HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

**SCHOOL OF ELECTRICAL AND ELECTRONIC ENGINEERING**



**PROJECT REPORT**

**MOVIE GALERY**

Name of students: Class code: 152564*–* Team 3

Nguyen Tien Dat – 20233836

Truong Cong Duc – 20233840

Bui Tuan Minh – 20233866

Name of instructors: Dr. Pham Van Tien

Assoc. Prof. Tran Thi Thanh Hai

Hanoi, 12/2024

Table of Contents

[LIST OF TABLES 6](#_Toc186750158)

[ACKNOWLEDGEMENT 7](#_Toc186750159)

[ABSTRACT 8](#_Toc186750160)

[1. Introduction 9](#_Toc186750161)

[1.1. Motivation 9](#_Toc186750162)

[1.2. Objectives 9](#_Toc186750163)

[Main objectives 9](#_Toc186750164)

[Specific objectives 9](#_Toc186750165)

[2. Methodology 10](#_Toc186750166)

[2.1. State of the art 10](#_Toc186750167)

[Methods for Movie Gallery 10](#_Toc186750168)

[Existing products for movie gallery 11](#_Toc186750169)

[Discussion 12](#_Toc186750170)

[2.2. Application of the 9 steps in engineering design process 12](#_Toc186750171)

[3. Project implementation 12](#_Toc186750172)

[3.1. Step 1: User requirement 12](#_Toc186750173)

[3.2. Step 2: Specifications 13](#_Toc186750174)

[Functionality 13](#_Toc186750175)

[Non functionality 13](#_Toc186750176)

[3.3. Step 3: Planning 14](#_Toc186750177)

[3.4. Step 4: Block Design 15](#_Toc186750178)

[3.5. Step 5: Detail block design 17](#_Toc186750179)

[3.5.1. Activity Diagram 18](#_Toc186750180)

[3.5.2. Sequence Diagram 26](#_Toc186750181)

[3.6. Step 6: Best selection 33](#_Toc186750182)

[3.7. Step 7: Prototyping 35](#_Toc186750183)

[3.8. Step 8: Testing 40](#_Toc186750184)

[4. Discussions 43](#_Toc186750185)

[5. Conclusions and future works 43](#_Toc186750186)

[6. Reference 44](#_Toc186750187)

[APPENDIX – TEST RESULT 45](#_Toc186750188)

**LIST OF FIGURES**

[Figure 2.2.1: Design Process](#_Toc186178727) 12

[Figure 3.3.1: Planning of the project](#_Toc186178728) 14

Figure 3.4.1: System diagram 15

Figure 3.4.2: Functional diagram 16

Figure 3.4.3: Use Case Diagram 17

Figure 3.5.1: Activity Diagram for (a) Sign Up and (b) Sign In 18

Figure 3.5.2: Activity Diagram for Add Contents 19

Figure 3.5.3: Activity Diagram for Update Contents 20

Figure 3.5.4: Activity Diagram for Delete Contents 21

Figure 3.5.5: Activity Diagram for Searching by Movie's Genre 22

Figure 3.5.6: Activity Diagram for Opening 'Specific' Tab 23

Figure 3.5.7: Activity Diagram for Download Contents 24

Figure 3.5.8: Activity Diagram for Rating: (a)video (b)image 25

Figure 3.5.9: Sequence Diagram for Loading Contents:  
(a) load all videos,  
(b) load all images,  
(c) load videos by name, and  
(d) load images by name. 27

Figure 3.5.10: Sequence Diagram for  
(a) load videos by Movie Genre, and  
(b) load images by Movie Genre. 28

Figure 3.5.11: Sequence Diagram for Add Contents  
(a) videos, and (b) images. 29

Figure 3.5.12: Sequence Diagram for Update Contents  
(a) videos, and (b) images. 30

Figure 3.5.13: Sequence Diagram for Delete Contents  
(a) videos, and (b) images. 31

Figure 3.5.14: Sequence Diagram for Rating  
(a) videos, and (b) images. 32

Figure 3.7.1: Preliminary Design for Main page 35

Figure 3.7.2: Preliminary Design for Add Contents site. 36

Figure 3.7.3: Preliminary Design for Update Contents site 37

Figure 3.7.4: Preliminary Design for Delete Contents site. 38

Figure 3.7.5: Final look of the Main page 39

Figure 3.7.6: Final look of the Delete Content site 39

# LIST OF TABLES

Table i: Comparison of existing products 11

Table ii: Summary of the main functions of our proposed system / product 14

Table iii: Summary of task distribution and completeness 15

Table iv: Comparison of the RMDBS 33

Table v: Comparison of the server’s programing language 34

Table vi: Comparison of storing content method. 35

Table vii: Add, Update, Delete contents Testing. 40

Table viii: Rating Testing 41

Table ix: Sign up Testing. 42

Table x: Sign in Testing 42

# ACKNOWLEDGEMENT

On our journey of learning and growth, every step brings new opportunities to explore ourselves and sharpen our skills. The project of creating a movie storage website was a significant challenge for our team-both a chance to apply the knowledge we had gained in practice and an opportunity to push beyond our own limits. However, to complete the project in the most comprehensive way, we cannot overlook the invaluable support and guidance from our dedicated instructors.

We would like to express our heartfelt gratitude to Dr. Pham Van Tien and Assoc. Prof. Tran Thi Thanh Hai, who consistently accompanied and supported our team throughout the project. From the initial stages, with countless difficulties in shaping ideas and building the system structure, to solving complex technical problems, you not only provided meticulous guidance but also inspired us with their passion and profound expertise. Thanks to these solid foundations, we were able to successfully complete the project while gaining valuable new skills, including programming, research abilities, teamwork, and more.

Once again, we sincerely thank you both for everything you have done for our team. We deeply hope to have the opportunity to work with and learn from you in future projects!

# ABSTRACT

Under the growth of cinema, many great films have been created. However, no matter how outstanding they are, there will always be priceless scenes that leave a lasting impression on everyone. People, especially movie enthusiasts, wish to preserve them. The goal of this project is to design and build a digital library (movie gallery) to store outstanding movie clips and images. The project was implemented with a simple user interface. The design and functionality of the system were tested to optimize the user experience. Most of the project's objectives were achieved, but further research is needed to expand the system's features in the future.

# Introduction

## Motivation

The main reason for conducting this project is that more and more people want to preserve and share memorable movie moments in an easy and organized way. Currently, existing platforms still have many limitations. They often do not allow users to store or download movie clips, nor do they support discussions or sharing of favorite images and videos. This makes it difficult for movie enthusiasts to manage and enjoy their favorite movie content as they wish.

Therefore, this project aims to create a digital library (compilation) for movies. The platform will be simple and easy to use, allowing users to store, watch, download, and discuss memorable movie clips or images. One significant challenge is that movie images often cannot be downloaded directly due to the lack of a clear or official source for these images.

## Objectives

### Main objectives

The website as a compilation of scene (videos and images) from movies

### Specific objectives

* Specific objective 1: Allow people to view contents clearly, also they can download them.
* Specific objective 2: Allow people to give their opinion for the shortcuts.
* Specific objective 3: Decentralization, for the owner, allow him to adjust contents (add, update, delete).

# Methodology

## State of the art

### Methods for Movie Gallery

Approach to Web Development: Developing project ideas using software tools recommended by instructors, such as Figma, FigJam, JIRA, etc. Specifically, this involves outlining content, documentation, and problem-solving approaches.

From there, basic requirements are outlined in sequence: researching user needs, identifying suitable tools, creating a prototype or rough draft of the website, adding additional features, refining, and finalizing the product.

### Existing products for movie gallery

Table : Comparison of existing products

|  |  |  |  |
| --- | --- | --- | --- |
| Product | Main features | Advantages | Drawbacks |
| Netflix | - Movie recommendations based on user preferences.  - Manage movie watchlist and continue watching.  - Video quality in HD, 4K, and HDR. | - Easy-to-use interface, user-friendly.  - Fast and stable loading speed.  Rich and diverse content in various genres and countries | - High subscription cost for premium plans.  - Some content is region-restricted.  - Ads (in lower-cost subscription plans). |
| HBO | - Stream original HBO content.  - Provides HD and 4K video quality.  - Integration with paid cable services. | - Easy-to-use interface and easy content search.  - Fast loading speed with minim - Exclusive and well-known original content.al issues | - High subscription cost, especially without promotional offers.  - Content is limited to certain regions or countries.  - Ads may appear in some subscription plans. |
| IMDb | - Provides detailed information about movies, actors, ratings, and reviews  - Movie suggestions based on trends and user ratings.  - Offers trailers, videos, and updated information about films. | - Easy-to-use interface with a visual design.  - Fast loading speed with accurate search functionality.  - Rich content with reviews and ratings. | - No streaming service like the others.  - Ads appear on the website and mobile app.  - Some features are only available for paid users (IMDb Pro) |

### Discussion

## Application of the 9 steps in engineering design process

A diagram of a process

Description automatically generated

Figure .: Design Process

# Project implementation

## Step 1: User requirement

Method: *Study existing system*­

Final requirement: The website must allow users to view contents, download them and give their opinion about them. For an owner, it must let him add, update, and delete contents privately.

## Step 2: Specifications

### Functionality

Function 1: View contents: view on the main page or on the dynamic tab with more specific details.

Function 2: Search contents: search by name or by the original movie genre.

Function 3: Download contents: download as adjustable resolution.

Function 4: Rating contents: rate as point from 0 to 5.

Function 5: Decentralization: divide into three roles: Admin, Member and Guest (tasks that each role can do are depicted later)

Function 6: Comments: Permit users to give their opinion by word to the contents.

### Non functionality

Non- Function 1: When searching contents, the obtained results are returned fast.

Non- Function 2: Guests are allowed to view, searching and download contents. Members must log in to rate, or comment. Only admins can manage content approval.

Non- Function 3: The system should be accessible from modern web browsers (e.g., Chrome, Firefox, Edge) and mobile devices.

Table : Summary of the main functions of our proposed system / product

|  |  |  |
| --- | --- | --- |
| Function | Description of the function | Priority (Required / Optional) |
| 1 | View contents | Required |
| 2 | Search contents | Required |
| 3 | Download contents | Required (Optional for adjustable resolution) |
| 4 | Rating contents | Optional |
| 5 | Decentralization | Required |
| 6 | Comments | Optional |

## Step 3: Planning

A screenshot of a computer

Description automatically generated

Figure .: Planning of the project

Table : Summary of task distribution and completeness

|  |  |  |
| --- | --- | --- |
| Member | Tasks | Completeness |
| Bui Tuan Minh | Back-end and draw diagram | 80% |
| Nguyen Tien Dat | Front-end | 70% |
| Truong Cong Duc | Front-end | 70% |

## Step 4: Block Design

* System diagram
* Functional diagram
* Use Case diagram.

A diagram of a process

Description automatically generated

Figure .: System diagram[[1]](#footnote-1)

A diagram of a company

Description automatically generated

Figure .: Functional diagram[[2]](#footnote-2)

A diagram of a company

Description automatically generated

Figure .: Use Case Diagram[[3]](#footnote-3)

## Step 5: Detail block design

* Activity Diagram
* Sequence Diagram

### Activity Diagram

|  |  |
| --- | --- |
|  |  |
| (a) | (b) |

Figure .: Activity Diagram for (a) Sign Up[[4]](#footnote-4) and (b) Sign In[[5]](#footnote-5)

A diagram of a diagram

Description automatically generated

Figure .: Activity Diagram for Add Contents[[6]](#footnote-6)

A diagram of a diagram

Description automatically generated

Figure .: Activity Diagram for Update Contents[[7]](#footnote-7)

A diagram of a diagram

Description automatically generated

Figure .: Activity Diagram for Delete Contents[[8]](#footnote-8)

A graph with red arrows and a yellow rectangle with black text

Description automatically generated

Figure .: Activity Diagram for Searching by Movie's Genre[[9]](#footnote-9)

A diagram of a website

Description automatically generated with medium confidence

Figure .: Activity Diagram for Opening 'Specific' Tab[[10]](#footnote-10)

A diagram of a computer program

Description automatically generated

Figure .: Activity Diagram for Download Contents[[11]](#footnote-11)

|  |  |
| --- | --- |
|  | |
| (a) | (b) |

Figure .: Activity Diagram for Rating: (a)video (b)image[[12]](#footnote-12)

### Sequence Diagram

|  |
| --- |
| (a) |
| (b) |

|  |
| --- |
| (c) |
| (d) |

Figure .: Sequence Diagram for Loading Contents:[[13]](#footnote-13)  
(a) load all videos,  
(b) load all images,  
(c) load videos by name, and  
(d) load images by name.

|  |
| --- |
| (a) |
| (b) |

Figure .: Sequence Diagram for[[14]](#footnote-14)  
(a) load videos by Movie Genre, and  
(b) load images by Movie Genre.

|  |
| --- |
| (a) |
| (b) |

Figure .: Sequence Diagram for Add Contents[[15]](#footnote-15)  
(a) videos, and (b) images.

|  |
| --- |
| (a) |
| (b) |

Figure .: Sequence Diagram for Update Contents[[16]](#footnote-16)  
(a) videos, and (b) images.

|  |
| --- |
| (a) |
| (b) |

Figure .: Sequence Diagram for Delete Contents[[17]](#footnote-17)  
(a) videos, and (b) images.

|  |
| --- |
| (a) |
| (b) |

Figure .: Sequence Diagram for Rating[[18]](#footnote-18)  
(a) videos, and (b) images.

## Step 6: Best selection

Table : Comparison of the RMDBS[[19]](#footnote-19)

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Criteria 1  Price | Criteria 2  Reliability | Criteria 3  Appearance |
| MySQL | Free (Community Edition), Paid (Enterprise Edition) | High | Friendly |
| SQL Server | Paid (with free Express edition) | High | Professional |
| PostgreSQL | Free (Open Source) | High | Complicated for new users |

* Choice: MySQL, because one of the members of the group has used it before

Table : Comparison of the server’s programing language [[20]](#footnote-20)

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Criteria 1  Price | Criteria 2  Reliability | Criteria 3  Appearance |
| C# | Free | High | Rigorous datatype, static typing |
| JavaScript (JS) | Free | Medium | Non-requirement datatype declaration, dynamic typing |
| Python | Free | High | Clean, readable syntax, versatile usage |

* Choice: JS for synchronization with the client

Table : Comparison of storing content method.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Criteria 1  Price | Criteria 2  Reliability | Criteria 3  Appearance |
| Online (by someone else) | Free | Medium | Available |
| Online (by ourselves) | Free | High | Require editing and uploading |
| Local | Free | High | Not portable |

* Choice: Priorly the first option, then the second option. However, still store the resources locally for avoiding risks.

## Step 7: Prototyping

Drawn by Goodnotes.

A grid paper with writing on it

Description automatically generated

Figure .: Preliminary Design for Main page

A graph paper with text and arrows

Description automatically generated

Figure .: Preliminary Design for Add Contents site.

A grid paper with writing on it

Description automatically generated

Figure .: Preliminary Design for Update Contents site

A grid paper with text and images

Description automatically generated

Figure .: Preliminary Design for Delete Contents site.

A screenshot of a computer

Description automatically generated

Figure .: Final look of the Main page

A screenshot of a computer

Description automatically generated

Figure .: Final look of the Delete Content site

## Step 8: Testing

Table : Add, Update, Delete contents Testing.

|  |  |
| --- | --- |
| Affective factors | * Contents’ ID: user can type negative value or non-existing values. * Links: They should start with *http://* or *https://* |
| Scenario for testing | * Provide two type of JSON data: one is invalid, and the others is valid. * Using Postman, to test the server independently. * After obtained a good result, test with the client’s site. |
| Conducting | * Data:   + Invalid data:     - Let the id be negative or non-existing.     - Let the links not to start with *http://* or *https://*   + Valid data |
| Analyze the test result | * Result:   + Invalid data: response the error status   + Valid data: response the successful status * Comments: The program realizes the invalid data and response to the user |

Table : Rating Testing

|  |  |
| --- | --- |
| Affective factors | * Contents’ ID: user can type negative value or non-existing values. * Point: it must be the integer number from 0 to 5 * Only Admin and Member are allowed to do this feature. |
| Scenario for testing | * Provide two type of JSON data: one is invalid, and the others is valid. * Using Postman, to test the server independently. * After obtained a good result, test with the client’s site. |
| Conducting | * Data:   + Invalid data:     - Let the id be negative or non-existing.     - Let the point be not the integer number from 0 to 5.   + Valid data * At the client’s site, try that whether the feature works if the user has not signed in. |
| Analyze the test result | * Result:   + Invalid data: When testing the server independently, response the error status. (We didn’t test this situation at the client’s site because rigorous settings).   + Valid data: response the successful status and change the average point for each video and image too.   + If users have not signed in yet, the program show out the problem. * Comments: The program works successfully. |

Table : Sign up Testing.

|  |  |
| --- | --- |
| Affective factors | * The username must be non-existing |
| Scenario for testing | * Provide two type of JSON data: one is invalid, and the others is valid. * Using Postman, to test the server independently. * After obtained a good result, test with the client’s site. |
| Conducting | * Data:   + Invalid data: The existing username   + Valid data |
| Analyze the test result | * Result:   + Invalid data: When testing the server independently, response the error status.   + Valid data: response the successful * Comments: The program works successfully. |

Table : Sign in Testing

|  |  |
| --- | --- |
| Affective factors | * The username must be existing |
| Scenario for testing | * Provide two type of JSON data: one is invalid, and the others is valid. * Using Postman, to test the server independently. * After obtained a good result, test with the client’s site. |
| Conducting | * Data:   + Invalid data: The non-existing username   + Valid data |
| Analyze the test result | * Result:   + Invalid data: When testing the server independently, response the error status.   + Valid data: response the successful * Comments: The program works successfully. |

# Discussions

* Limitation:
  + Update, Delete Contents using two independent form requiring id   
    -> not good for the UX because the user must look up the id in dynamic tabs.
  + Do not contain Comment, Adjustable resolution for downloading.
  + Users have to view contents randomly that not based on their interests.

# Conclusions and future works

* Achievement compared to the objective.
  + Achievement: get a local version of the website whose features: View Contents (main page and dynamic tab); Decentralization; Add, Update, Delete Contents; Download Contents (no adjustable resolution), Rating, No Comments feature.
  + Compare to the objectives: 75%
* Future Work:
  + Redesign the method about technicality and user’s experience.
  + Redesign the database and adjust the source code to add Comments features let the program display the contents based on users’ interests.
  + Learn a method to let users download contents that theses’ resolution is adjustable.

# Reference

1. JavaScript: W3Schools, <https://www.w3schools.com/js/default.asp>
2. CSS: W3Schools, <https://www.w3schools.com/css/default.asp>
3. HTML: W3Schools, <https://www.w3schools.com/html/default.asp>
4. How the web work: <https://developer.mozilla.org/en-US/docs/Learn_web_development/Getting_started/Web_standards/How_the_web_works>
5. Express JS: <https://expressjs.com/>
6. mysql2 API: <https://sidorares.github.io/node-mysql2/docs>
7. UML: freeCodeCamp.org <https://www.youtube.com/watch?v=WnMQ8HlmeXc>

# APPENDIX – TEST RESULT

1. Test result for Update Image (valid data)

|  |  |
| --- | --- |
| Method: POST | url: <http://localhost:3000/api/admin/UpdateImage> |
| Input | {      "id": 17[[21]](#footnote-21),      "title": "Wayne’s World (1992) Head banging to Bohemian Rhapsody",      "storage": "cloud",      "src": "https://panandslam.com/wp-content/uploads/2020/04/waynes-world-car-queen.jpg?w=636&h=424",      "movie\_genre": "Comedy",      "image\_style": "Happy",      "width": 636,      "height": 424,      "format": "png",      "download\_link": "https://panandslam.com/wp-content/uploads/2020/04/waynes-world-car-queen.jpg?w=636&h=424",      "upload\_date": **null**  } |
| Output | {      "message": "Image content updated successfully",      "data": {          "fieldCount": 0,          "affectedRows": 1,          "insertId": 0,          "info": "Rows matched: 1  Changed: 1  Warnings: 0",          "serverStatus": 2,          "warningStatus": 0,          "changedRows": 1      }  } |

1. Test result for Update Image (negative ID)

|  |  |
| --- | --- |
| Method: POST | url: <http://localhost:3000/api/admin/UpdateImage> |
| Input | {      "id": -1,      "title": "Wayne’s World (1992) Head banging to Bohemian Rhapsody",      "storage": "cloud",      "src": "https://panandslam.com/wp-content/uploads/2020/04/waynes-world-car-queen.jpg?w=636&h=424",      "movie\_genre": "Comedy",      "image\_style": "Happy",      "width": 636,      "height": 424,      "format": "png",      "download\_link": "https://panandslam.com/wp-content/uploads/2020/04/waynes-world-car-queen.jpg?w=636&h=424",      "upload\_date": **null**  } |
| Output | {      "error": "Failed to update image content",      "details": "Image ID not found"  } |

1. Test result for Rating

|  |  |
| --- | --- |
|  | The user has not rated the content before |
|  | The user has already rated the content before |

1. The system is divided into two parts: front-end and back-end. The bac k-end includes the server and the DAL to interact with the database. The front-end includes the interface, created by HTML and CSS, and corresponding script, created by JavaScript. [↑](#footnote-ref-1)
2. After access the website, users can view contents, sign up and sign in. After signed in, they must sign up before, if they are admins, they are allowed to add, update, and delete contents, otherwise, only rating and comments. [↑](#footnote-ref-2)
3. The diagram demonstrates the use cases. There are three use cases: Guest, Member and Admin.

   Guests are allowed to view, searching and download contents. They can sign up to become Members.

   Members can rate, or comment.

   Only admins can manage content approval.

   However, Member and Admin must sign in before to use their privilege. [↑](#footnote-ref-3)
4. The diagram (a) depicts the ‘Sign Up’ process: In this case, the form includes these attributes: full name, day of birth, username, and password. The value is considered as ‘valid’ if all attributes are filled and the username is not existing. [↑](#footnote-ref-4)
5. The diagram (b) depicts the ‘Sign In’ process: In this case, the form includes these attributes: username, and password. The value is considered as ‘valid’ if the username existing and the password is correct.

   In each diagram, the process occurs as each following step:

   The user clicks on the function button, a form will be displayed.

   The user fills the form, then click the ‘Send’ button (it named ‘Sign Up’ or ‘Sign In’).

   The program checks if the value is valid? If not, the user must input again, else, he continues the next step, the program will switch to the main page (Sign Up) or display the signature interface (Sign In) [↑](#footnote-ref-5)
6. The video form includes title, storage, source link, movie’s genre, format, download link, upload date, video’s mood, and duration. Therein, title, storage, source link, movie’s genre, download link and video’s mood are compulsory.  
   The image form includes title, storage, source link, movie’s genre, format, download link, upload date, image’s mood, width, and height. Therein, title, storage, source link, movie’s genre, download link and image’s mood are compulsory.

   The value is considered as ‘valid’ if all compulsory attributes are filled. Also, links must start with ‘https://’ or ‘http://’

   The process occurs as each following step:

   The user clicks on the function button, a form will be displayed.

   The user fills one of two forms (video/image) then click the corresponding button of each form.

   The program checks if the value is valid? If not, the user must input again, else, a notification will appear, states that the process is successful. [↑](#footnote-ref-6)
7. The video form includes video’s id, title, storage, source link, movie’s genre, format, download link, upload date, video’s mood, and duration. Therein, video’s id, title, storage, source link, movie’s genre, download link and video’s mood are compulsory.  
   The image form includes image’s id, title, storage, source link, movie’s genre, format, download link, upload date, image’s mood, width, and height. Therein, video’s id, title, storage, source link, movie’s genre, download link and image’s mood are compulsory.

   The value is considered as ‘valid’ if all compulsory attributes are filled. Also, links must start with ‘https://’ or ‘http://’ and the id must be a positive integer number and existing.

   The process occurs as each following step:

   The user clicks on the function button, a form will be displayed.

   The user fills one of two forms (video/image) then click the corresponding button of each form. (if he wants to keep the old value, he must type the old value)

   The program checks if the value is valid? If not, the user must input again, else, a notification will appear, states that the process is successful. [↑](#footnote-ref-7)
8. Each form contains only one attribute named ‘id’. The value is considered as ‘valid’ if the id is a positive integer number and existing.

   The process occurs as each following step:

   The user clicks on the function button, a form will be displayed.

   The user fills one of two forms (video/image) then click the corresponding button of each form.

   The program checks if the value is valid? If not, the user must input again, else, a notification will appear, states that the process is successful. [↑](#footnote-ref-8)
9. There is a drop-down in the main page for this feature.

   The process occurs as each following step:

   The user selects the genre in the drop-down, then clicks the ‘Send’ button (lies next to the drop-down).

   Videos and images whose original movie selected will be displayed. [↑](#footnote-ref-9)
10. In the main page, videos and images are displayed. To open the ‘specific’ tab (the tab displaying specific details of a certain video or image), users click on the title of that video or image. [↑](#footnote-ref-10)
11. There are two ways to download contents:

    One, on the main page, there are ‘download’ buttons displayed with contents. Users click on the button corresponding to a certain video or image to download.

    Two, user open the ‘specific’ tab by clicking the titles of a certain video or image. In this tab, there is a ‘download’ button too, click on it to download. [↑](#footnote-ref-11)
12. The process occurs following these steps:

    In the main page, the user clicks on the title of a certain content to open the ‘specific’ tab.

    There is a drop-down at the bottom of the site, the user selects the point and click the ‘Send’ button, a notification will appear, states that the process is successful.

    Note: the user must sign in before to use this feature. [↑](#footnote-ref-12)
13. The process occurs as following steps:

    Based on what users type in the textbox (for searching), the program will send a GET request to a suitable API, `api/videos` or `api/images` if the value of the textbox is null, and `api/videos?value=` or `api/images?value=` if the value of the textbox is not null to the server.

    The server then calls the appreciate functions in DAL layers to get results from the server: ‘loadAllVideos’ or ‘loadAllImage’ if the value of the textbox is null, and ‘loadVideosByName’ or ‘loadImagesByName’ if the value of the textbox is not null.

    The result then sent in a reverse way to users. [↑](#footnote-ref-13)
14. The process occurs as following steps:

    Based on what users select, the program will send a GET request to a suitable API, `api/videos/genreFilter?value=` or `api/images/genreFilter?value=` to the server.

    The server then calls the appreciate functions in DAL layers to get results from the server: ‘loadVideosByMovieGenre’ or ‘loadImagesByMovieGenre’.

    The result then sent in a reverse way to users. [↑](#footnote-ref-14)
15. After the admin clicks on the ‘Add Video’ or ‘Add Image’ button, the interface sends a POST request to the API `/api/admin/AddVideo` or `/api/admin/AddImage`. The server calls the `addVideoContent` or ‘addImageContent’ function in the DAL layer to save the content to the database and returns the result. [↑](#footnote-ref-15)
16. After the admin clicks on the ‘Update Video’ or ‘Update Image’ button, the interface sends a POST request to the API `/api/admin/UpdateVideo` or `/api/admin/UpdateImage`. The server calls the `addVideoContent` or ‘addImageContent’ function in the DAL layer to save the content to the database and returns the result. [↑](#footnote-ref-16)
17. After the admin clicks on the ‘Delete Video’ or ‘Delete Image’ button, the interface sends a POST request to the API *‘/api/admin/DeleteVideo’* or *‘/api/admin/DeleteImage’*. The server calls the `deleteVideoContent` or ‘deleteImageContent’ function in the DAL layer to save the content to the database and returns the result. [↑](#footnote-ref-17)
18. The process occurs as following steps:

    After users give a point, the interface sends a POST request to the API *`/api/member/RateVideo`* or *`api/member/RateImage`.*

    The server calls the `checkVideoRatingExists` or `checkImageRatingExists` function in the DAL layer to check that weather that content is rated before, then return the result to the server.

    The server then calls the suitable function: `addVideoRating`/`addImageRating` if that content has not been rated before; and `updateVideoRating`/`updateImageRating` has been rated before.

    A notification will also be sent to users. [↑](#footnote-ref-18)
19. <https://aws.amazon.com/vi/compare/the-difference-between-sql-and-mysql/> , <https://aws.amazon.com/vi/compare/the-difference-between-mysql-vs-postgresql/> [↑](#footnote-ref-19)
20. <https://learn.microsoft.com/en-us/dotnet/csharp/>   
    <https://developer.mozilla.org/en-US/docs/Web/JavaScript>   
    <https://www.python.org/doc/> [↑](#footnote-ref-20)
21. In this case, we insert an image whose id is 17 before, so it is valid. [↑](#footnote-ref-21)