = request.getParameter("release");
    String end_date = request.getParameter("end_date");
```

```
String starring = request.getParameter("starring");
String story = request.getParameter("story");
String key = request.getParameter("key");

if((id.equals("")) || (id == null))
{ request.setAttribute("error", "Movie Number is required.");
  addmoviePage.forward(request, response);
  return;
}

Connection con= null;
String sql= null;
Statement stmt= null;
ResultSet rs= null;
try
{
        con = DriverManager.getConnection("jdbc:postgresql://127.0.0.1/ticket?user=ticket&password=762316");
        sql = "select * from movie where movie_nu = '"+ id +"';";
        stmt = con.createStatement();
        rs = stmt.executeQuery(sql);

        if(rs.next())
        { request.setAttribute("error", "Movie Number already exists, please try another.");
          addmoviePage.forward(request, response);
          return;
        }
 }catch (SQLException e)
 { String error = " error !";
        throw new ServletException(error);
    }finally{
        try
        { stmt.close();
   rs.close();
   con.close();
        }catch(SQLException ingored){}
    }

PreparedStatement pstmt= null;
try
{  Class.forName("org.postgresql.Driver").newInstance();
  con = DriverManager.getConnection("jdbc:postgresql://127.0.0.1/ticket?user=ticket&password=762316");
  sql=("insert into movie values (?,?,?,?,?,?,?,?,?);");
  pstmt = con.prepareStatement(sql);
  pstmt.setString(1, id);
  pstmt.setString(2, title);
  pstmt.setString(3, status);
  pstmt.setString(4, room_nu);
  pstmt.setString(5, genre);
  pstmt.setString(6, release);
  pstmt.setString(7, end_date);
  pstmt.setString(8, starring);
  pstmt.setString(9, story);
  pstmt.executeUpdate();
 }catch (Exception e)
 { String error = "create account error !";
     throw new IOException(error);
 }finally{
     try
     { pstmt.close();
         con.close();
     }catch(SQLException ingored){}
 }

String[] run=request.getParameterValues("runtime") ;
if(run != null){
```

```
    for (int i=0;i <run.length; i++)
    {
     try
     { con = DriverManager.getConnection("jdbc:postgresql://127.0.0.1/ticket?user=ticket&password=762316");
       sql=("insert into movie_time values (?,?);");
       pstmt = con.prepareStatement(sql);
       pstmt.setString(1, id);
       pstmt.setString(2, run[i]);
       pstmt.executeUpdate();
       pstmt.close();
     }
     catch (SQLException e)
     { String error = " error !";
          throw new ServletException(error);
     }
     finally{
            try
            { stmt.close();
       rs.close();
       con.close();
            }catch(SQLException ingored){}
     }
    }
  }
  session.setAttribute("checkmovie", id);
  response.sendRedirect("display");
 }
}
```

The Addseat.java

```
package ticket;
import java.util.Hashtable;
import javax.naming.*;
import java.sql.*;
import javax.sql.DataSource;
import java.io.IOException;

public  class Addseat
{
  private String id;
  private String moviedate;
  private String time;
  private int seat;

  public Addseat(String id, String moviedate, String time, int seat)
  {  this.id = id;
     this.moviedate = moviedate;
     this.time = time;
     this.seat = seat;
  }

  public String getId() { return id; }
  public String getMoviedate() { return moviedate; }
  public String getTime() { return time; }
  public int getSeat() { return seat; }

  public Addseat add(String id, String moviedate, String time, int quantity) throws Exception
  {  Addseat addseat= null;
     Connection con= null;
     PreparedStatement pstmt= null;
     try
     { Class.forName("org.postgresql.Driver").newInstance();
       con = DriverManager.getConnection("jdbc:postgresql://127.0.0.1/ticket?user=ticket&password=762316");
       String sql="insert into seat values (?,?,?,?);";
```

```
      pstmt = con.prepareStatement(sql);
      pstmt.setString(1,id);
      pstmt.setString(2,moviedate);
      pstmt.setString(3,time);
      pstmt.setInt(4,quantity);
      pstmt.executeUpdate();
      pstmt.close();
     }
     catch (SQLException e)
     { String error = " error checking !";
            throw new IOException(error);
        }
        return addseat;
  }
}
```

The Booking.java

```
package ticket;
import java.util.*;
import java.io.*;

public class Booking
{
 private int numChar;
 private String account;
 String table=
    "0123456789"+"ABCDEFGHIJKLMNOPQRSTUVWXYZ" +
    "0123456789" + "abcdefghijklmnopqrstuvwxyz" +
    "123456";
 int n = table.length()-1;

 public Booking()
 { numChar=8;
  account="ABCabc";
 }

 public void setNumChar(int nc)
 {
  if (nc<=16)
    numChar=nc;
  else
    numChar=16;
 }

 public int getNumChar()
 {return numChar;
 }

 public void createAccount()
 { char ca[] = new char[numChar];
  int k = 0;
  for (int i=0; i<numChar; i++)
  { k=(int)(Math.random()*n);
    ca[i] = table.charAt(k);
  }
  account = new String(ca);
 }
 public String getAccount()
 { return account;
 }
}
```

The BuyTicketServlet.java

```
package ticket;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.sql.*;
import java.util.*;

public class BuyTicketServlet extends HttpServlet
{
  RequestDispatcher buyPage;

  public void init() throws ServletException
  { ServletContext context = getServletContext();
    buyPage = context.getRequestDispatcher(Constants.buyPagePath);
    if (buyPage == null) {
      throw new ServletException(Constants.buyPagePath + " not found");
    }
  }

  public void doPost (HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException
  { doGet(request, response);
  }

  public void doGet(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException
  {
     Connection con = null;
     Statement stmt = null;
     ResultSet rs = null;
     HttpSession session = request.getSession();
     String adult="",child="",senior="",mon="",day="" ;
     int om,od,endmonth,enddate,a1,c1,s1,quantity;
     int year, month, date;
     int Apr=4, Jun=6, Sep=9, Nov=11, Feb=2;
     int  date30 = 30, date28=28, ticket_quantity=0 ;
     double price;
     String id = ((String) session.getAttribute("movie_nu")).trim();
     String title = (String) session.getAttribute("title");
     String room_nu = ((String) session.getAttribute("room_nu")).trim();
     String end_date = (String) session.getAttribute("end_date");
     String time = (request.getParameter("time")).trim();
     mon = request.getParameter("month");
     day = request.getParameter("day");
     adult = request.getParameter("adult");
     child = request.getParameter("child");
     senior = request.getParameter("senior");
     om=Integer.parseInt((String)request.getParameter("month"));
     od=Integer.parseInt((String)request.getParameter("day"));
     a1=Integer.parseInt((String)request.getParameter("adult"));
     c1=Integer.parseInt((String)request.getParameter("child"));
     s1=Integer.parseInt((String)request.getParameter("senior"));
     endmonth=Integer.parseInt(end_date.substring(0,2));
     enddate=Integer.parseInt(end_date.substring(3,5));
     Calendar now= Calendar.getInstance();
     year = now.get(Calendar.YEAR);
     month = ((now.get(Calendar.MONTH))+1);
     date = now.get(Calendar.DATE);
     quantity = a1+c1+s1;
     String moviedate = mon+"/"+day;

    if(om==Apr || om==Jun || om==Sep || om==Nov )
    { if( od > date30 )
      { request.setAttribute("error", "The date doesn't exist in calendar.");
```

```
    buyPage.forward(request, response);
    return;
   }
}else if(om==Feb )
{  if( od > date28 )
   { request.setAttribute("error", "The date doesn't exist in calendar.");
    buyPage.forward(request, response);
    return;
   }
}

if(om<month )
{  request.setAttribute("error", "The date is passed.");
  buyPage.forward(request, response);
  return;
}else if(om==month)
{ if(od<date)
  { request.setAttribute("error", "The date is passed.");
    buyPage.forward(request, response);
    return;
  }else if(od==date)
  { request.setAttribute("error", "Please make your movie date one day ahead today.");
    buyPage.forward(request, response);
    return;
  }
}else if(om>endmonth )
{ request.setAttribute("error", "Please chose the date before "+end_date);
 buyPage.forward(request, response);
 return;
}else if(om==endmonth )
{  if(od > enddate)
   { request.setAttribute("error", "Please chose the date before"+end_date);
    buyPage.forward(request, response);
    return;
   }
}

if (quantity==ticket_quantity )
{ request.setAttribute("error", "Please select the ticket quantity.");
 buyPage.forward(request, response);
 return;
}

Check check=null;
Checkroom checkroom=null;
try
{    check =Check.find(id,moviedate,time);
}
catch (Exception e)
{ String error = "id= " + id + " moviedate= " + moviedate + " time= " + time;
 throw new ServletException(error);
}

int seat = check.getSeat();
int capacity=0;
int total= quantity+seat;

if(seat !=ticket_quantity)
{
 try
  {  checkroom = Checkroom.roomcapacity(room_nu);
  }catch (Exception e)
  { String error = room_nu+"roomcapacity";
   throw new ServletException(error);
  }
```

```
      capacity = checkroom.getCapacity();
       if(capacity < total)
       { request.setAttribute("error", "there are only "+(capacity-seat)+" seats left!");
        buyPage.forward(request, response);
        return;
       }
     }
    response.sendRedirect("https://tickets.ias.csusb.edu:8443/ticket/payment.jsp");
  }
}
```

The Check.java

```
package ticket;
import java.util.Hashtable;
import javax.naming.*;
import java.sql.*;
import javax.sql.DataSource;
import java.io.IOException;

public class Check
{
  private String id;
  private String moviedate;
  private String time;
  private int seat;

  public Check(String id, String moviedate, String time, int seat)
  {  this.id = id;
    this.moviedate = moviedate;
    this.time = time;
    this.seat = seat;
  }

  public String getId() { return id; }
  public String getMoviedate() { return moviedate; }
  public String getTime() { return time; }
  public int getSeat() { return seat; }

  public Check find(String id, String moviedate, String time) throws Exception
  {  Check check= null;
    Connection Conn= null;
    Statement stmt= null;
    try
    { Class.forName("org.postgresql.Driver").newInstance();
      Conn = DriverManager.getConnection("jdbc:postgresql://127.0.0.1/ticket?user=ticket&password=762316");
      stmt= Conn.createStatement();
      String sql="select * from seat where (movie_nu = '" + id + "' and moviedate= '"+moviedate+"' and time=
    "'+time+"');";
      ResultSet rs = stmt.executeQuery(sql);
      String i="",m="",t="";
      int s=0;

        while(rs.next())
        { i= rs.getString("movie_nu");
         m= rs.getString("moviedate");
         t= rs.getString("time");
     s= rs.getInt("seat");
        }
        check = new Check(i,m,t,s);
    stmt.close();
    rs.close();

  }
```

```
    catch (SQLException e)
    { String error = " error checking !";
            throw new IOException(error);
       }
    return check;
  }
}
```

The Checkroom.java

```
package ticket;
import java.util.Hashtable;
import javax.naming.*;
import java.sql.*;
import javax.sql.DataSource;
import java.io.IOException;

public class Checkroom
{
  private String room_nu;
  private int capacity;

  public Checkroom(String room_nu, int capacity)
  { this.room_nu = room_nu;
    this.capacity = capacity;
  }

  public String getRoom_nu() { return room_nu; }
  public int getCapacity() { return capacity; }

  public Checkroom roomcapacity(String room_nu) throws Exception
  { Checkroom checkroom= null;
    Connection Conn= null;
    Statement stmt= null;
    try
    { Class.forName("org.postgresql.Driver").newInstance();
        Conn = DriverManager.getConnection("jdbc:postgresql://127.0.0.1/ticket?user=ticket&password=762316");
        stmt= Conn.createStatement();
        String sql="select * from room where room_nu= '"+room_nu+"';";
        ResultSet rs = stmt.executeQuery(sql);
        String nu="";
        int cap=0;

            while(rs.next())
            { nu= rs.getString("room_nu");
              cap= rs.getInt("capacity");
            }
            checkroom = new Checkroom(nu,cap);
      stmt.close();
      rs.close();
    }catch (SQLException e)
    { String error = " error checking !";
            throw new IOException(error);
       }
    return checkroom;
  }
}
```

The Constants.java

```
package ticket;
import java.util.Vector;

public class Constants
```

```
{
  public static final String ticketPagePath = "/hidden/ticket.jsp";
  public static final String accountPagePath = "/signup.jsp";
  public static final String buyPagePath = "/buy.jsp";
  public static final String checkseatPagePath = "/payment.jsp";
  public static final String confirmPagePath = "/payment_confirm.jsp";
  public static final String comingPagePath = "/coming.jsp";
  public static final String displayPagePath = "/schedule.jsp";
  public static final String homePagePath = "/index.jsp";
  public static final String loginPagePath = "/login.jsp";
  public static final String orderconfirmPagePath = "/payment_confirm.jsp";
  public static final String schedulePagePath = "/schedule.jsp";
  public static final String thanksPagePath = "/thanks.jsp";
  public static final String addmoviePagePath = "/admin/add_movie.jsp";
  public static final String addmoviecomfirmPagePath = "/admin/addmovie_comfirm.jsp";
  public static final String adminPagePath = "/admin/index.jsp";
  public static final String adminaccountPagePath = "/admin/admin_signup.jsp";
  public static final String admindeletePagePath = "/admin/search";
  public static final String adminupdatePagePath = "/admin/member_update.jsp";
  public static final String adminschedulePagePath = "/admin/schedule.jsp";
  public static final String deletePagePath = "/admin/admin_menu.jsp";
  public static final String movieupdatePagePath = "/admin/movie_update.jsp";
  public static final String searchPagePath = "/admin/admin_menu.jsp";
  public static final String memberPagePath = "/member/index.jsp";
  public static final String memberupdatePagePath = "/member/member_update.jsp";
  public static final String memberschedulePagePath = "/member/schedule.jsp";
  public static final String jndiContainerContext = "java:comp/env";
  public static final String jndiDatabaseName = "database";;
  public static final String SMTP_KEY = " silicon.csci.csusb.edu";
  public static final String FROM_KEY = "cchiu@csci.csusb.edu";
  public static final String SUBJ_KEY = " Modify Your Password";
}
```

The DisplayServlet.java

```
package ticket;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.sql.*;
import java.util.*;

public class DisplayServlet extends HttpServlet
{
  RequestDispatcher displayPage;
  RequestDispatcher comingPage;

  public void init() throws ServletException
  {
    ServletContext context = getServletContext();
    displayPage = context.getRequestDispatcher(Constants.displayPagePath);
    if (displayPage == null) {
      throw new ServletException(Constants.displayPagePath + " not found");
    }
    comingPage = context.getRequestDispatcher(Constants.comingPagePath);
    if (comingPage == null) {
      throw new ServletException(Constants.comingPagePath + " not found");
    }
  }

  public void doPost (HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException
  { doGet(request, response);
```

```
}

public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
{
  Connection con = null;
  Statement stmt = null;
  ResultSet rs = null;
  HttpSession session = request.getSession();
  String id="";
  id =(String)session.getAttribute("checkmovie");
  Movie movie = null;
  try
  { movie = movie.find(id);
  }
  catch (Exception e)
  { throw new ServletException(e.toString());
  }

        session.setAttribute("movie", movie);
      String movie_nu = movie.getId();
      String title = movie.getTitle();
      String status = movie.getStatus();
      String room_nu = movie.getRoom_nu();
      String genre = movie.getGenre();
      String release = movie.getRelease();
      String end_date = movie.getEnd_date();
      String starring = movie.getStarring();
      String story = movie.getStory();
      session.setAttribute("movie_nu", movie_nu);
      session.setAttribute("title", title );
      session.setAttribute("status", status );
      session.setAttribute("room_nu", room_nu );
      session.setAttribute("genre", genre );
      session.setAttribute("release", release );
      session.setAttribute("end_date", end_date );
      session.setAttribute("starring", starring );
      session.setAttribute("story", story);

      response.sendRedirect("checkmovie.jsp");
  }
}
```

The EditServlet.java

```
package ticket;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.sql.*;
import java.util.*;

public class EditServlet extends HttpServlet
{
  RequestDispatcher searchPage;
  RequestDispatcher displayPage;
  RequestDispatcher comingPage;

  public void init() throws ServletException
  {
   ServletContext context = getServletContext();
   searchPage = context.getRequestDispatcher(Constants.searchPagePath);
   if (searchPage == null) {
     throw new ServletException(Constants.searchPagePath + " not found");
```

```
}
displayPage = context.getRequestDispatcher(Constants.displayPagePath);
if (displayPage == null) {
  throw new ServletException(Constants.displayPagePath + " not found");
}
comingPage = context.getRequestDispatcher(Constants.comingPagePath);
if (comingPage == null) {
  throw new ServletException(Constants.comingPagePath + " not found");
}
}

public void doPost (HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
{ doGet(request, response);
}

public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
{
  Connection con = null;
  Statement stmt = null;
  ResultSet rs = null;
  String key = request.getParameter("key");
  String id = request.getParameter("id");
  HttpSession session = request.getSession();

  if(key.equals("movie") || key.equals("display")|| key.equals("coming"))
  { Movie movie = null;
    try
    { movie = movie.find(id);
    }
    catch (Exception e)
    { throw new ServletException(e.toString());
    }

        session.setAttribute("movie", movie);
      String movie_nu = movie.getId();
      String title = movie.getTitle();
      String status = movie.getStatus();
      String room_nu = movie.getRoom_nu();
      String genre = movie.getGenre();
      String release = movie.getRelease();
      String end_date = movie.getEnd_date();
      String starring = movie.getStarring();
      String story = movie.getStory();
      session.setAttribute("movie_nu", movie_nu);
      session.setAttribute("title", title );
      session.setAttribute("status", status );
      session.setAttribute("room_nu", room_nu );
      session.setAttribute("genre", genre );
      session.setAttribute("release", release );
      session.setAttribute("end_date", end_date );
      session.setAttribute("starring", starring );
      session.setAttribute("story", story);

      if(key.equals("movie"))
      { response.sendRedirect("movie_update.jsp");
      }
      else if(key.equals("coming"))
      { response.sendRedirect(request.getContextPath()+Constants.comingPagePath);
      }
      else if(key.equals("display"))
      { response.sendRedirect(request.getContextPath()+Constants.displayPagePath);
      }
  }//if
```

```
    if (key.equals("member"))
    { Member member = null;
      try
      { member = member.find(id);
      }
      catch (Exception e)
      { throw new ServletException(e.toString());
      }
        session.setAttribute("member", member);
        String mem_id = member.getId();
        String passwd = member.getPasswd();
        String fname = member.getFname();
        String lname = member.getLname();
        String addr = member.getAddr();
        String city = member.getCity();
        String state = member.getState();
        String zip = member.getZip();
        String email = member.getEmail();
        String auth= member.getAuth();

        session.setAttribute("mem_id", mem_id);
        session.setAttribute("fname", fname );
        session.setAttribute("lname", lname );
        session.setAttribute("addr", addr );
        session.setAttribute("city", city );
        session.setAttribute("state", state );
        session.setAttribute("zip", zip );
        session.setAttribute("email", email );
        session.setAttribute("passwd", passwd);
        session.setAttribute("auth", auth);
        response.sendRedirect("member_update.jsp");
    }
  }
}
```

The ErrorTag.java

```
package ticket;
import javax.servlet.jsp.tagext.TagSupport;
import javax.servlet.jsp.JspException;
import javax.servlet.jsp.PageContext;
import javax.servlet.jsp.JspWriter;
import java.io.IOException;

public class ErrorTag extends TagSupport
{
  private String clazz;
  public void setClazz(String clazz) { this.clazz = clazz; }
  public String getClazz() { return clazz; }
  public int doAfterBody() throws JspException
  { String error = (String) pageContext.getAttribute("error", PageContext.REQUEST_SCOPE);

    if (error == null) return EVAL_PAGE;
    try
    { JspWriter out = pageContext.getOut();
      out.println("Hello out there.");
      if (clazz != null)
      { out.println("<div class="" + clazz + ">" + error + "</div>");
      }else
      { out.println(error);
      }
    }catch (IOException e) { throw new JspException(e.toString()); }
    return EVAL_PAGE;
```

```
  }
}
```

The Hiddenfilter.java

```
package ticket;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.IOException;
public class HidingFilter implements Filter
{
  public void init(FilterConfig filterConfig) throws ServletException { }
  public void destroy() { }

  public void doFilter(ServletRequest req,ServletResponse resp,FilterChain chain)
  throws IOException, ServletException
  {
    HttpServletRequest request = (HttpServletRequest) req;
    HttpServletResponse response = (HttpServletResponse) resp;
    HttpSession session = request.getSession();
    response.sendRedirect(request.getContextPath() + "/");
  }
}
```

The HomeServlet.java

```
package ticket;

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.IOException;

public class HomeServlet extends HttpServlet
{
  RequestDispatcher homePage;
  RequestDispatcher adminPage;
  RequestDispatcher memberPage;

  public void init() throws ServletException
  {
    ServletContext context = getServletContext();
    homePage = context.getRequestDispatcher(Constants.homePagePath);
    if (homePage == null) {
      throw new ServletException(Constants.homePagePath + " not found");
    }
    adminPage = context.getRequestDispatcher(Constants.adminPagePath);
    if (adminPage == null) {
      throw new ServletException(Constants.adminPagePath + " not found");
    }
    memberPage = context.getRequestDispatcher(Constants.memberPagePath);
    if (memberPage == null) {
      throw new ServletException(Constants.memberPagePath + " not found");
    }
  }

  public void doPost (HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException
  {
    doGet(request, response);
  }

  protected void doGet(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException
```

```
    {
     HttpSession session = request.getSession();
     String auth = (String)session.getAttribute("auth");

     if(auth.equals("1"))
     {
      response.sendRedirect("http://ticket.ias.csusb.edu:8080/ticket/member/index.jsp");
     }
     if(auth.equals("2"))
     {
      response.sendRedirect("http://ticket.ias.csusb.edu:8080/ticket/admin/index.jsp");
     }
     if(auth.equals("0"))
     { response.sendRedirect("http://ticket.ias.csusb.edu:8080/ticket/index.jsp");
     }
    }
}
```

The LoginServlet.java

```
package ticket;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.IOException;

public class LoginServlet extends HttpServlet
{
  RequestDispatcher homePage;
  RequestDispatcher adminPage;
  RequestDispatcher memberPage;

  public void init() throws ServletException
  {
    ServletContext context = getServletContext();
    homePage = context.getRequestDispatcher(Constants.homePagePath);
    if (homePage == null) {
      throw new ServletException(Constants.homePagePath + " not found");
    }
    adminPage = context.getRequestDispatcher(Constants.adminPagePath);
    if (adminPage == null) {
      throw new ServletException(Constants.adminPagePath + " not found");
    }
    memberPage = context.getRequestDispatcher(Constants.memberPagePath);
    if (memberPage == null) {
      throw new ServletException(Constants.memberPagePath + " not found");
    }
  }

  protected void doPost(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException
  {
    String userid = request.getParameter("id");
    if (userid.equals(""))
    { request.setAttribute("error", "User ID is a required.");
      homePage.forward(request, response);
      return;
    }
    String password = request.getParameter("passwd");
    if (password.equals(""))
    { request.setAttribute("error", "User password is required.");
      homePage.forward(request, response);
      return;
     }
```

```
    User user = null;
    try
    { user = User.find(userid);

    }catch (Exception e)
    { throw new ServletException(e.toString());
    }
    if(user == null)
    { request.setAttribute("error", "User ID doesn't exist.");
      homePage.forward(request, response);
      return;
    }else if (!user.getPassword().equals(password))
    { request.setAttribute("error", "Password incorrect.");
      homePage.forward(request, response);
      return;
    }

    HttpSession session = request.getSession();
    session.setAttribute("user", user);
    String id = user.getUserid();
    String auth = user.getAuth();
    session.setAttribute("userid", id);
    session.setAttribute("auth", auth);

    if(user.getAuth().equals("1"))
    {
      response.sendRedirect(request.getContextPath() + Constants.memberPagePath);
    }
    if(user.getAuth().equals("2"))
    {
      response.sendRedirect(request.getContextPath() + Constants.adminPagePath);
    }
  }
}
```

The LogoutServlet.java

```
package ticket;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.IOException;

public class LogoutServlet extends HttpServlet
{
  public void init() throws ServletException
  {
  }

  protected void doGet(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException
  {
    HttpSession session = request.getSession();
    session.invalidate();

    String contextPath = request.getContextPath();
    response.sendRedirect(contextPath + "/");
  }
}
```

The MailServlet.java

```
package ticket;
```

```
import javax.servlet.*;
import javax.mail.*;
import javax.mail.internet.*;
import javax.servlet.http.*;
import javax.activation.*;
import java.sql.*;
import java.io.*;
import java.util.*;

public class MailServlet extends HttpServlet
{  RequestDispatcher mailPage;

  public void init() throws ServletException
  {  ServletContext context = getServletContext();
    mailPage = context.getRequestDispatcher(Constants.mailPagePath);
    if (mailPage == null)
    {  throw new ServletException(Constants.mailPagePath + " not found");
    }
  }

  public void doPost (HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  {  doGet(request, response);
  }

  public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
  {  String id = request.getParameter("id");
    String mail = "";
    String text = "";
    protected String from = Constants.FROM_KEY;
    protected String smtp= Constants.SMTP_KEY;
    protected String subject= Constants.SUBJ_KEY;

      protected void setFromEmail(String fromEmail)
    {  this.fromEmail=fromEmail;
    }

    protected void setSubject(String subject)
    {  this.subject=subject;
      }

      protected void setSmtp(String smtp)
    {  this.smtp=smtp;
      }

   if ((id.equals("")) || (id == null))
   {  request.setAttribute("error", "Please type your User ID");
    mailPage.forward(request, response);
    return;
   }

   Booking booking=null;
   booking= new Booking();
   booking.createAccount();
   String book = booking.getAccount();

   Connection con=null;
   String sql= "";
   Statement stmt=null;
   ResultSet rs=null;
   String url="http://ticket.ias.csusb.edu:8080/ticket/change?id="+id+"&pass="+book;

   try
   {  Class.forName("org.postgresql.Driver").newInstance();
       con = DriverManager.getConnection("jdbc:postgresql://127.0.0.1/ticket?user=ticket&password=762316");
       sql = "select passwd, email from member where mem_id = '"+ id +"';";
```

```
        stmt = con.createStatement();
        rs = stmt.executeQuery(sql);
        if(rs.next())
        {  mail = rs.getString("email");
           text = "Please click on the link to modify your password.\n\n"+url+"\n";
        Properties props = new Properties();
        props.put("mail.smtp.host", smpt);
        Session s = Session.getInstance(props, null);
           MimeMessage msg = new MimeMessage(s);
           InternetAddress from = new InternetAddress("fromEmail");
           msg.setFrom(from);
           InternetAddress to = new InternetAddress();
           to = new InternetAddress(mail);
           msg.addRecipient(Message.RecipientType.TO, to);
        msg.setSubject(subject);
        msg.setText(text);
        Transport.send(msg);
        } else
        { request.setAttribute("error", "Cannot retrieve your password, Please check your input User ID");
       mailPage.forward(request, response);
       return;
        }
    }catch (Exception e)
    {  String error = "Sent mail error !";
         throw new IOException(error);
    }

    PreparedStatement pstmt= null;
    try
    {  Class.forName("org.postgresql.Driver").newInstance();
      con = DriverManager.getConnection("jdbc:postgresql://127.0.0.1/ticket?user=ticket&password=762316");
      sql="insert into passwd values (?,?);";
      pstmt = con.prepareStatement(sql);
      pstmt.setString(1,id);
      pstmt.setString(2,book);
      pstmt.executeUpdate();
      pstmt.close();
      con.close();
    }catch (Exception e)
    {  String error = "create account error !";
       throw new IOException(error);
    }
    response.sendRedirect("index.jsp");
  }
}
```

The Member.java

```
package ticket;
import java.util.Hashtable;
import javax.naming.*;
import java.sql.*;
import javax.sql.DataSource;
import java.io.IOException;

public class Member
{
  private String id;
  private String passwd;
  private String fname;
  private String lname;
  private String addr;
  private String city;
  private String state;
```

```
  private String zip;
  private String email;
  private String auth;

  public Member(String id, String passwd, String auth,String fname, String lname, String addr,String city, String state, String
  zip,String email)
  {  this.id = id;
     this.passwd = passwd;
     this.fname = fname;
     this.lname = lname;
     this.addr = addr;
     this.city = city;
     this.state = state;
     this.zip = zip;
     this.email = email;
     this.auth = auth;
  }

  public String getId() { return id; }
  public String getPasswd() { return passwd; }
  public String getFname() { return fname; }
  public String getLname() { return lname; }
  public String getAddr() { return addr; }
  public String getCity() { return city; }
  public String getState() { return state; }
  public String getZip() { return zip; }
  public String getEmail() { return email; }
  public String getAuth() { return auth; }

  public static Member find(String id) throws Exception
  {  Member member = null;
     Connection Conn= null;
     Statement stmt= null;
     try
     {
         Class.forName("org.postgresql.Driver").newInstance();
         Conn = DriverManager.getConnection("jdbc:postgresql://127.0.0.1/ticket?user=ticket&password=762316");
         stmt= Conn.createStatement();
         String qs="select * from member where mem_id = '" + id + "';";
         ResultSet rs = stmt.executeQuery(qs);
      while(rs.next())
      {
       String memberid= rs.getString("mem_id");
       String passwd = rs.getString("passwd");
       String fname= rs.getString("firstname");
       String lname = rs.getString("LastName");
       String addr = rs.getString("address");
       String city = rs.getString("city");
       String state = rs.getString("state");
       String zip = rs.getString("zip_code");
       String email = rs.getString("email");
       String auth= rs.getString("auth");
       member = new Member(memberid, passwd, auth, fname,lname,addr,city,state,zip,email);
      }
     }finally
     { stmt.close();
         Conn.close();
     }
     return member;
   }
}
```

The Movie.java

```
package ticket;
import java.util.Hashtable;
import javax.naming.*;
import java.sql.*;
import javax.sql.DataSource;
import java.io.IOException;

public class Movie
{
  private String id;
  private String title;
  private String status;
  private String room_nu;
  private String genre;
  private String release;
  private String end_date;
  private String starring;
  private String story;

  public Movie(String id, String title, String status,String room_nu, String genre, String release,String end_date, String starring,
String story)
  { this.id = id;
    this.title = title;
    this.status = status;
    this.room_nu = room_nu;
    this.genre = genre;
    this.release = release;
    this.end_date = end_date;
    this.starring = starring;
    this.story = story;
  }

  public String getId() { return id; }
  public String getTitle() { return title; }
  public String getStatus() { return status; }
  public String getRoom_nu() { return room_nu; }
  public String getGenre() { return genre; }
  public String getRelease() { return release; }
  public String getEnd_date() { return end_date; }
  public String getStarring() { return starring; }
  public String getStory() { return story; }

  public static Movie find(String id) throws Exception
  { Movie movie = null;
    Connection Conn= null;
    Statement stmt= null;
    try
    {
        Class.forName("org.postgresql.Driver").newInstance();
        Conn = DriverManager.getConnection("jdbc:postgresql://127.0.0.1/ticket?user=ticket&password=762316");
        stmt= Conn.createStatement();
        String qs="select * from movie where movie_nu = '" + id + "';";
        ResultSet rs = stmt.executeQuery(qs);
     while(rs.next())
     {
      String movie_nu= rs.getString("movie_nu");
      String title = rs.getString("movie_title");
      String status= rs.getString("status");
      String room_nu = rs.getString("room_nu");
      String genre = rs.getString("genre");
      String release = rs.getString("release");
      String end_date = rs.getString("end_date");
      String starring = rs.getString("starring");
      String story = rs.getString("story");
      movie = new Movie(movie_nu, title, status, room_nu, genre, release, end_date, starring, story);
```

```
    }
    }finally
    { stmt.close();
        Conn.close();
    }
    return movie;
  }
}
```

The MovieupdateServlet.java

```
package ticket;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;
import java.io.*;

public class MovieupdateServlet extends HttpServlet
{
  RequestDispatcher adminPage;
  RequestDispatcher movieupdatePage;

  public void init() throws ServletException
  { ServletContext context = getServletContext();
    adminPage = context.getRequestDispatcher(Constants.adminPagePath);
    if (adminPage == null) {
      throw new ServletException(Constants.adminPagePath + " not found");
    }
    movieupdatePage = context.getRequestDispatcher(Constants.movieupdatePagePath);
    if (movieupdatePage == null) {
      throw new ServletException(Constants.movieupdatePagePath + " not found");
    }
  }

  public void doPost (HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException
  { doGet(request, response);
  }

  public void doGet(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException
  { String id ="";
    id= request.getParameter("id");
    String title="";
    title = request.getParameter("title");
    String status= "";
    status = request.getParameter("status");
    String room_nu="";
    room_nu = request.getParameter("room_nu");
    String genre = "";
    genre= request.getParameter("genre");
    String release="";
    release = request.getParameter("release");
    String end_date="";
    end_date = request.getParameter("end_date");
    String starring="";
    starring = request.getParameter("starring");
    String story="";
    story =request.getParameter("story");
    Connection con= null;
    String sql= null;
    Statement stmt= null;
    ResultSet rs= null;
```

```
try
{ con = DriverManager.getConnection("jdbc:postgresql://127.0.0.1/ticket?user=ticket&password=762316");
        sql = ("update movie set  movie_title="'+title+'", status="'+status+'", room_nu="'+room_nu+'",
    genre="'+genre+'", release="'+release+'", end_date="'+end_date+'", starring="'+starring+'", story="'+story+'"
    where movie_nu="'+id+'";");
        stmt = con.createStatement();
        stmt.executeUpdate(sql);
}catch (SQLException e)
{  String error = " error !";
    throw new ServletException(error);
    }finally
    { try
     { stmt.close();
   con.close();
        }catch(SQLException ingored){}
    }

String[] run=request.getParameterValues("runtime") ;
if(run != null){
try
{  Class.forName("org.postgresql.Driver").newInstance();
  con = DriverManager.getConnection("jdbc:postgresql://127.0.0.1/ticket?user=ticket&password=762316");
  stmt = con.createStatement();
  sql = " delete from movie_time where movie_nu="'+id+'";";
  stmt.executeUpdate(sql);
}catch (Exception e)
{ String error = "create account error !";
    throw new IOException(error);
}finally{
try
{ stmt.close();
        con.close();
    }catch(SQLException ingored){}
}

for (int i=0;i <run.length; i++)
{
try
{ con = DriverManager.getConnection("jdbc:postgresql://127.0.0.1/ticket?user=ticket&password=762316");
 sql=("insert into movie_time values (?,?);");
 PreparedStatement pstmt = con.prepareStatement(sql);
 pstmt.setString(1, id);
 pstmt.setString(2, run[i]);
 pstmt.executeUpdate();
}catch (SQLException e)
{ String error = " error !";
    throw new ServletException(error);
}
finally{
    try
    { con.close();
    }catch(SQLException ingored){}
}
}
}
response.sendRedirect(request.getContextPath()+Constants.adminPagePath);
}
}
```

The OrderInfoServlet.java

```
package ticket;
import javax.servlet.*;
import javax.servlet.http.*;
```

```
import java.io.*;
import java.sql.*;
import java.util.*;

public class OrderInfoServlet extends HttpServlet
{
  public void init() throws ServletException
  { ServletContext context = getServletContext();
  }

  public void doPost (HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException
  { doGet(request, response);
  }

  public void doGet(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException
  { String id = (request.getParameter("id")).trim();
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    Connection con = null;
    Statement stmt = null;
    ResultSet rs = null;

    try
    { Class.forName("org.postgresql.Driver");
      con = DriverManager.getConnection("jdbc:postgresql://127.0.0.1/ticket?user=ticket&password=762316");
      stmt = con.createStatement();
      rs = stmt.executeQuery("select book_nu, book_date, movie_date, movie_ time,movie_ title from books where
      mem_id="+id+";");
      out.println("<html>");
      out.println("<head><title> Order Information </title></head>");
      out.println("<body>");
      out.println("<table width=60% border=0 cellspacing=1>");
      out.println("<tr> <td width=20%> </td><td width=80%> </td></tr>");
      out.println("<tr><td colspan=2><div align=left><strong><font size=+1>You have
      booked : </font></strong></div></td></tr>");
      out.println("<tr> <td width=20%> </td><td width=80%> </td></tr>");
      while (rs.next())
      { String book_nu = rs.getString("book_nu");
        String bookdate = rs.getString("book_date");
        String moviedate = rs.getString("movie_date");
        String movietime= rs.getString("movie_time");
        String title= rs.getString("movie_title");

        out.println("<tr><td colspan=2><font color=000066>Movie Title:</font> "+title+"</td></tr>");
        out.println("<tr><td colspan=2><font color=000066>Book#</font> "+book_nu+"</td></tr>");
        out.println("<tr><td colspan=2><font color=000066>Book Date:</font>  "+bookdate+"</td></tr>");
        out.println("<tr><td colspan=2><font color=000066>Movie Date:</font>  "+moviedate+"</td></tr>");
        out.println("<tr><td colspan=2><font color=000066>Movie Time:</font>  "+movietime+"</td></tr>");
        out.println("<tr> <td width=20%> </td><td width=80%> </td></tr>");
       }
        out.println("<tr><td colspan=2><font color=000066>If you wish to inquire about the status of your order, please
        indicate </font></td></tr>");
        out.println("<tr><td colspan=2><font color=000066>your book#, the movie you are inquiring
        bout .</font></td></tr>");
        out.println("<tr><td colspan=2> </td></tr>");
        out.println("</table> ");
        out.println("<a href=index.jsp>Member Home</a>  <a href=logout>Log Out</a>");
        out.println("</body></html>");
      }catch (ClassNotFoundException e)
      { out.println("can't load database driver"  + e.getMessage());
      }
      catch (SQLException e)
      { out.println("SQLException: "  + e.getMessage());
```

```
    }finally
    { try
     { if(con != null) con.close();
     }
     catch(SQLException ignored){}
    }
  }
}
```

The SearchServlet.java

```
package ticket;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.sql.*;

public class SearchServlet extends HttpServlet
{
  RequestDispatcher searchPage;

  public void init() throws ServletException
  {
    ServletContext context = getServletContext();
    searchPage = context.getRequestDispatcher(Constants.searchPagePath);
    if (searchPage == null) {
      throw new ServletException(Constants.searchPagePath + " not found");
    }
  }

  public void doPost (HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException
  { doGet(request, response);
  }

  public void doGet(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException
  { Connection con = null;
    Statement stmt = null;
    ResultSet rs = null;
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String key = request.getParameter("key");
    String[] memid = request.getParameterValues("memid");
    if(key.equals("member"))
    {
     if(memid != null)
     {
      try
      { Class.forName("org.postgresql.Driver");
        con = DriverManager.getConnection("jdbc:postgresql://127.0.0.1/ticket?user=ticket&password=762316");
        stmt = con.createStatement();
        for(int i=0; i<memid.length; i++)
        {
          String sql = ("delete from member where mem_id='" + memid [i]+ "';");
          stmt.executeUpdate(sql);
          stmt.close();
        }
      }
      catch (Exception e)
      {
        String error = "delete member error !";
        throw new IOException(error);
      }
```

```
}

try
{ Class.forName("org.postgresql.Driver");
 con = DriverManager.getConnection("jdbc:postgresql://127.0.0.1/ticket?user=ticket&password=762316");
 stmt = con.createStatement();
 rs = stmt.executeQuery("select mem_id, firstname, lastname, auth from member order by mem_id");
 out.println("<html>");
 out.println("<head><title> search member </title></head>");
 out.println("<body><form action=search method=post>");
 out.println("<table  width=70%  border=0 align=center cellpadding=3 cellspacing=1>");
 out.println("<tr bgcolor=#33CCFF> <td colspan=5 <div align=center><strong><font size=+1>List of Member");
 out.println("</font></strong></div></td></tr>");//first row
 out.println("<tr bgcolor=#FFFF00>");
 out.println("<td width=20%><div align=center><font color=#000033><strong>User
 ID</strong></font></div></td>");
 out.println("<td width=35%><div align=center><font
 color=#000033><strong>Name</strong></font></div></td>");
 out.println("<td width=15%><div align=center><font
 color=#000033><strong>Auth</strong></font></div></td>");
 out.println("<td width=15%><div align=center><font color=#000033><strong>Edit</strong></font></div></td>");
 out.println("<td width=15%><div align=center><font
 color=#000033><strong>Delete</strong></font></div></td>");
 out.println("</tr>");//secnod row

 while (rs.next())
 { String id = rs.getString("mem_id");
  String fname = rs.getString("firstname");
  String lname = rs.getString("lastname");
  String auth= rs.getString("auth");

  out.println("<tr bgcolor=#CCFFFF>");
  out.println("<td width=20%><div align=center> "+ id +"</strong></font></div></td>");
  out.println("<td width=35%><div align=center> "+ fname +" "+ lname +"</strong></font></div></td>");
  out.println("<td width=15%><div align=center> "+ auth +"</strong></font></div></td>");
  out.println("<td width=15%><div align=center> <a href=edit?id="+ id +"&key=member>Edit</a></div></td>");
  out.println("<td width=15%><div align=center><input type=checkbox name=memid value="+id+"> <input
  type=hidden name=key value=member> </div></td>");
  out.println("</tr>");//third row
 }
  out.println("<tr bgcolor=#CCFFFF>");
  out.println("<td width=20%>  </td>");
  out.println("<td width=35%>  </td>");
  out.println("<td width=15%>  </td>");
  out.println("<td width=15%> </td>");
  out.println("<td width=15%><div align=center><input type=submit value=Delete> </div></td>");
  out.println("</tr>");//fourth row
  out.println("</table></form>");
  out.println("<a href=index.jsp>Admin Home</a>  <a href=logout>Log Out</a>");
  out.println("</body></html>");
}catch (ClassNotFoundException e)
{ out.println("can't load database driver"  + e.getMessage());
}catch (SQLException e)
{ out.println("SQLException: "  + e.getMessage());
}finally
{ try
  {  if(con != null) con.close();
  }catch(SQLException ignored){}
 }
}

String[] movie_id= request.getParameterValues("movieid");
if(key.equals("movie"))
{
 try
```

```
{  Class.forName("org.postgresql.Driver");
  con = DriverManager.getConnection("jdbc:postgresql://127.0.0.1/ticket?user=ticket&password=762316");
  stmt = con.createStatement();

  if(movie_id != null)
  {  for(int i=0; i<movie_id.length; i++)
     {  String sql = ("delete from movie where movie_nu=" + movie_id [i]+ ";");
        stmt.executeUpdate(sql);
        stmt.close();
     }
  }
 }catch (Exception e)
 { String error = "delete movie error !";
  throw new IOException(error);
 }
 try
 {  Class.forName("org.postgresql.Driver");
   con = DriverManager.getConnection("jdbc:postgresql://127.0.0.1/ticket?user=ticket&password=762316");
   stmt = con.createStatement();
   if(movie_id != null)
   { for(int i=0; i<movie_id.length; i++)
     { String sql = ("delete from movie_time where movie_nu=" + movie_id [i]+ ";");
       stmt.executeUpdate(sql);
       stmt.close();
     }
   }
 }catch (Exception e)
 { String error = "delete movie error !";
   throw new IOException(error);
 }

  try
  {  Class.forName("org.postgresql.Driver");
     con = DriverManager.getConnection("jdbc:postgresql://127.0.0.1/ticket?user=ticket&password=762316");
     stmt = con.createStatement();
     rs = stmt.executeQuery("select movie_nu, movie_title, status from movie order by movie_nu");

     out.println("<html>");
     out.println("<head><title> search movie </title><meta http-equiv=Content-Type content=text/html></head>");
     out.println("<body><form action=search method=post>");
     out.println("<table  width=70%  border=0 align=center cellpadding=3 cellspacing=1>");
     out.println("<tr bgcolor=#33CCFF > <td colspan=5> <div align=center><strong><font size=+1>List of
     Movie");
     out.println("</font></strong></div></td></tr>");//first row
     out.println("<tr bgcolor=#FFFF00>");
     out.println("<td width=15%><div align=center><font
     color=#000033><strong>Movie#</strong></font></div></td>");
     out.println("<td width=40%><div align=center><font color=#000033><strong>Movie
     Title</strong></font></div></td>");
     out.println("<td width=15%><div align=center><font
     color=#000033><strong>Status</strong></font></div></td>");
     out.println("<td width=15%><div align=center><font
     color=#000033><strong>Edit</strong></font></div></td>");
     out.println("<td width=15%><div align=center><font
     color=#000033><strong>Delete</strong></font></div></td>");
     out.println("</tr>");//secnod row

     while (rs.next())
     { String number = rs.getString("movie_nu");
       String title = rs.getString("movie_title");
       String status = rs.getString("status");

       out.println("<tr bgcolor=#CCFFFF>");
       out.println("<td width=15%><div align=center> "+ number +"</strong></font></div></td>");
       out.println("<td width=40%><div align=center> "+ title +"</strong></font></div></td>");
```

```
            out.println("<td width=15%><div align=center> "+ status +"</strong></font></div></td>");
            out.println("<td width=15%><div align=center> <a href=edit?id="+ number
            +"&key=movie>Edit</a></div></td>");
            out.println("<td width=15%><div align=center><input type=checkbox name=movieid value="+number+">
            <input type=hidden name=key value=movie> </div></td>");
            out.println("</tr>");//third row
          }
            out.println("<tr bgcolor=#CCFFFF>");
            out.println("<td width=15%>  </td>");
            out.println("<td width=40%> </td>");
            out.println("<td width=15%> </td>");
            out.println("<td width=15%> </td>");
          out.println("<td width=15%><div align=center><input type=submit value=Delete> </div></td>");
            out.println("</tr>");//fourth row
            out.println("</table></form>");
            out.println("<a href=index.jsp>Admin Home</a>  <a href=logout>Log Out</a>");
            out.println("</body></html>");
        }catch (ClassNotFoundException e)
        { out.println("can't load database driver"  + e.getMessage());
        }
        catch (SQLException e)
        { out.println("SQLException: "  + e.getMessage());
        }finally
        { try
          {  if(con != null) con.close();
          }catch(SQLException ignored){}
        }
      }
    }
}
```

The SubmitPaymentServlet.java

```
package ticket;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.sql.*;
import java.util.*;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;

public class SubmitPaymentServlet extends HttpServlet
{
  private Log log = LogFactory.getLog(this.getClass());
  RequestDispatcher orderconfirmPage;

  public void init() throws ServletException
  {
    ServletContext context = getServletContext();
    orderconfirmPage = context.getRequestDispatcher(Constants.orderconfirmPagePath);
    if (orderconfirmPage == null) {
      throw new ServletException(Constants.orderconfirmPagePath + " not found");
    }
  }

  public void doPost (HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException
  {  doGet(request, response);
  }

  public void doGet(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException
  {   HttpSession session = request.getSession();
```

```
String userid = (String) session.getAttribute("userid");
String id = ((String) session.getAttribute("id")).trim();
String title = (String) session.getAttribute("title");
String room_nu = (String) session.getAttribute("room_nu");
String moviedate = ((String) session.getAttribute("moviedate")).trim();
String amount ="$"+(String) session.getAttribute("amount");
String time = ((String) session.getAttribute("time")).trim();
String adult = (String) session.getAttribute("adult");
String child = (String) session.getAttribute("child");
String senior = (String) session.getAttribute("senior");
Calendar now= Calendar.getInstance();
String year = String.valueOf(now.get(Calendar.YEAR));
String month = String.valueOf(now.get(Calendar.MONTH)+1);
String day = String.valueOf(now.get(Calendar.DATE));
String book_date = month+"/"+day+"/"+year;
int a1=Integer.parseInt((String)session.getAttribute("adult")) ;
int c1=Integer.parseInt((String)session.getAttribute("child")) ;
int s1=Integer.parseInt((String)session.getAttribute("senior")) ;
int se=0,cq=0;
Booking booking=null;
booking= new Booking();
booking.createAccount();
String book = booking.getAccount();

Addseat addseat=null;
Check check=null;
Updateseat updateseat=null;
Updatebook updatebook=null;
try
{  check =Check.find(id,moviedate,time);
}catch (Exception e)
{ String error = "id= " + id + " moviedate= " + moviedate + " time= " + time;
  throw new ServletException(error);
}

int seat = check.getSeat();
int total= a1+c1+s1+seat ;
if(seat != cq)
{
  try
  { updateseat=Updateseat.editseat(id,moviedate,time,total);
  }catch (Exception e)
  { String error ="can't update seat to database";
    throw new ServletException(error);
  }
}
else
{
  try
  {  addseat= Addseat.add(id,moviedate,time,total);
  }catch (Exception e)
  { String error ="can't add seat to database";
    throw new ServletException(error);
  }
}

try
{  updatebook= Updatebook.add(book,book_date,moviedate,time,amount,userid,title);
}catch (Exception e)
{  String error ="can't add seat to database";
   throw new ServletException(error);
}
session.setAttribute("book", book);
session.setAttribute("book_date", book_date );
response.sendRedirect("thx.jsp");
```

```
  }
}
```

The Updatebook.java

```
package ticket;
import java.util.Hashtable;
import javax.naming.*;
import java.sql.*;
import javax.sql.DataSource;
import java.io.IOException;

public  class Updatebook
{
  private String book;
  private String book_date;
  private String moviedate;
  private String time;
  private String amount;
  private String userid;
  private String title;

  public Updatebook(String book,String book_date, String moviedate, String time,String amount,String userid,String title)
  {  this.book = book;
    this.book_date = book_date;
    this.moviedate = moviedate;
    this.time = time;
    this.amount = amount;
    this.userid = userid;
    this.title = title;
  }

  public String getBook() { return book; }
  public String getBook_date() { return book_date; }
  public String getMoviedate() { return moviedate; }
  public String getTime() { return time; }
  public String getAmount() { return amount; }
  public String getUserid() { return userid; }
  public String getTitle() { return title; }

  public Updatebook add(String book,String book_date, String moviedate, String time,String amount,String userid,String title)
  throws Exception
  {  Updatebook updatebook= null;
    Connection con= null;
    PreparedStatement pstmt= null;
    try
    {
        Class.forName("org.postgresql.Driver").newInstance();
     con = DriverManager.getConnection("jdbc:postgresql://127.0.0.1/ticket?user=ticket&password=762316");\
     String sql="insert into books values (?,?,?,?,?,?,?);";
     pstmt = con.prepareStatement(sql);
     pstmt.setString(1,book);
     pstmt.setString(2,book_date);
     pstmt.setString(3,moviedate);
     pstmt.setString(4,time);
     pstmt.setString(5,amount);
     pstmt.setString(6,userid);
     pstmt.setString(7,title);
     pstmt.executeUpdate();
     pstmt.close();\
    }catch (SQLException e)
    { String error = " error updatebooking !";
          throw new IOException(error);
      }
```

```
        return updatebook;
    }
}
```

The Updateseat.java

```
package ticket;
import java.util.Hashtable;
import javax.naming.*;
import java.sql.*;
import javax.sql.DataSource;
import java.io.IOException;

public  class Updateseat
{
  private String id;
  private String moviedate;
  private String time;
  private int seat;

  public Updateseat(String id, String moviedate, String time, int seat)
  {  this.id = id;
     this.moviedate = moviedate;
     this.time = time;
     this.seat = seat;
  }

  public String getId() { return id; }
  public String getMoviedate() { return moviedate; }
  public String getTime() { return time; }
  public int getSeat() { return seat; }

  public Updateseat editseat(String id, String moviedate, String time, int total) throws Exception
  {  Updateseat updateseat= null;
     Connection Conn= null;
     PreparedStatement pstmt= null;
     try
     {
         Class.forName("org.postgresql.Driver").newInstance();
         Conn = DriverManager.getConnection("jdbc:postgresql://127.0.0.1/ticket?user=ticket&password=762316");
         String qs="update seat set seat="+total+" where movie_nu = '" + id + "' and moviedate= '"+moviedate+"' and time=
       '"+time+"';";
       pstmt= Conn.prepareStatement(qs);
         pstmt.executeUpdate();
       pstmt.close();
     }catch (SQLException e)
     { String error = " error checking !";
            throw new IOException(error);
        }
        return updateseat;
  }
}
```

The UpdateServlet.java

```
package ticket;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;
import java.io.*;

public class UpdateServlet extends HttpServlet
{
```

```
RequestDispatcher adminupdatePage;
RequestDispatcher memberupdatePage;
RequestDispatcher adminPage;
RequestDispatcher memberPage;

public void init() throws ServletException
{
  ServletContext context = getServletContext();
  adminupdatePage = context.getRequestDispatcher(Constants.adminupdatePagePath);
  if (adminupdatePage == null) {
    throw new ServletException(Constants.adminupdatePagePath + " not found");
  }
  memberupdatePage = context.getRequestDispatcher(Constants.memberupdatePagePath);
  if (memberupdatePage == null) {
    throw new ServletException(Constants.memberupdatePagePath + " not found");
  }
  adminPage = context.getRequestDispatcher(Constants.adminPagePath);
  if (adminPage == null) {
    throw new ServletException(Constants.adminPagePath + " not found");
  }
  memberPage = context.getRequestDispatcher(Constants.memberPagePath);
  if (memberPage == null) {
    throw new ServletException(Constants.memberPagePath + " not found");
  }
}

public void doPost (HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
{ doGet(request, response);
}

public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
{ String id = (String) request.getParameter ("id");
 String fname = (String) request.getParameter("fname");
 String lname = (String) request.getParameter("lname");
 String addr = (String) request.getParameter("addr");
 String city = (String) request.getParameter("city");
 String state = (String) request.getParameter("state");
 String zip = (String) request.getParameter("zip");
 String email = (String) request.getParameter("email");
 String auth = (String) request.getParameter("auth");
 String host = (String) request.getParameter("host");
 String key = (String) request.getParameter("key");
 String passwd = (String) request.getParameter("passwd");
 String repasswd = (String) request.getParameter("repasswd");

 if(key.equals("member") && host.equals("mem"))
{
  if (!passwd.equals(repasswd))
  { request.setAttribute("error", "Password and Re-type Password are not matched.");
    memberupdatePage.forward(request, response);
      return;
  }else if (email.equals(""))
  { request.setAttribute("error", "E-Mail is required.");
    memberupdatePage.forward(request, response);
    return;
  }
}

if( host.equals("admin") && key.equals("member"))
{
  if (!passwd.equals(repasswd))
  { request.setAttribute("error", "Password and Re-type Password are not matched.");
    adminupdatePage.forward(request, response);
```

```
    return;
  }else if (email.equals("") || email == null)
  { request.setAttribute("error", "E-Mail is required.");
    adminupdatePage.forward(request, response);
    return;
  }
 }

 Connection con= null;
 String sql= null;
 Statement stmt= null;
 try
 { Class.forName("org.postgresql.Driver");
  con = DriverManager.getConnection("jdbc:postgresql://127.0.0.1/ticket?user=ticket&password=762316");
  if(passwd == null || passwd.equals(""))
  { sql = "UPDATE member set FirstName='"+fname +"',LastName='"+lname+"',Address='"+ addr
    +"',City='"+city+"',State='"+ state +"', Zip_Code='" + zip + "', Email='" + email + "',auth='"+auth+"' Where Mem_Id='"
    + id + "';";
  }
  else
  {
  sql = "UPDATE member set Passwd='"+ passwd +"', FirstName='"+fname +"',LastName='"+lname+"',Address='"+ addr
  +"',City='"+city+"',State='"+ state +"', Zip_Code='" + zip + "', Email='" + email + "',auth='"+auth+"' Where Mem_Id='" +
  id + "';";
  }
  stmt = con.createStatement();
  stmt.executeUpdate(sql);
 }catch (Exception e)
 { String error = "create account error !";
     throw new IOException(error);
 }finally{
     try
     { con.close();
     }catch(SQLException ingored){}
 }

 if(auth.equals("1"))
 { response.sendRedirect(request.getContextPath()+Constants.memberPagePath);
 }
 if(auth.equals("2"))
 { response.sendRedirect(request.getContextPath()+Constants.adminPagePath);
 }
 }

}
```

The UpdatetopServlet.java

```
package ticket;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;
import java.io.*;

public class UpdateTopServlet extends HttpServlet
{
  RequestDispatcher adminupdatePage;

  public void init() throws ServletException
  {
    ServletContext context = getServletContext();
    adminupdatePage = context.getRequestDispatcher(Constants.adminupdatePagePath);
    if (adminupdatePage == null) {
      throw new ServletException(Constants.adminupdatePagePath + " not found");
```

```
    }
  }

  public void doPost (HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException
  {  doGet(request, response);
  }

  public void doGet(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException
  { String top = (String) request.getParameter ("top");
   String t1 = (String) request.getParameter("t1");
   String t2 = (String) request.getParameter("t2");
   String t3 = (String) request.getParameter("t3");
   String t4 = (String) request.getParameter("t4");
   String t5 = (String) request.getParameter("t5");
   String t6 = (String) request.getParameter("t6");
   String t7 = (String) request.getParameter("t7");
   String t8 = (String) request.getParameter("t8");
   String t9 = (String) request.getParameter("t9");
   String t0 = (String) request.getParameter("t0");
   String sql="";
   try
   {
   Class.forName("org.postgresql.Driver");
   Connection con = DriverManager.getConnection("jdbc:postgresql://127.0.0.1/ticket?user=ticket&password=762316");
   Statement stmt=con.createStatement();
   sql = ("delete from top10 where top=" +top+ ";");
   stmt.executeUpdate(sql);
   stmt.close();
   }catch (Exception e)
   { String error = "delete top10 error !";
        throw new IOException(error);
   }

   PreparedStatement pstmt;
   try
   {
   Class.forName("org.postgresql.Driver").newInstance();
   Connection con = DriverManager.getConnection("jdbc:postgresql://127.0.0.1/ticket?user=ticket&password=762316");
   sql=("insert into top10 values (?,?,?,?,?,?,?,?,?,?,?);");
   pstmt = con.prepareStatement(sql);
   pstmt.setString(1, top);
   pstmt.setString(2, t1);
   pstmt.setString(3, t2);
   pstmt.setString(4, t3);
   pstmt.setString(5, t4);
   pstmt.setString(6, t5);
   pstmt.setString(7, t6);
   pstmt.setString(8, t7);
   pstmt.setString(9, t8);
   pstmt.setString(10,t9);
   pstmt.setString(11,t0);
   pstmt.executeUpdate();
   pstmt.close();
   }catch (Exception e)
   { String error = "create account error !";
        throw new IOException(error);
   }
    response.sendRedirect("index.jsp");
  }
}
```

The User..java

```
package ticket;
import java.util.Hashtable;
import javax.naming.*;
import java.sql.*;
import javax.sql.DataSource;
import java.io.IOException;

public class User
{
  private String userid;
  private String password;
  private String auth;

  public User(String userid, String password, String auth)
  {  this.userid = userid;
     this.password = password;
     this.auth = auth;
  }

  public String getUserid() { return userid; }
  public String getPassword() { return password; }
  public String getAuth() { return auth; }

  public static User find(String userid) throws Exception
  {
     User user = null;
     Connection Conn= null;
     Statement stmt= null;
     try
     { Class.forName("org.postgresql.Driver").newInstance();
          Conn = DriverManager.getConnection("jdbc:postgresql://127.0.0.1/ticket?user=ticket&password=762316");
          stmt= Conn.createStatement();
          String qs="select * from member where mem_id = '" + userid + "';";
          ResultSet rs = stmt.executeQuery(qs);
      if (!rs.first()) return null;
      String password = rs.getString("passwd");
      String auth= rs.getString("auth");
      user = new User(userid, password, auth);
     }finally
     { stmt.close();
          Conn.close();
     }
   return user;
  }
}
```

REFERENCES


[1] "Beginning JSP Web Development." Jayson Falkner. August 2001.

[2] IEEE Recommended Practice for Software Requirements Specifications (IEEE Std 830-1993).

[3] "JavaScript: The Definitive Guide 4th edition." David Flanagan. O'Reilly & Associates; December, 2001

[4] "JavaServer Pages." Larne Pekowsky. April 2000.

[5] "*Java Servlet Programming Second Edition*." Jason Hunter and William Crawford. O'Reilly and Assoicates, 2002

[6] "JavaSpaces Principles, Patterns, and Practice." Eric Freeman. Novermber 1999.

[7] "PostgreSQL: Introduction and Concepts", Bruce Momjian. December 2000. <http://postgresql.org/docs>

[8] SSL 3.0 Specification. http://wp.netscape.com/eng/ssl3/, November 1996.

[9] "The Java Programming Language Second Edition." Ken Arnold and James Gosling. February 2000.

[10] "UML Distilled, A Brief Guide to Standard Object Modeling Language", Fowler & Scott, 1999.