|  |  |
| --- | --- |
|  | **Cognizant Academy**  **Movie Cruiser**  **Web Interface Specification Document**  **Version 1.0** |
| |  |  |  |  | | --- | --- | --- | --- | |  | **Prepared By / Last Updated By** | **Reviewed By** | **Approved By** | | **Name** | Chandrasekaran Janardhanan | Vimalathithan Krishnan | Ramadevanahalli Lingachar, Shashidhara Murthy | | **Role** | Learning Solution Designer | Learning Solution Architect | Learning Solution Lead | | **Signature** |  |  |  | | **Date** | 20 May 2019 | 21 May 2019 | 17 Jun 2019 | |
|  |

Table of Contents

[1.0 Introduction 3](#_Toc12395241)

[1.1 Purpose of this document 3](#_Toc12395242)

[1.2 Definitions & Acronyms 3](#_Toc12395243)

[1.3 Project Overview 3](#_Toc12395244)

[1.4 Scope 3](#_Toc12395245)

[1.5 Intended Audience 3](#_Toc12395246)

[1.6 Hardware and Software Requirement 3](#_Toc12395247)

[2.0 Git Project Cloning 4](#_Toc12395248)

[3.0 Screen Flow 6](#_Toc12395249)

[4.0 View Movie List (MCUC001 & MCUC002) 7](#_Toc12395250)

[4.1 Movie List Admin 7](#_Toc12395251)

[4.2 Movie List Customer 8](#_Toc12395252)

[5.0 Edit Movie (MCUC003) 9](#_Toc12395253)

[5.1 Edit Movie 9](#_Toc12395254)

[5.2 Edit Movie Status 10](#_Toc12395255)

[6.0 Add to Favorites (MCUC004) 11](#_Toc12395256)

[6.1 Movie List Customer (Add to Favorites success) 11](#_Toc12395257)

[7.0 View Favorites (MCUC005) 12](#_Toc12395258)

[7.1 Favorites 12](#_Toc12395259)

[7.1 Favorites Empty 13](#_Toc12395260)

[8.0 Remove Movie from Favorites (MCUC005) 14](#_Toc12395261)

[8.1 Remove Favorites 14](#_Toc12395262)

[9.0 Standards and Guidelines 14](#_Toc12395263)

[9.1 HTML 14](#_Toc12395264)

[9.2 CSS 15](#_Toc12395265)

[9.3 JavaScript 15](#_Toc12395266)

[10.0 Submission 16](#_Toc12395267)

[10.1 Code submission instructions 16](#_Toc12395268)

[11.0 Change Log 17](#_Toc12395269)

# Introduction

## Purpose of this document

The purpose of this document is to define the user interface specification for Movie Cruiser project.

## Definitions & Acronyms

|  |  |
| --- | --- |
| Definition / Acronym | Description |
| HTML | Hyper Text Markup Language |
| CSS | Cascading Style Sheet |

## Project Overview

Refer movie-cruiser-use-case-specification.docx for understanding the functionality and features.

## Scope

Develop static HTML web pages for Movie Cruiser application.

## Intended Audience

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

## Hardware and Software Requirement

1. Hardware Requirement:
   1. Developer Desktop PC with 4GB RAM
2. Software Requirement
   1. Notepad ++ for editing HTML and CSS
   2. Chrome Browser

# Git Project Cloning

Code for Movie Cruiser project should be submitted to a Git server repository. This will enable the Facilitator to validate your code. The following paragraphs will give an overview about Git and the steps required to setup the folder structure before starting to code.

There is an existing project in GitLab server that contains the folder structure required for developing this application. Our first step is a create a copy of that project in your GitLab profile. Before starting the cloning steps, let us understand about GitLab.

**What is GitLab?**

GitLab is a software that can be installed in a centralized server to manage code from various developers. For the purpose of learning and collaboration within employees of Cognizant, a separate GitLab instance is available for use. This is hosted in the URL <https://code.cognizant.com>. Each employee in Cognizant has access to this repository and can login into this portal using Cognizant network credentials. Each employee can create upto 10 projects under his/her profile. The code for Movie Cruiser has to be submitted in this server.

**How Movie Cruiser project is organized in GitLab?**

There is an existing project in GitLab that contains the necessary folder structure for Movie Cruiser project. You need to create a copy of this project in your profile before downloading the code from the server. The process of creating a copy of this project in your profile is called forking. Find the instructions befow to fork the project:

**Steps for Forking**

1. In browser, login with you network credentials at <https://code.cognizant.com>
2. Copy and paste the below URL in the browser after login.

<https://code.cognizant.com/genc-java/final-check>

1. Click fork
2. Click on your name
3. The above steps creates a new project in your profile.
4. If you click on GitLab link on the top left corner, you can find the project listed under your name in the “Your Projects” section. This means that you have successfully completed forking the project.
5. Click on the project and go through the folders and files in the project

**Steps for Cloning**

The steps done earlier had created copy of the project under your profile. Now, this project is only available in the server. We need to get a copy of this project in your local machine, so that you can start writing your code in the folder created locally. This process of creating a copy from the server to local PC is called cloning. The steps below will guide to clone the project.

1. Let us download the code from the server to D:
2. Open Windows Explorer and click on D:
3. Right click on the empty space in right hand side of the explorer window and select “Git Bash here” option
4. Execute the following commands. Replace content within angular brackets <> with appropriate values:

**git config --global user.name "<LastName>, <FirstName> (Cognizant)"**

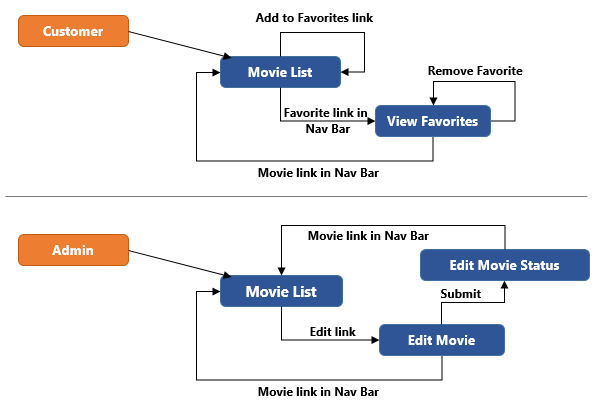
**git config --global user.email "<CognizantEmailId>"**

**git clone** [**https://code.cognizant.com/<EmployeeId>/final-check.git**](https://code.cognizant.com/%3cEmployeeId%3e/final-check.git)

1. After completion of above steps you can find a new folder ‘movie-cruiser’ created in D:, which will contain few folders and files. The folder structure should be similar to the folders and files in the server.
2. Once the above steps are completed, you are ready for writing the code. Place the files as per the instructions provided in the following sections.

# Screen Flow

Flow of screens for Customer and Admin provided below. Blue boxes represent pages. Orange boxes represent the user role. The label in the lines denote the link in the page used to arrive from the specific page.

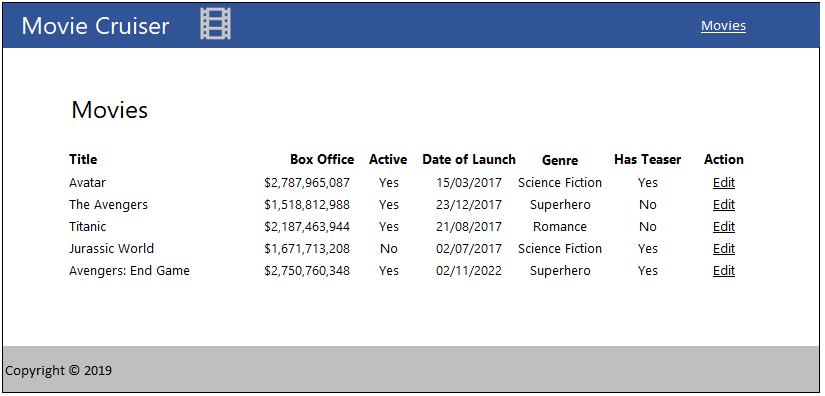


# View Movie List (MCUC001 & MCUC002)

## Movie List Admin

Displays the list of all movies for Admin. Each movie will have option to edit a particular movie

File: movie-cruiser/WebContent/movie-list-admin.html



**Field Specification**

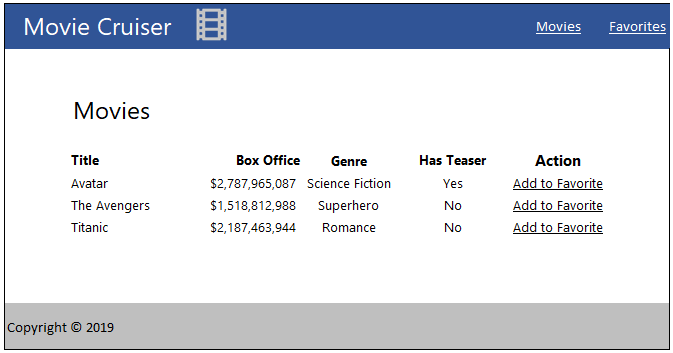
The movies are listed in a table kind of structure. Find below the specification for each column in the Movie List table.

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Field Type** | **Description** |
| Title | Label | Displays the title of the movie |
| Box Office | Label | Display the box office collection of the movie in international currency format. Should include the currency symbol as well. |
| Active | Label | Display ‘Yes’ or ‘No’ based on whether the movie is active or not |
| Date of Launch | Label | Display date of launch in DD/MM/YYYY format |
| Genre | Label | Displays the genre of the movie |
| Has Teaser | Label | Display ‘Yes’ or ‘No’ based on the respective movie teaser status |
| Action | Link | Display ‘Edit’ as a link that points to edit-movie.html |
| Movies | Link | Displays movie-list-admin.html |

## Movie List Customer

Displays the list of movies for customer. Each movie will have option to add the specific movie to favorites

File: movie-cruiser/WebContent/movie-list-customer.html



**Field Specification**

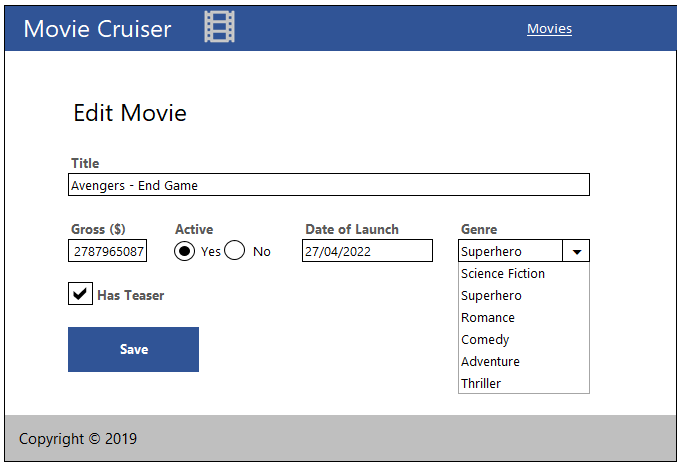
|  |  |  |
| --- | --- | --- |
| **Field Name** | **Field Type** | **Description** |
| Title | Label | Displays the title of the movie |
| Box Office | Label | Display the box office collection of the movie in international currency format. Should include the currency symbol as well. |
| Genre | Label | Displays the genre of the movie |
| Has Teaser | Label | Display ‘Yes’ or ‘No’ based on the respective movie teaser status |
| Action | Link | Display ‘Add to Favorites’ link, clicking this link will display the same page with notification message that the Movie is added into the favorites or not |
| Movies | Link | Displays movie-list-customer.html |
| Favorites | Link | Displays favorites.html |

# Edit Movie (MCUC003)

## Edit Movie

Screen with form fields to edit the movie details. This page is arrived by clicking the Edit link in movie-list-admin.html.

File: movie-cruiser/WebContent/edit-movie.html



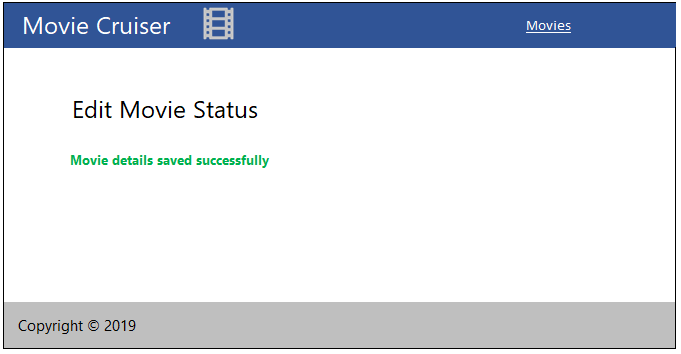
**Field Specification**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Field Type** | **Name in HTML** | **Validation Message** |
| Title | Text Box | title | * Title is required. * Title should have 2 to 100 characters. |
| Box Office | Text Box | boxOffice | * Box Office is required. * Box Office has to be a number. |
| Date of Launch | Text Box | dateOfLaunch | * Date of Launch is required |
| Active | Radio Buttons | active |  |
| Genre | Drop Down List | genre | * Select one genre |
| Has Teaser | Checkbox | hasTeaser |  |
| Save | Submit | submit | Button used by user to submit the form for saving a Movie. Above mentioned validations needs to be done on click of this button. The message needs to be displayed using JavaScript alert. |
| Movies | Link |  | Displays movie-list-admin.html |

## Edit Movie Status

Display the status of submission of edit movie form. This page is arrived after clicking save in edit-movie.html.

File: movie-cruiser/WebContent/edit-movie-status.html



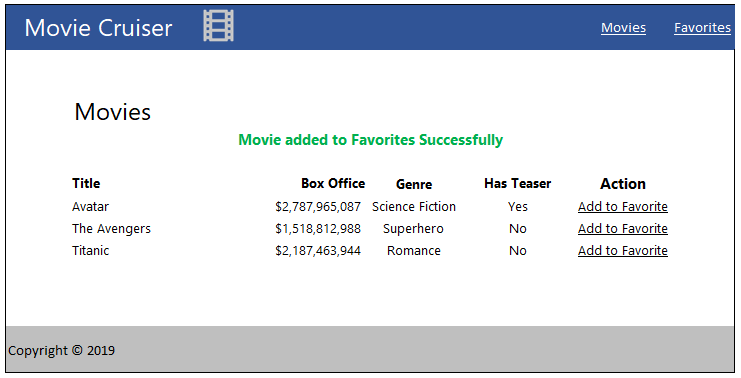
|  |  |  |
| --- | --- | --- |
| **Field Name** | **Field Type** | **Description** |
| Movies | Link | Displays movie-list-admin.html |

# Add to Favorites (MCUC004)

## Movie List Customer (Add to Favorites success)

Customer can add a movie to Favorites by clicking “Add to Favorites” link in movie-list-customer.html. On clicking this link a page similar to Movie List page is displayed with the message that the Movie has been added to the favorites successfully.

File: movie-cruiser/WebContent/movie-list-customer-notification.html



**Field Specification**

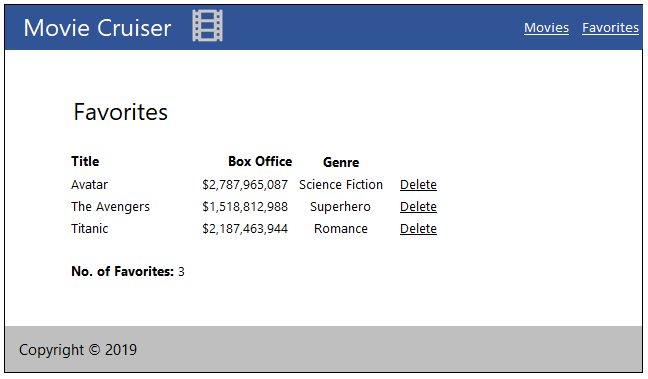
Refer Movie List Customer Section

# View Favorites (MCUC005)

## Favorites

Displays the list of movies added to the favorites. This screen is arrived based on the navigation link in the top of the screen from movie-list-customer.html.

File: movie-cruiser/WebContent/favorites.html



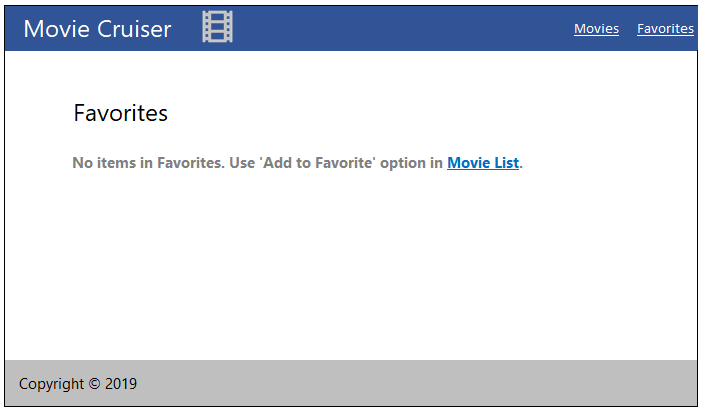
**Field Specification**

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Field Type** | **Description** |
| Title | Label | Displays the title of the movie |
| Has Teaser | Label | Display ‘Yes’ or ‘No’ based the respective movie delivery status |
| Box Office | Label | Display the box office of the movie in currency format with two decimal places. Should include the currency symbol as well. |
| Delete | Link | Display ‘Delete’ link, which displays the same page with notification message that the movie is deleted from the Favorites successfully. |
| No. of Favorites | Label | Displays the number of favorites that the customer had added |
| Movies | Link | Displays movie-list-customer.html |
| Favorites | Link | Displays favorites.html |

## Favorites Empty

This page will be used when there are no movies in the favorites. The Movie List link in the message should point to movie-list-customer.html.

File: movie-cruiser/WebContent/favorites-empty.html



**Field Specification**

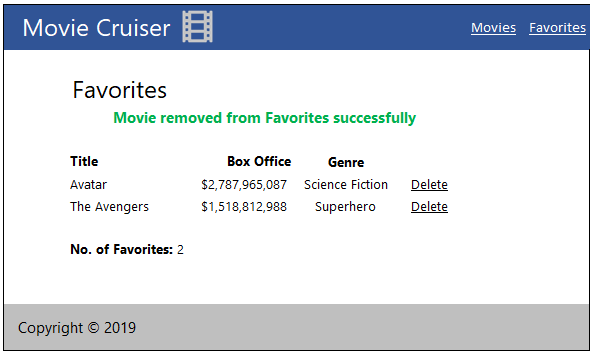
|  |  |  |
| --- | --- | --- |
| **Field Name** | **Field Type** | **Description** |
| Movies | Link | Displays movie-list-customer.html |
| Favorites | Link | Displays favorites.html |

# Remove Movie from Favorites (MCUC005)

## Remove Favorites

When Customer click ‘Delete’ link in favorites.html, this page is displayed with the message that the movie had been removed from the Favorites successfully.

File: movie-cruiser/WebContent/favorites-notification.html



**Field Specification**

Refer section View Favorites

# Standards and Guidelines

## HTML

1. Place all HTML files under WebContent folder.
2. HTML5 semantic tags has to be used wherever applicable. Avoid using div tags for header, footer, article and navigation bar.
3. Document Type should be defined as

<!DOCTYPE html>

1. Form fields must always include a <label> element with a "for" attribute matching the "id".

<label for="field-email">email</label>

<input type="email" id="field-email" name="email" value="" />

1. Each form field should have "name" attribute.
2. All tag names and attributes must be written in lowercase.
3. Inline style should be never used. Style should be in an external CSS and should be linked to the document.
4. Image files should be placed in WebContent/images folder.
5. Add one more tab on the next line after opening a tag
6. Reduce one tab on line when tag is closed

## CSS

1. CSS files names should be style.css
2. CSS files should be placed in WebContent/style folder.
3. Feel free to change the Background Color, Font, Font Size and Font Color of the screen. But it should be applied consistently across screens using CSS. Following are the element in the screen that can be targeted for style change:
   1. Navigation Bar
   2. Page Heading
   3. Form Elements
   4. Messages
4. Reuse styles wherever applicable. Potential areas are Application Name, Page Title, Form Fields and Buttons
5. Selector names should be in lower case with words separated by hyphen.

.page-title {

margin: 20px;

}

1. Selector class names should be relevant to the purpose of the element where the style is applied. For example if common style needs to be applied for all text boxes in an application.
2. Properties should be followed by a colon and a space
3. Properties and values should be in lower case
4. Use hexadecimal code for color definition

Example:

.login-form {

background: #fdfdfd;

display: block;

margin: 0;

margin-left: 20px;

}

## JavaScript

1. JavaScript file name should **script.js**
2. JavaScript file should be placed in folder **WebContent/js**
3. Use camelCase for variables and functions.

var firstName = "John";

function alert() {

alert("This is an alert message.");

}

1. All variable names start with a letter.
2. Always put a space before and after an operator.

var profit = sellingPrice - buyingPrice;

1. Always end a statement with semicolon.
2. Open curly braces should be in the same line.
3. All lines following open curly braces should have one level of indent
4. Closing curly braces should be in next line with reduction of indent by one level

Example

if (time < 20) {

greeting = "Good day";

} else {

greeting = "Good evening";

}

# 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 "web module"

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 <dd-Mon-yy> by <Name of Author> | | | |
| 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** |
|  |  |  |  |