|  |  |
| --- | --- |
|  | **Cognizant Academy**  **Movie Cruiser**  **ASP.NET Specification Document**  **Version 0.1** |
| |  |  |  |  | | --- | --- | --- | --- | |  | **Prepared By / Last Updated By** | **Reviewed By** | **Approved By** | | **Name** | Ramamoorthy Selvam | Vimalathithan Krishnan | Ramadevanahalli Lingachar, Shashidhara Murthy | | **Role** | Learning Solution Designer | Learning Solution Architect | Learning Solution Lead | | **Signature** |  |  |  | | **Date** |  |  |  | |
|  |

Table of Contents

[1.0 Introduction 4](#_Toc11945841)

[1.1 Purpose of this document 4](#_Toc11945842)

[1.2 Definitions & Acronyms 4](#_Toc11945843)

[1.3 Project Overview 4](#_Toc11945844)

[1.4 Scope 4](#_Toc11945845)

[1.5 Intended Audience 4](#_Toc11945846)

[1.6 Hardware and Software Requirment 4](#_Toc11945847)

[2.0 Design for View Movie List Admin (TYUC001) 6](#_Toc11945848)

[2.1 Sequence Diagram 6](#_Toc11945849)

[2.2 ASPX 6](#_Toc11945850)

[3.0 Design for View Movie List Customer (TYUC002) 7](#_Toc11945851)

[3.1 Sequence Diagram 7](#_Toc11945852)

[3.2 ASPX 8](#_Toc11945853)

[4.0 Design for Edit Movie (TYUC003) 9](#_Toc11945854)

[4.1 Sequence Diagram (Show Edit Movie Form) 9](#_Toc11945855)

[4.2 Sequence Diagram (Edit Movie) 10](#_Toc11945856)

[4.3 ASPX (EditMovie.aspx) 10](#_Toc11945857)

[4.1 ASPX (EditMovieStatus.aspx) 11](#_Toc11945858)

[5.0 Design for Add Favorite (TYUC004) 12](#_Toc11945859)

[5.1 Sequence Diagram 12](#_Toc11945860)

[5.1 ASPX (MovieListCustomer.aspx) 13](#_Toc11945861)

[6.0 Design for View Favorite (TYUC005) 14](#_Toc11945862)

[6.1 Sequence Diagram 14](#_Toc11945863)

[6.2 ASPX (Favorite.aspx) 15](#_Toc11945864)

[6.1 ASPX (FavoriteEmpty.aspx) 15](#_Toc11945865)

[7.0 Design for Remove Item from Favorite (TYUC006) 17](#_Toc11945866)

[7.1 Sequence Diagram 17](#_Toc11945867)

[7.2 ASPX (Favorite.ASPX) 18](#_Toc11945868)

[8.0 Standards and Guidelines 18](#_Toc11945869)

[8.1 ASPX 18](#_Toc11945870)

[9.0 Submission 19](#_Toc11945871)

[9.1 Code submission instructions 19](#_Toc11945872)

[10.0 Change Log 19](#_Toc11945873)

# Introduction

## Purpose of this document

The purpose of this document is to define the server side implementation of the movie cruiser application.

## Definitions & Acronyms

|  |  |
| --- | --- |
| Definition / Acronym | Description |
| ASP.NET | Active Server Pages using .NET framework |

## Project Overview

Refer Use Case specification document for understanding the functionality and features.



## Scope

1. Creation of ASP.Net webforms for movie cruiser application

## Intended Audience

* Product Owner
* Scrum Master
* Application Architect
* Project Manager
* Test Manager
* Development Team
* Testing Team

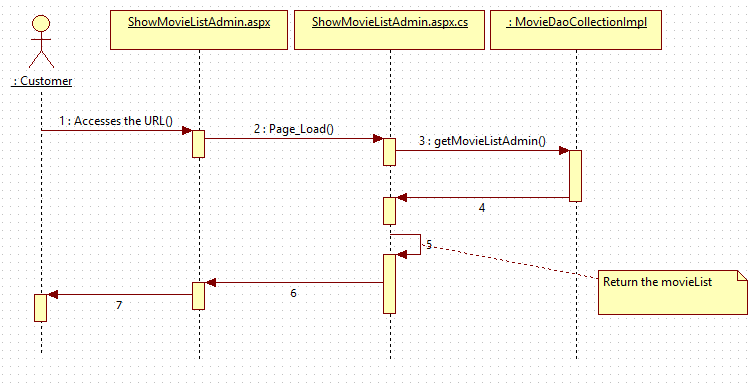
## Hardware and Software Requirment

1. Hardware Requirement:
   1. Developer PC with 4GB Ram
2. Software Requirement
   1. Git
   2. IE or Chrome
   3. .Net Framework 4.5
   4. Visual Studio Professional Edition 2015
   5. SQL Server enterprise edition 2014

# Design for View Movie List Admin (TYUC001)

## Sequence Diagram

The following diagram explains the flow of execution when admin accesses the specific URL. Refer steps included after the sequence diagram.



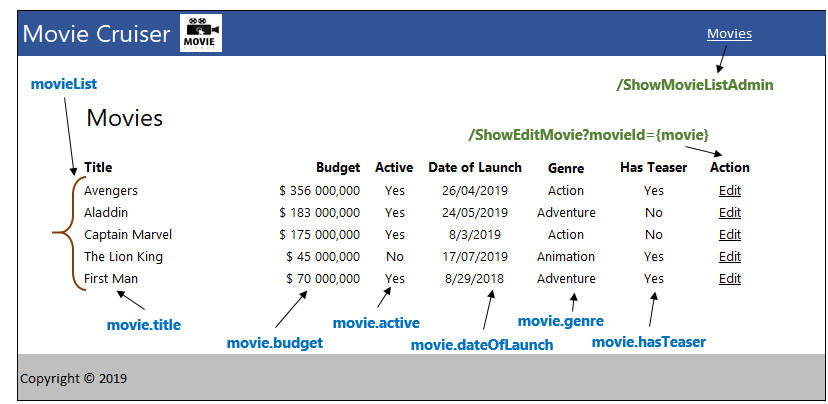
**Steps Explanation**

1. Customer access the ShowMovieListAdmin.aspx
2. Instantiate MovieDaoCollectionImpl class in Page\_Load of ShowMovieListAdmin.aspx.cs and assign it to MovieDao reference variable movieDao.
3. Invoke movieDao.getMovieListAdmin() and get the list of movies.
4. The getMovieListAdmin which holds the movies list for admin will be bound to gridview control ( available in ShowMovieListAdmin.aspx ) using datasource and databind methods.

## ASPX

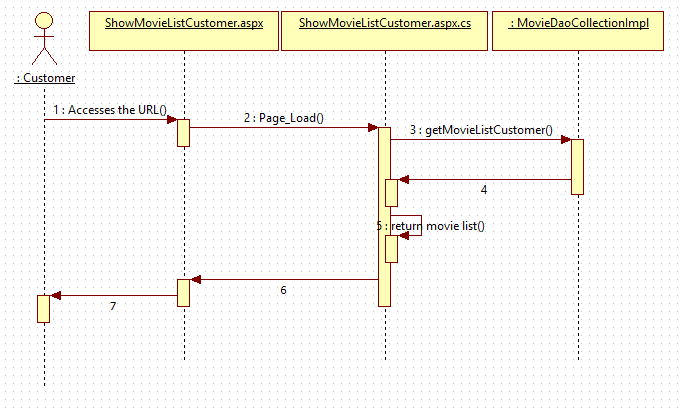
List the movies for admin.

|  |  |  |
| --- | --- | --- |
| **HTML** | **Renamed aspx** | **Description** |
| movie cruiser/MovieListAdminitem.html | WebContent/ MovieListAdminitem.aspx | List the movies for admin |



# Design for View Movie List Customer (TYUC002)

## Sequence Diagram



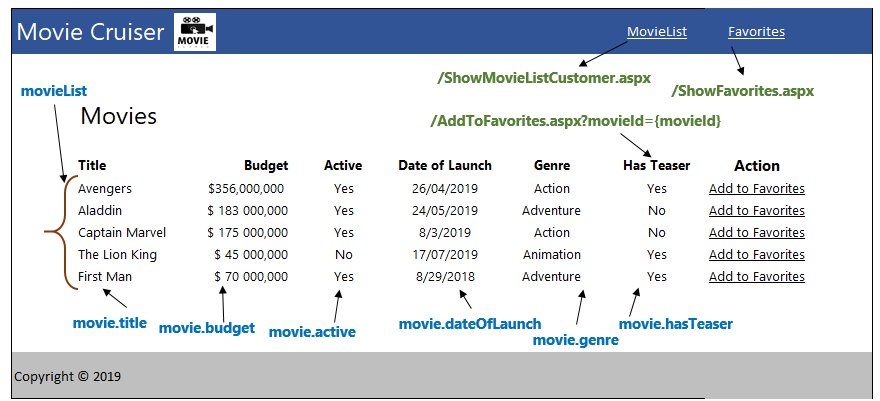
**Steps Explanation**

1. Customer access the ShowMovieListCustomer.aspx
2. Instantiate MovieDaoCollectionImpl class in Page\_Load of ShowMovieListCustomer.aspx.cs and assign it to MovieDao reference variable movieDao.
3. Invoke movieDao.getMovieListCustomer() and get the list of movies.
4. The getMovieListCustomer which holds the movies list for customer will be bound to gridview control ( available in ShowMovieListCustomer.aspx ) using datasource and databind methods.

## ASPX

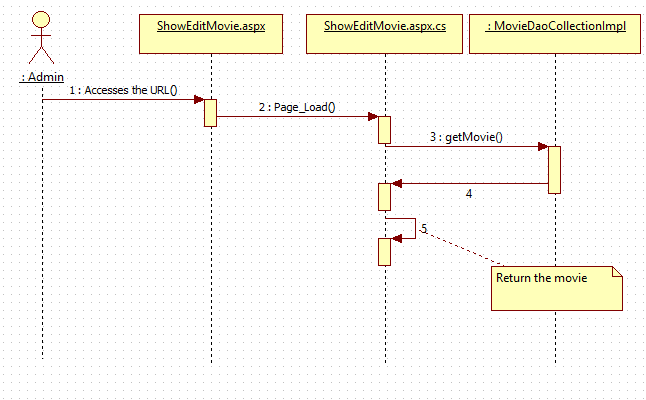
List the movies for customer.

|  |  |  |
| --- | --- | --- |
| **HTML** | **Renamed ASPX** | **Description** |
| movie cruiser/MovieListCustomer.html | movie cruiser/MovieListCustomer.aspx | List the movies for customer |



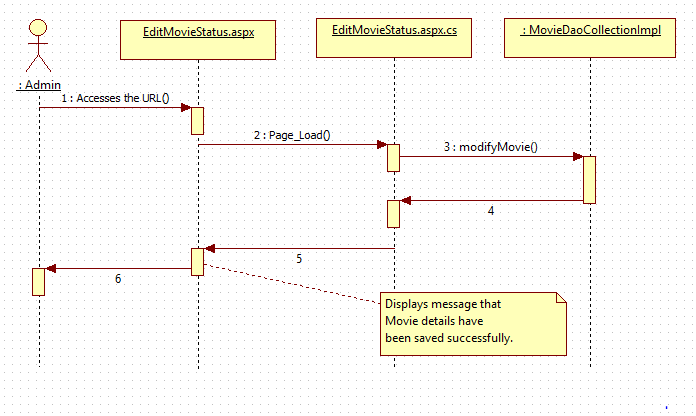
# Design for Edit Movie (TYUC003)

## Sequence Diagram (Show Edit Movie Form)

**Steps Explanation**

1. Customer clicks on Edit link on a movie.
2. Customer access the ShowEditMovie.aspx
3. The request is receieved in the Page\_load method of ShowEditMovie.aspx.cs.
4. Instantiate MovieDaoCollectionImpl class in Page\_Load of ShowEditMovie.aspx.cs and assign it to MovieDao reference variable movieDao.
5. The movieID that is edited by the user is passed as a input parameter to getMovie() method. User is naviaged to ShowEditMovie.aspx page which populates the movie details in the respective form fields for the passed movieId.

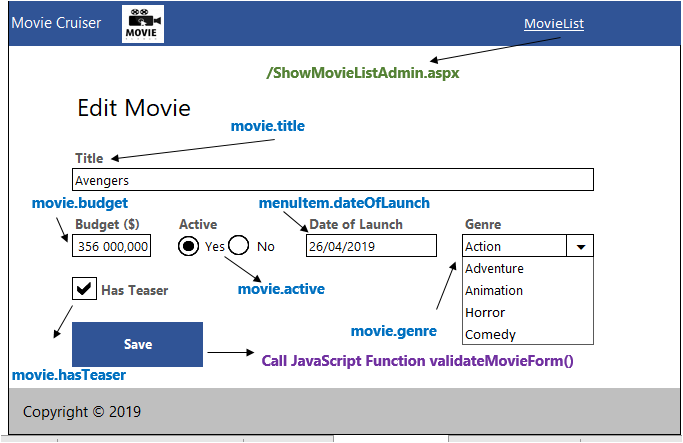
## Sequence Diagram (Edit Movie)



Explanation steps are not included for the above diagram as well as subsequent sequence diagrams. Earlier explanation steps can be referred to arrive at the understanding.

## ASPX (EditMovie.aspx)

|  |  |  |
| --- | --- | --- |
| **HTML** | **Renamed ASPX** | **Description** |
| movie cruiser/EditMovie.html | movie cruiser/EditMovie.aspx | Page with form to edit a movie |

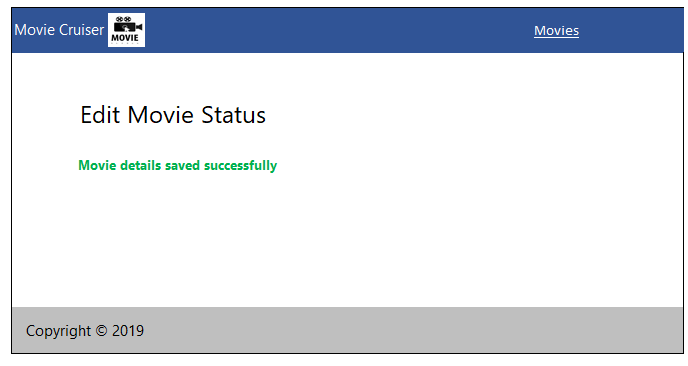


**Category Dropdown Handling**

1. Set the value from movie.category as the first item in the drop down.
2. Let the other items be added directly in the aspx with hard coded values.

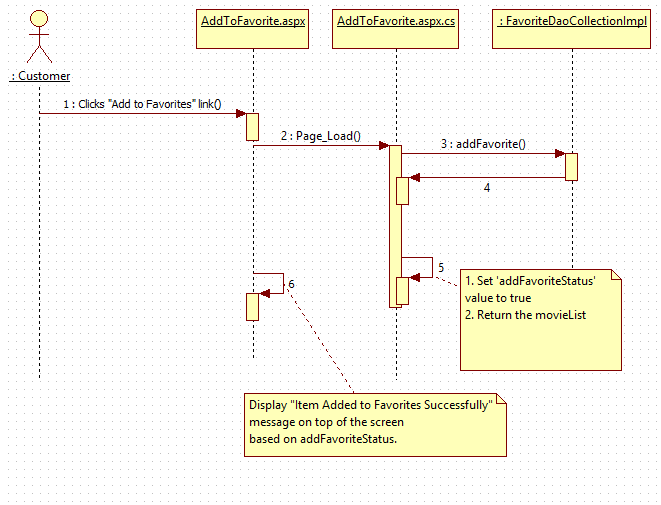
## ASPX (EditMovieStatus.aspx)

|  |  |  |
| --- | --- | --- |
| **HTML** | **Renamed ASPX** | **Description** |
| movie cruiser/EditMovieStatus.html | movie cruiser/EditMovieStatus.aspx | Display the status of modifying movie. |



# Design for Add Favorite (TYUC004)

## Sequence Diagram



**Steps Explanation**

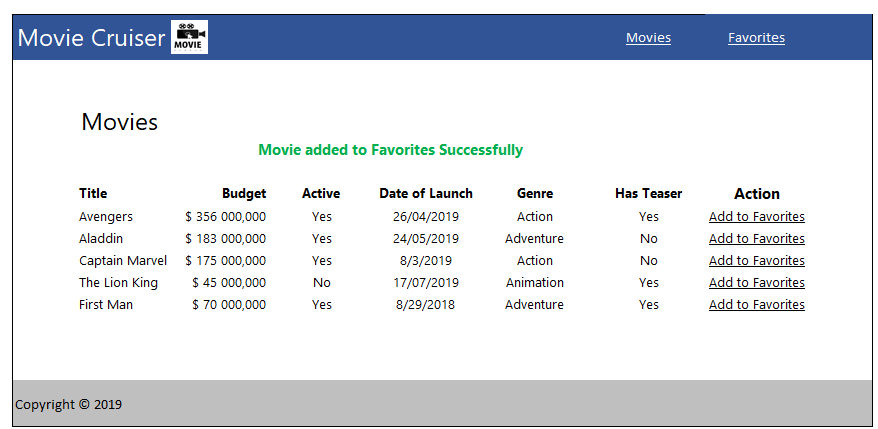
1. Customer clicks on “Add to Favorite” link on a movie from Movie List.
2. Customer is directed to AddToFavorite.aspx?movieId=1 URL
3. Set variable userId of type long to value 1. Identifier for a specific user.
4. Read the movieId from the request and assign it to a variable
5. Instantiate FavoriteDaoCollectionImpl and assign it to FavoriteDao.
6. Invoke the addFavoriteItem() method passing the userId and movieId.
7. Instantiate MovieDaoCollectionImpl class in Page\_Load of AddToFavorite.aspx.cs and assign it to MovieDao reference variable movieDao.
8. List of Movie returned is stored in a variable is bound to gridview control

( available in AddToFavorite.aspx ) using datasource and databind methods

1. The addFavoriteStatus with value true is stored in a variable and displayed on the page.

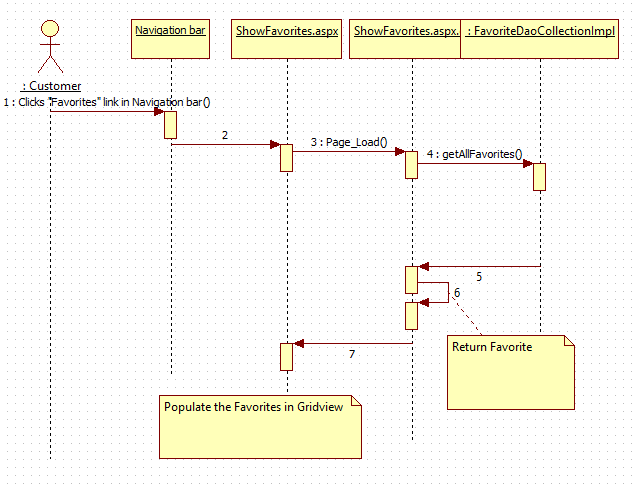
## ASPX (MovieListCustomer.aspx)

This aspx is the same the file defined in View Movie List Customer section.



# Design for View Favorite (TYUC005)

## Sequence Diagram



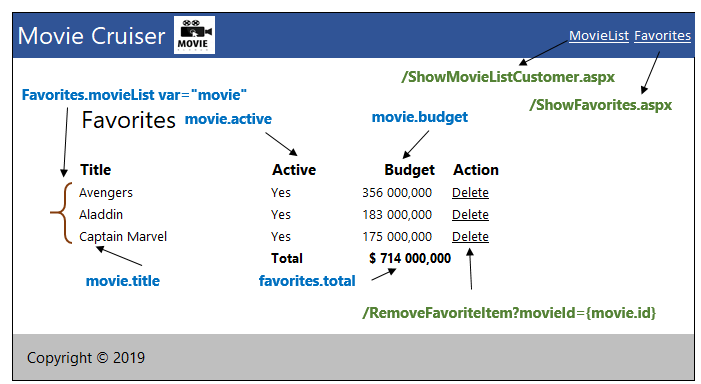
**NOTE:** If getAllFavoriteItems() method throws an exception, then the exception is passed on to FavoriteEmpty.aspx page.

**Steps Explanation**

1. Customer clicks on “Favorite” link on the navigation bar.
2. Customer access the /ShowFavorite.aspx
3. The request is received in Page\_Load method of ShowFavorite.aspx
4. Set variable userId with type long having value 1
5. Instantiate FavoriteDaoCollectionImpl and assign it to FavoriteDao.
6. Invoke the getAllFavoriteItems() method passing the userId.
7. If the getAllFavoriteItems() does not throw FavoriteEmptyException:
8. List of Movie returned is bound to gridview control ( available in ShowFavorite.aspx ) using datasource and databind methods
   1. Invoke the getAllFavoriteItems() method passing the userId.
   2. If the getAllFavoriteItems() does not throw FavoriteEmptyException:
9. If the getAllFavoriteItems() throws FavoriteEmptyException:
   1. Catch CatchEmptyException and redirect the user to FavoriteEmpty.aspx

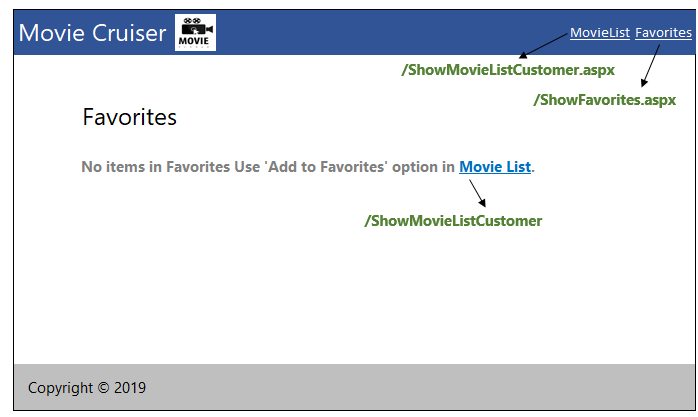
## ASPX (Favorite.aspx)

|  |  |  |
| --- | --- | --- |
| **HTML** | **Renamed ASPX** | **Description** |
| movie cruiser/Favorite.html | movie cruiser/Favorite.aspx | Displays the favorite with the list of movies added by the customer |



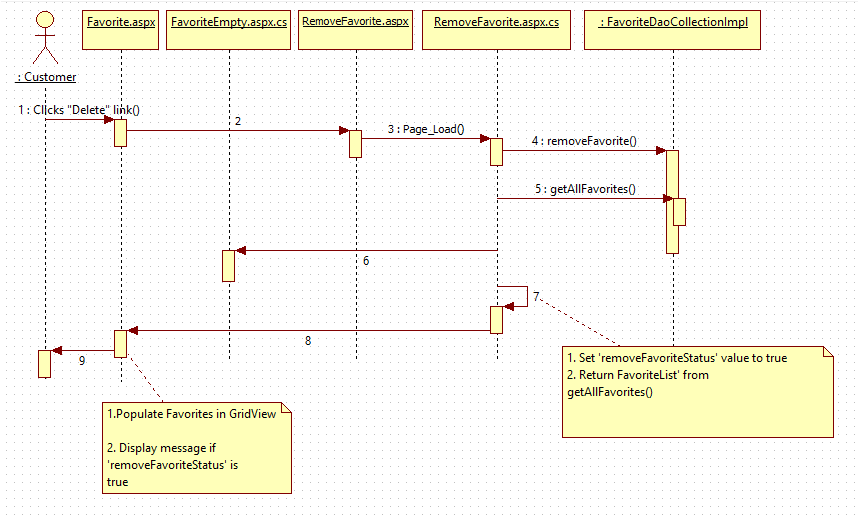
## ASPX (FavoriteEmpty.aspx)

|  |  |  |
| --- | --- | --- |
| **HTML** | **Renamed ASPX** | **Description** |
| movie cruiser/FavoriteEmpty.html | /FavoriteEmpty.aspx | Displays the favorite empty message |



# Design for Remove Item from Favorite (TYUC006)

## Sequence Diagram



**Steps Explanation**

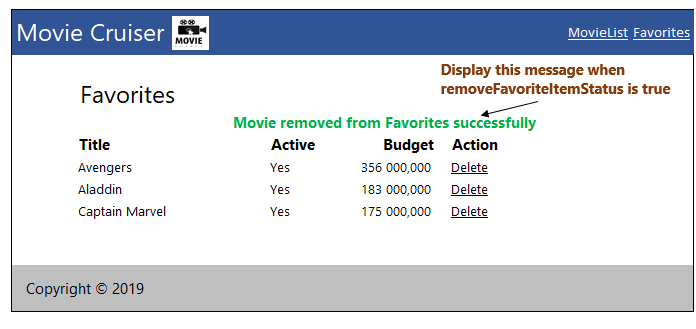
1. Customer clicks on “Remove” link on movie in Favorite.aspx.
2. Customer access the /RemoveFavorite.aspx?movieId=1 URL.
3. The request is received in Page\_Load() method
4. Read the movieId from the request and assign it to a variable
5. Instantiate FavoriteDaoCollectionImpl and assign it to FavoriteDao.
6. Set variable userId of type long having value 1.
7. Invoke removeFavoriteItem() method passing the userId and movieId.
8. Set the request with variable removeFavoriteItemStatus having value true.
9. Invoke the getAllFavoriteItems() method passing the userId.
10. If the getAllFavoriteItems() does not throw FavoriteEmptyException:
11. List of Movie returned is bound to gridview control ( available in RemoveFavorite.aspx ) using datasource and databind methods

Eg: < To be updated>

* 1. Invoke the getAllFavoriteItems() method passing the userId.
  2. If the getAllFavoriteItems() does not throw FavoriteEmptyException:

1. If the getAllFavoriteItems() throws FavoriteEmptyException:
   1. Catch CatchEmptyException and redirect the user to FavoriteEmpty.aspx

## ASPX (Favorite.ASPX)



For other field values refer MovieListCustomer.aspx specification in View Movie List Customer.

# Standards and Guidelines

## ASPX

1. Every server control should have meaningful Id
2. Server control naming convention to be followed in Id, to explain what type of control is used. Eg: txtUserName
3. Startup page should be set for the web application
4. Startup page should be set for the web application
5. No hard coded values in aspx/ascx files. To be referenced from Web.config file
6. Database connection string should be set in the ConnectionStrings section of Web.config and NOT in the AppSettings

# Submission

## Code submission instructions

Once your code is evaluated by the trainer and all the issues reported by the trainer are corrected, the code needs to be submitted to the remote repository. Follow the steps below to submit the code to remote repository.

1. In Windows Explorer go to the movie cruiser folder
2. Right click on the empty space in the right hand side of Windows Explorer and select “Git Bash here”
3. Execute the following commands

To display the added or modified files

git status

To stage the added or modified files

git add .

To display the staged files

git status

To save the code to local repository

git commit -m "Servlet & JSP"

To transfer the changes from local machine to server

git push origin master

1. Successful execution of the above commands will upload the files to the server repository.
2. Login into <https://code.cognizant.com>
3. Click on the project movie cruiser
4. Check if the files that are uploaded correctly with appropriate folder structure.

# Change Log

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Changes Made | | | |
| V1.0.0 | Initial baseline created on 25-May-19 by Ramamoorthy Selvam | | | |
| Vx.y.z | <Please refer the configuration control tool / change item status form if the details of changes are maintained separately. If not, the template given below needs to be followed> | | | |
| **Section No.** | **Changed By** | **Effective Date** | **Changes Effected** |
|  |  |  |  |