



Multiplex Online Ticketing System

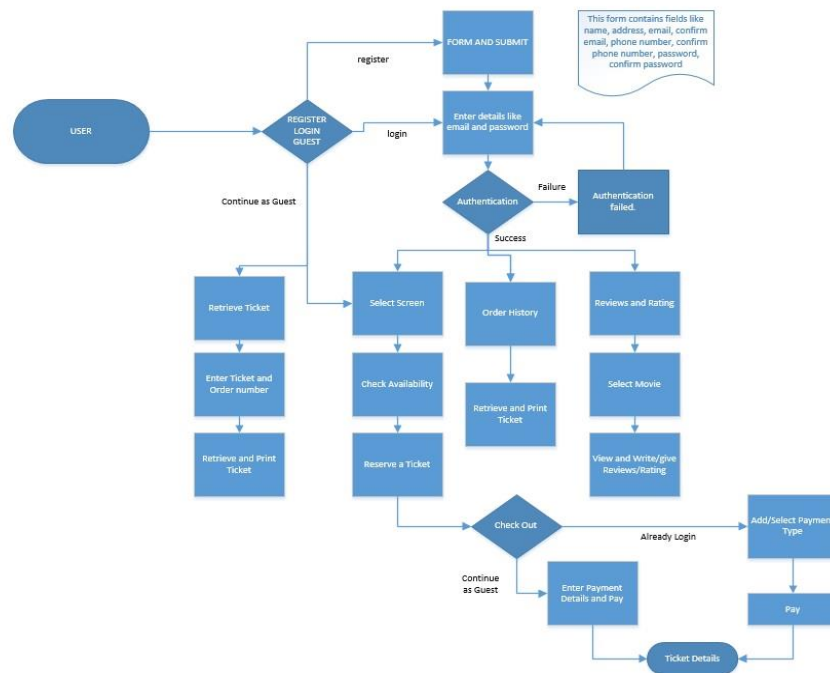
©WEBSPARTANS

Web Spartans

- Perala Laxman
- Varun Reddy
- Bhudev Sahu

Project Report

Architectural diagram of the System



Title: Multiplex online ticketing system for BVV theatre.

Summary:

TicketBook application is intended to facilitate users to book movie tickets that are screened in the multiplex BVV. The multiplex BVV has a fixed number of screens and movies are screened in them at various show times in a day.

Functionalities:

Register for an Account: The user can optionally register for a user account. The user account facilitates the user to login to the system and book tickets online. Registration details include Name, Address, Telephone Number, Email address and login details that will be used by the user to login thereafter.

This functionality also provides the system with the feature of **Authorization/Authentication-protected by username/ password**.

Also here we **replicate** the users' information in another instance to maintain the data redundancy.

Listing all the available movies: A user (registered or guest) can check for the movies that are being screened in the various screens, for the given date. The different show times for a movie, for a given screen, for a given date are then displayed.

Here we assume that most of the users will be booking the tickets for the next day (i.e. May, 2nd). Hence to reduce the load on the server we are **performing distributed caching** using **Memcached** and populating the movies information on the cache and retrieving it from the cache.

Checking Availability: Once the movies and the corresponding show times are listed on user's request, the user can check the ticket availability of a particular show by selecting the screen and show time. The system then provides the user with the number of tickets that are available for the particular show.

Booking a ticket: After the user checks the ticket availability, user can book number of tickets of his/her choice, for a particular show. User selects the number of ticket he/she wants to book. A user may book at most 10 tickets at a time. After the number of seats is selected by the user, the system verifies the availability and proceeds to checkout.

Checkout: The user is presented with the payable amount and is asked to provide credit card details to complete the booking transaction. Once the user provides those details, the payment is processed. After the successful transaction system provides the user with the printable form of the ticket (confirmation). The confirmation includes the reference number, among other details like number of tickets, show time and date.

Retrieve Ticket: This feature facilitates the user to retrieve a ticket that has been booked earlier, at a later time. If the user is logged in, he/she will be able to retrieve the ticket booked by only him/her. If a guest user wants to retrieve the ticket at some later point he/she can get it by providing the reference number. Thus a guest user will not be able to retrieve some other registered users' booked tickets.

The service is **secured for the data transmission** by providing encryption of the communication channel between the client and server using **TLS/SSL encryption mechanism**. The encryption channel has been done by creating a self-signed SSL certificate and deploying the project on **localhost:8443** port, where the communications take place in an encrypted manner.

The **requests** sent from the client side and the **responses** received from the server side are being compressed (using **gzip compression**).

Three services of the system, hosted as web service:

1) Listing all the available movies:

A person may wish to book a movie ticket online hence once he goes to the movie ticket-booking website, a list of movies which are currently playing are displayed. He gets to know about the available movies in the multiplex with their show timings. After deciding on which movie he is going to book the tickets for, he proceeds to reserve the tickets.

2) Making Reservation of Tickets:

Once a person looks after the available movies and their show timings, he makes the reservation of the tickets of his desired show. He selects the timing of the show and checks whether the tickets are available or not and if the tickets are available, he makes the selection of number of tickets he wants to book. Once he is done with this selection the ticket buyer proceeds to checkout.

3) Completion of Ticket Booking:

Once the customer selects the checks out after selecting the show and number of tickets, he is redirected to Payment Section where he would make the payment to purchase the selected amount of tickets. The customer puts the payment details depending upon the payment options which are, by the "Payment Authorization" service, taken to the intended bank or Credit Card Company where the payment gets processed accordingly. This service depends upon the service of third party services such as bank or credit card companies for the payment authentication. After the successful transaction of the payment the customer gets the payment and ticket confirmation message and also an email from the website with the movie show details.

Technology Stack

The application is designed in a client server architectural model. The client's requests are serviced by a set of web services and servlets. The web services and servlets access the Database to retrieve the information requested by the user.

- **Client**
The client module has been developed using JAVA. The view model is rendered using JSPs. The client side validations are done using JAVASCRIPT.
- **Server:**
Web Services
The web services are developed in Representational state transfer (REST) architectural style. The web services have been developed in JAVA Spring framework in tandem with Apache CXF API which provides the required JAX-RS API support.
Servlets also serve some client requests. The servlets are JAVA servlets, developed in the J2EE framework.
The webserver this application has been deployed is Tomcat v7.0.
- **Data Base**

The database used is MySQL 5.5. The connectivity between the server and the database is accomplished using JDBC connection.

- Cache Mechanism

Memcached has been used for the caching functionality that we have implemented

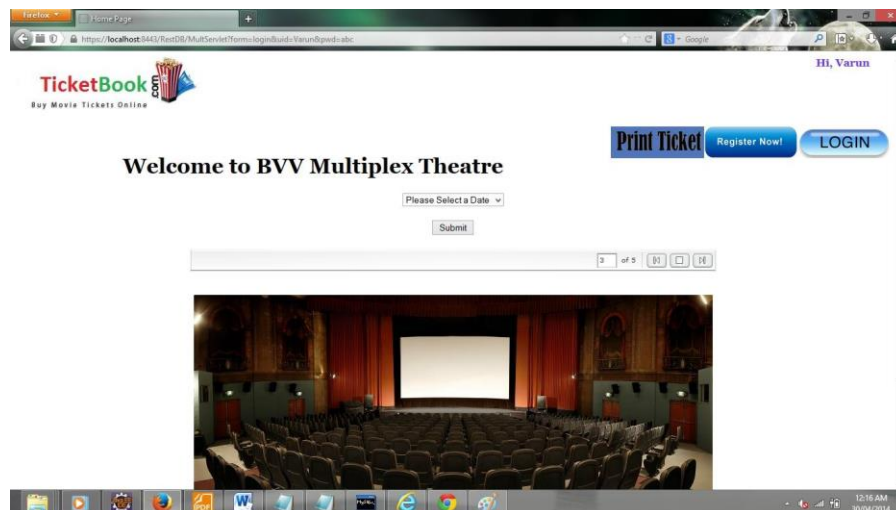
- **SSL Encryption:** The encryption mechanism has been enabled by using self-signed SSL certificate.
- **Compression** The request/response compression is enabled by using gzip content encoding.
 - *We have chosen JAVA technology as the development platform because of its rich API support and compatibility with third party APIs like Memcached.*
 - *MySQL server has been our choice of DB as it is an open source and widely available SQL enabled database.*

Project Flow with UI Specs

This is homepage of our website (TicketBook.com). Here user can register for the new account, login to his active account and can also print the ticket which he reserved previously and can also reserve a new ticket by selecting a date from the list.

The user can perform the following operations in this interface

1. Create an account in the web site.
2. Login using a personal id and password.
3. Print an earlier reserved/ saved ticket by choosing a date in the list



The Login option in home page redirects to this page where a user can enter his (valid) details and view his personal history. If user has no account or does not wants to login, he will be treated as a guest in the system. This information about the user is displayed at the top most right corner as the name of user for logged in users and as guest for others.

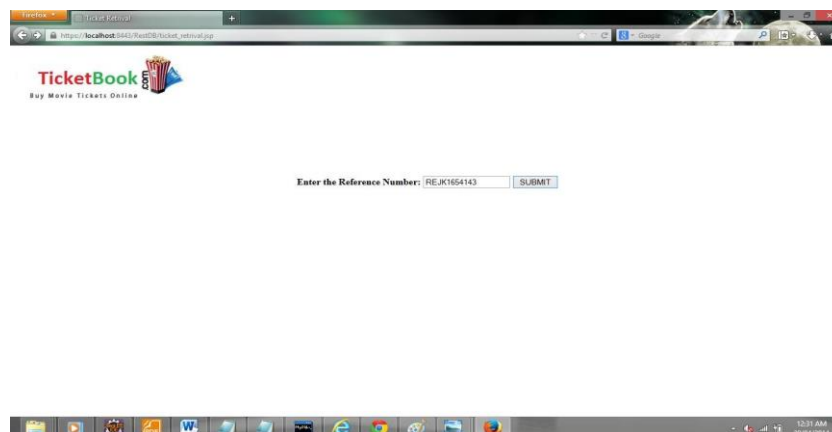


*Register option is the home page will redirect users to this registration page which helps the new users to create an account by providing all the required information (mentioned with * symbol).*

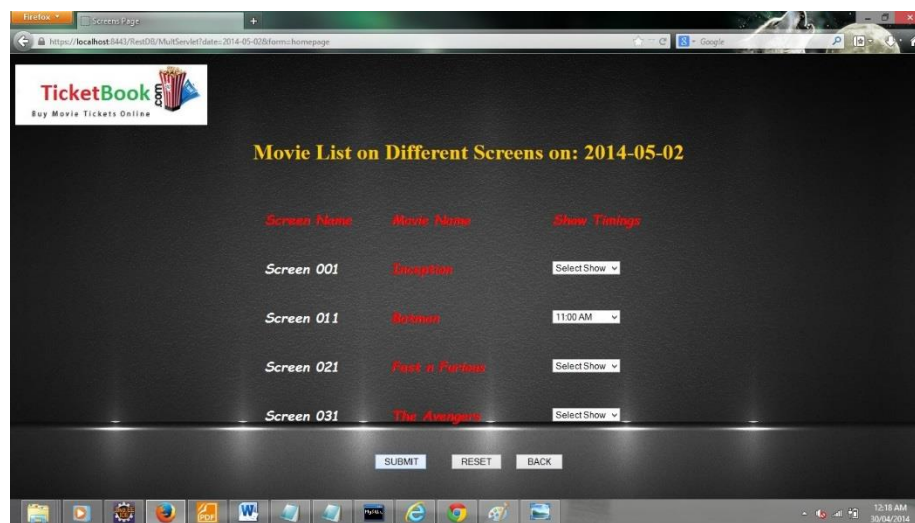


To retrieve a ticket reserved by the user, the following are the steps to be followed:

1. *If the user reserved a ticket from his account, then user have to first login and only then he will be able to retrieve the reserved ticket.*
2. *If the user reserved a ticket as a guest, then to retrieve the ticket user have to provide the reference number issued at the time of reservation.*

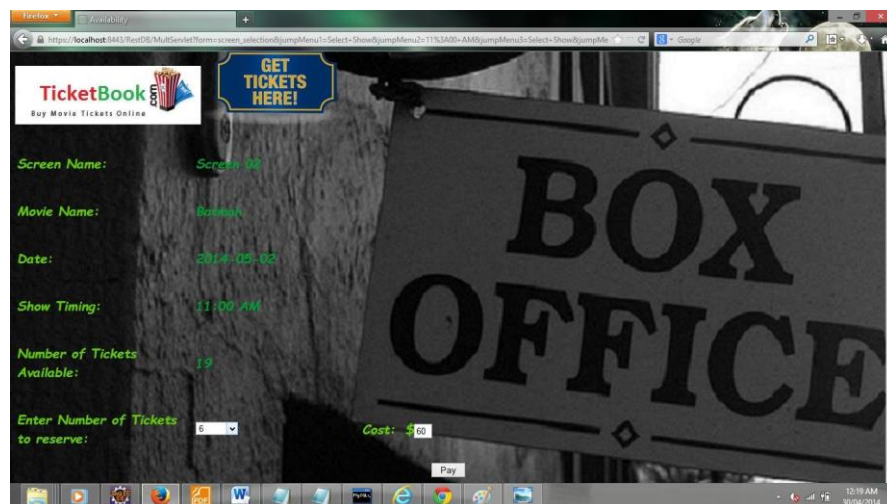


On any day, a user can check the movies that are being played on that day and after choosing one movie and one among the available show timings for that movie, user can check the number of available seats for that particular show.

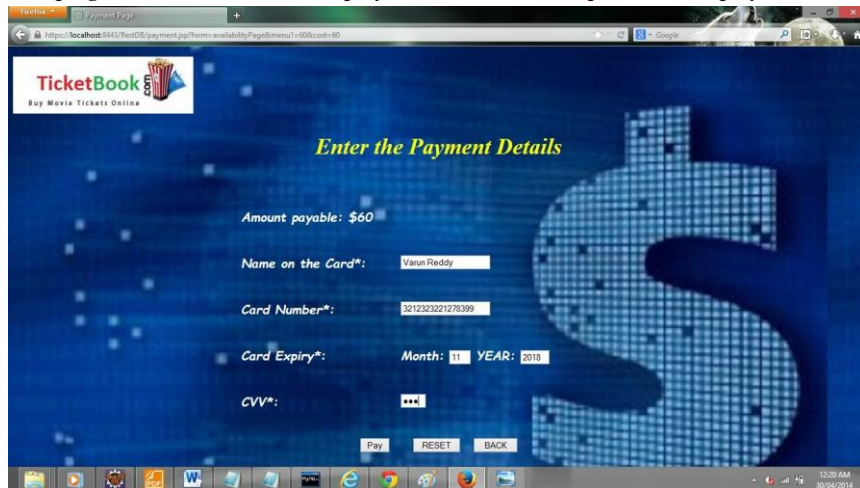


Box Office page allows

- 1. User to check the number of available tickets.*
- 2. User to book the tickets (within a limit of 10).*
- 3. User to make payment to the booked tickets.*



Here in this page he has to enter the payment details and proceed to pay.

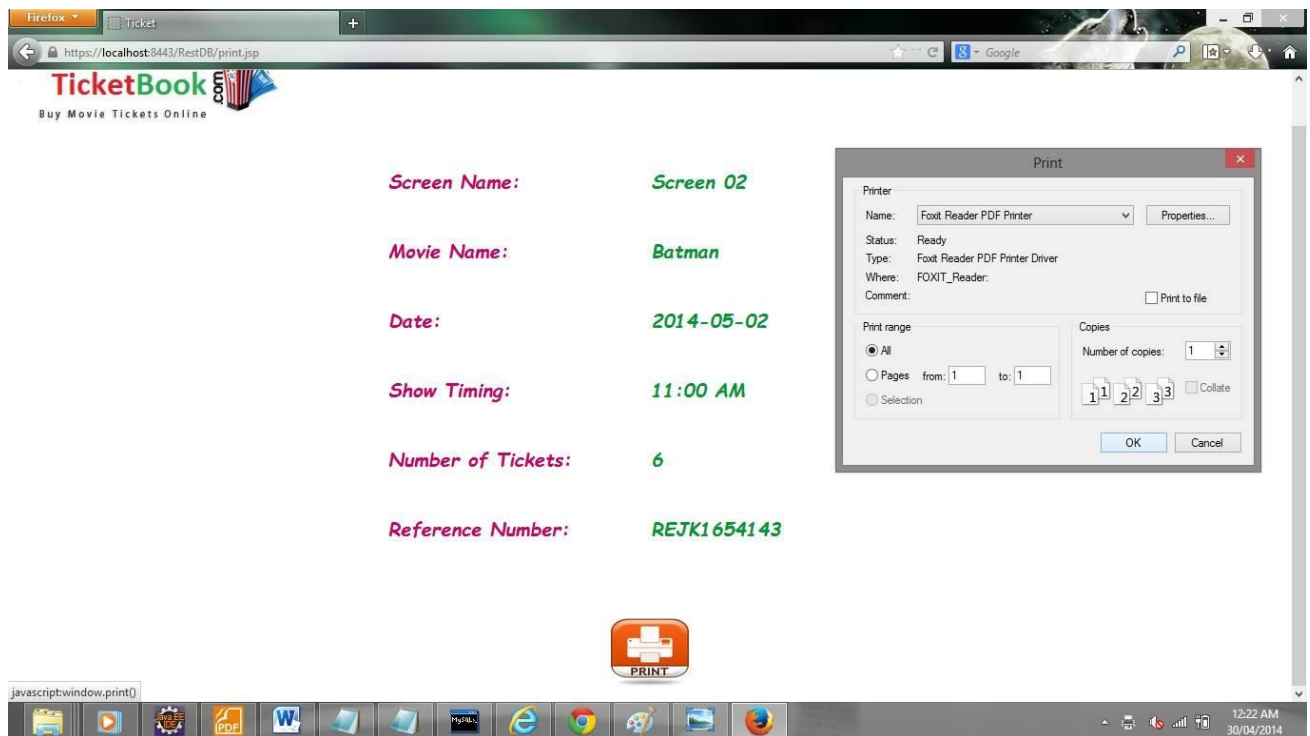


Then the payment will be processed and if payment is successful.



Your Payment is in Succesfull. Please wait.....

Then it shows the ticket with all the details and can take a print out by clicking on print.



Problems encountered and respective resolutions

- Once, the web service could not go past a point where it has to update the DB. After analysis we understood that it is because, prior to that we have flushed all the DB tables with read only restriction.
UNLOCK TABLES is the command to release the locks, in MySQL. The web service functioned properly after that.
- While using SPYCACHED, a Memcached client API, we could not access the required classes of that API. After analysis we understood that the API jars have not been added to the build path. Once the build path has been updated with the required JARs, the problem got resolved.
This particular issue applies to any third party libraries which we need to use in JAVA. The build path should be updated the appropriate JARs.
- While using Memcached, we faced problems persisting a complex array list object, as it kept throwing exception related to serialisation of the object being persisted. We resolved the issue by converting the array list to an array of strings and persisted it in the cache. Later after retrieving the array when required, we have populated the array list with the string array which resolved the issue.
- We constantly faced SQL Exception while trying to directly display information directly using the ResultSet object on a JSP using the `<%= rs.getString(index)%>` format.
This issue has been resolved by storing the result in a String and later using that string for display.

Acknowledgments

- We are immensely thankful to Prof.Mithun Balakrishna, for his support and the resources provided by him in the form of sample REST application and information on how to implement various core features like encryption etc.
- <http://stackoverflow.com/> has been helpful in resolution of various issues.
- <http://javapostsforlearning.blogspot.com/2013/04/create-restful-web-servicesjax-rs-using.html> - This blog had been quite helpful in building a simple REST application.
- <https://code.google.com/p/spymemcached/>
- <http://blog.couchbase.com/memcached-144-windows-32-bit-binary-now-available>