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 FIGURES

LIST OF TABLES

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.

1. Introduction

1.1. 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.

1.2. 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).

2. Methodology

2.1. 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 i: 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

2.2. Application of the 9 steps in engineering design process

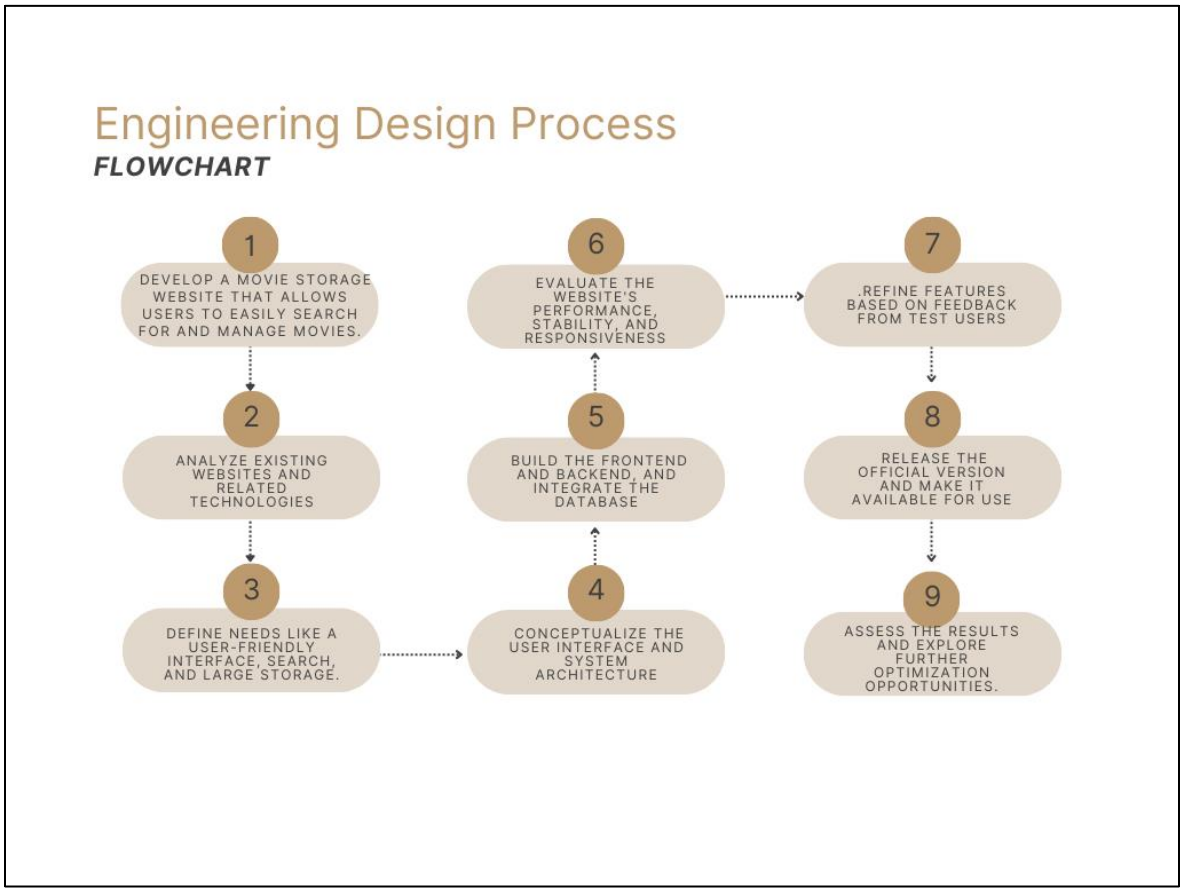

Engineering Design Process
FLOWCHART
1
DEVELOP A MOVIE STORAGE WEBSITE THAT ALLOWS USERS TO EASILY SEARCH FOR AND MANAGE MOVIES.
2
ANALYZE EXISTING WEBSITES AND RELATED TECHNOLOGIES
3
DEFINE NEEDS LIKE A USER-FRIENDLY INTERFACE, SEARCH, AND LARGE STORAGE.
4
CONCEPTUALIZE THE USER INTERFACE AND SYSTEM ARCHITECTURE
5
BUILD THE FRONTEND AND BACKEND, AND INTEGRATE THE DATABASE
6
EVALUATE THE WEBSITE'S PERFORMANCE, STABILITY, AND RESPONSIVENESS
7
.REFINE FEATURES BASED ON FEEDBACK FROM TEST USERS
8
RELEASE THE OFFICIAL VERSION AND MAKE IT AVAILABLE FOR USE
9
ASSESS THE RESULTS AND EXPLORE FURTHER OPTIMIZATION OPPORTUNITIES.

Figure 2.2.1: Design Process

3. Project implementation

3.1. 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.

3.2. 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.

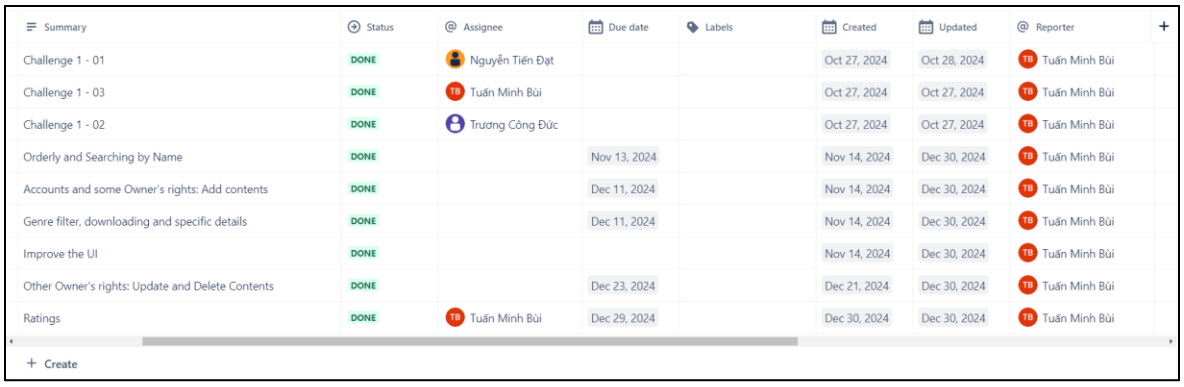
Non- Function 2: Guests are allowed to view, searching and download contents. Members must log in to rate, or comment. Only admins can manage content approval.

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

Table ii: 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

3.3. Step 3: Planning



Summary	Status	Assignee	Due date	Labels	Created	Updated	Reporter
Challenge 1 - 01	DONE	Nguyễn Tiến Đạt			Oct 27, 2024	Oct 28, 2024	Tuấn Minh Bùi
Challenge 1 - 03	DONE	Tuấn Minh Bùi			Oct 27, 2024	Oct 27, 2024	Tuấn Minh Bùi
Challenge 1 - 02	DONE	Trương Công Đức			Oct 27, 2024	Oct 27, 2024	Tuấn Minh Bùi
Orderly and Searching by Name	DONE		Nov 13, 2024		Nov 14, 2024	Dec 30, 2024	Tuấn Minh Bùi
Accounts and some Owner's rights: Add contents	DONE		Dec 11, 2024		Nov 14, 2024	Dec 30, 2024	Tuấn Minh Bùi
Genre filter, downloading and specific details	DONE		Dec 11, 2024		Nov 14, 2024	Dec 30, 2024	Tuấn Minh Bùi
Improve the UI	DONE				Nov 14, 2024	Dec 30, 2024	Tuấn Minh Bùi
Other Owner's rights: Update and Delete Contents	DONE		Dec 23, 2024		Dec 21, 2024	Dec 30, 2024	Tuấn Minh Bùi
Ratings	DONE	Tuấn Minh Bùi	Dec 29, 2024		Dec 30, 2024	Dec 30, 2024	Tuấn Minh Bùi

Figure 3.3.1: Planning of the project

Table iii: 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%

3.4. Step 4: Block Design

- System diagram
- Functional diagram
- Use Case diagram.

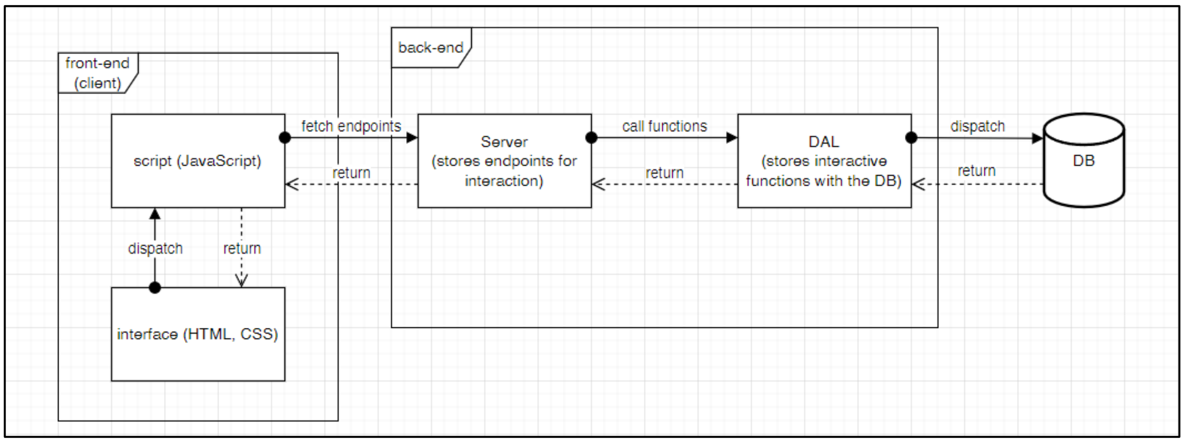


Figure 3.4.1: System diagram[1]

[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.

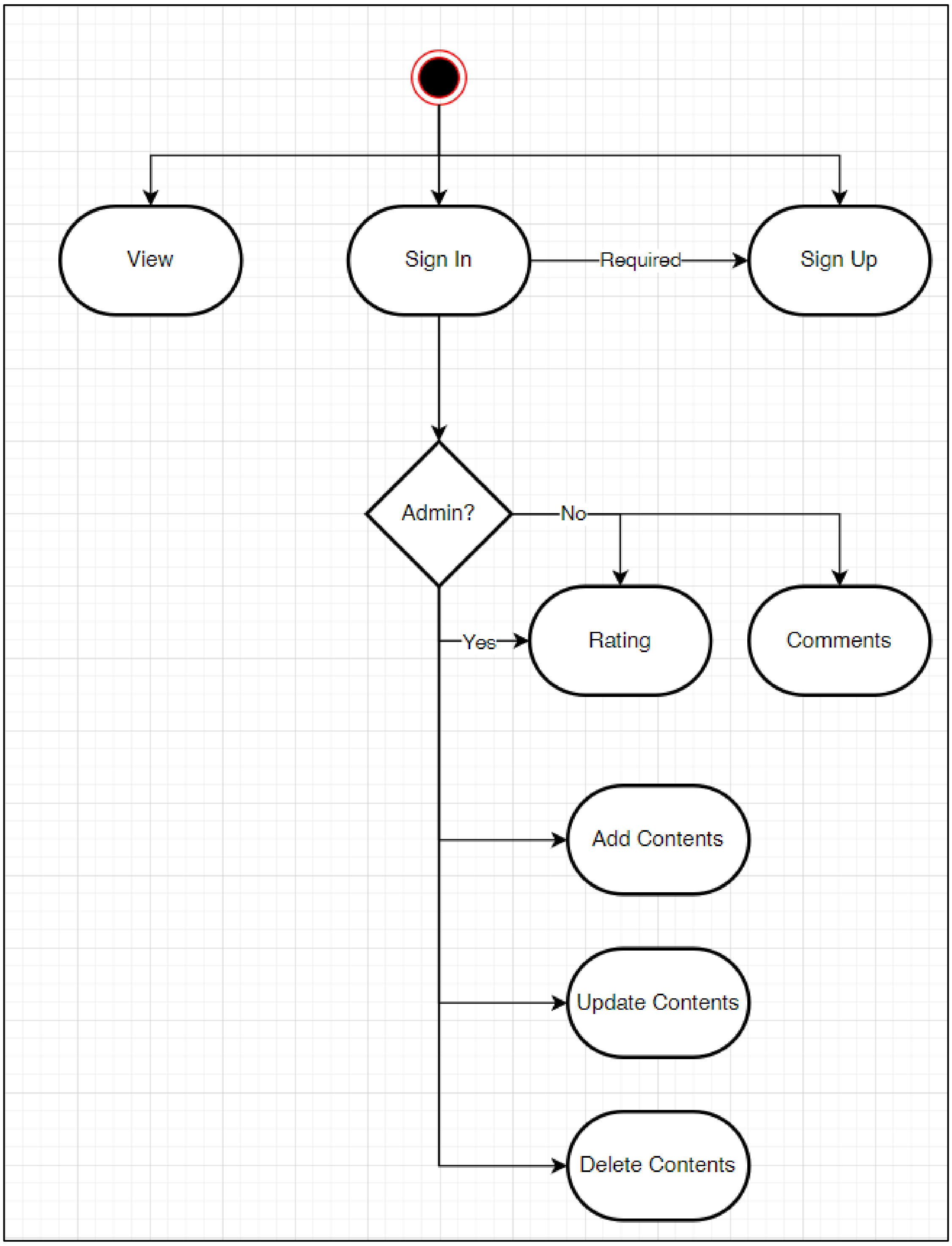


Figure 3.4.2: Functional diagram[2]

[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.

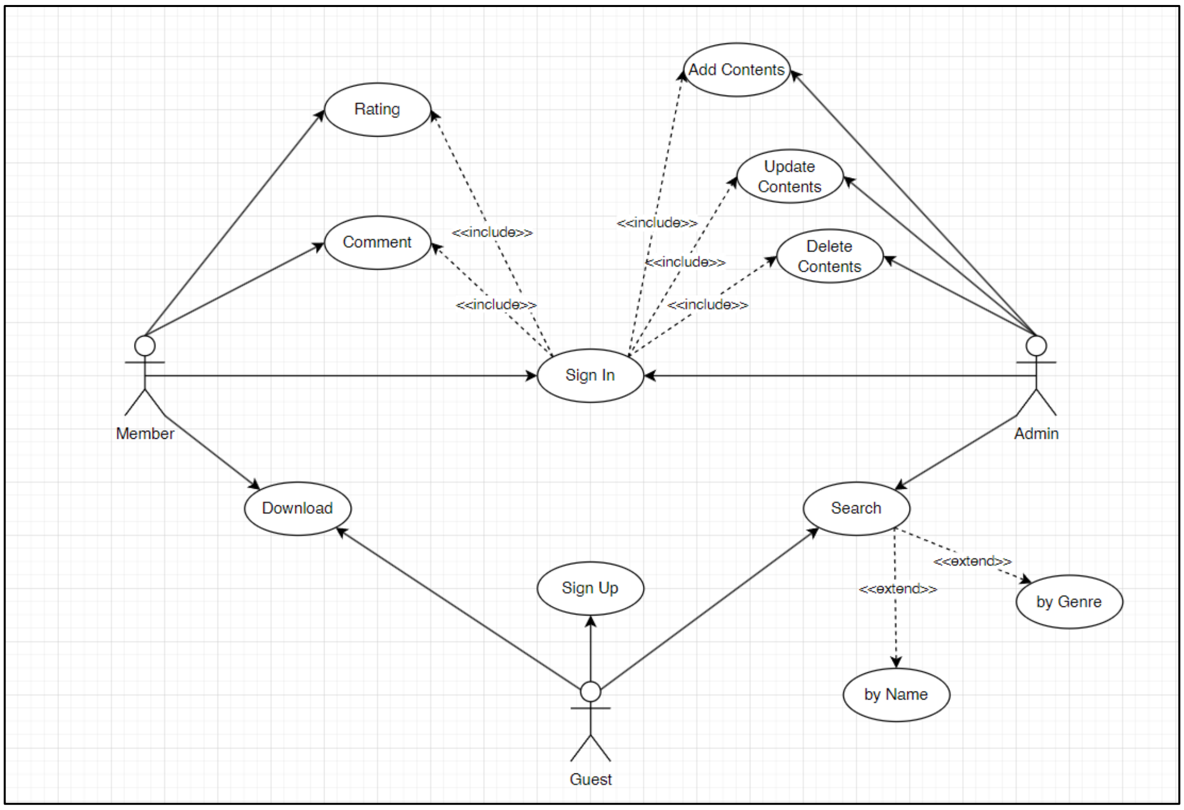


Figure 3.4.3: Use Case Diagram[3]

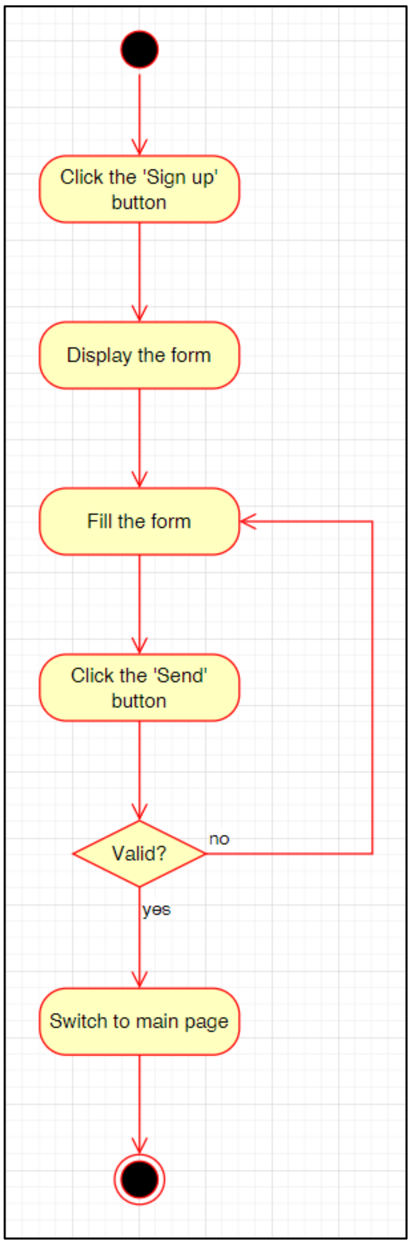
3.5. Step 5: Detail block design

- Activity Diagram
- Sequence Diagram

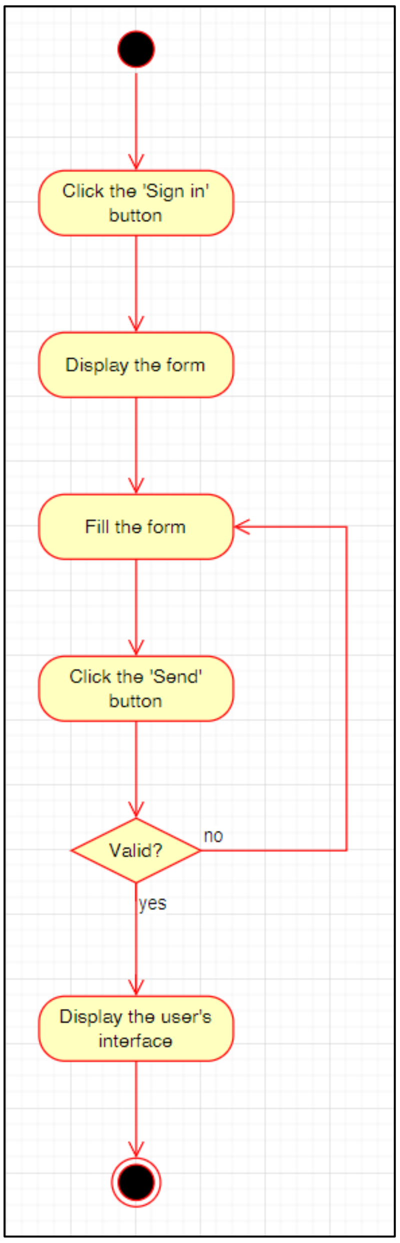
[3] The diagram demonstrates the use cases. There are three use cases: Guest, Member and Admin.
- Guests are allowed to view, searching and download contents. They can sign up to become Members.
- Members can rate, or comment.
- Only admins can manage content approval.

However, Member and Admin must sign in before to use their privilege.

3.5.1. Activity Diagram



(a)



(b)

Figure 3.5.1: Activity Diagram for (a) Sign Up[4] and (b) Sign In[5]

[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.
[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)

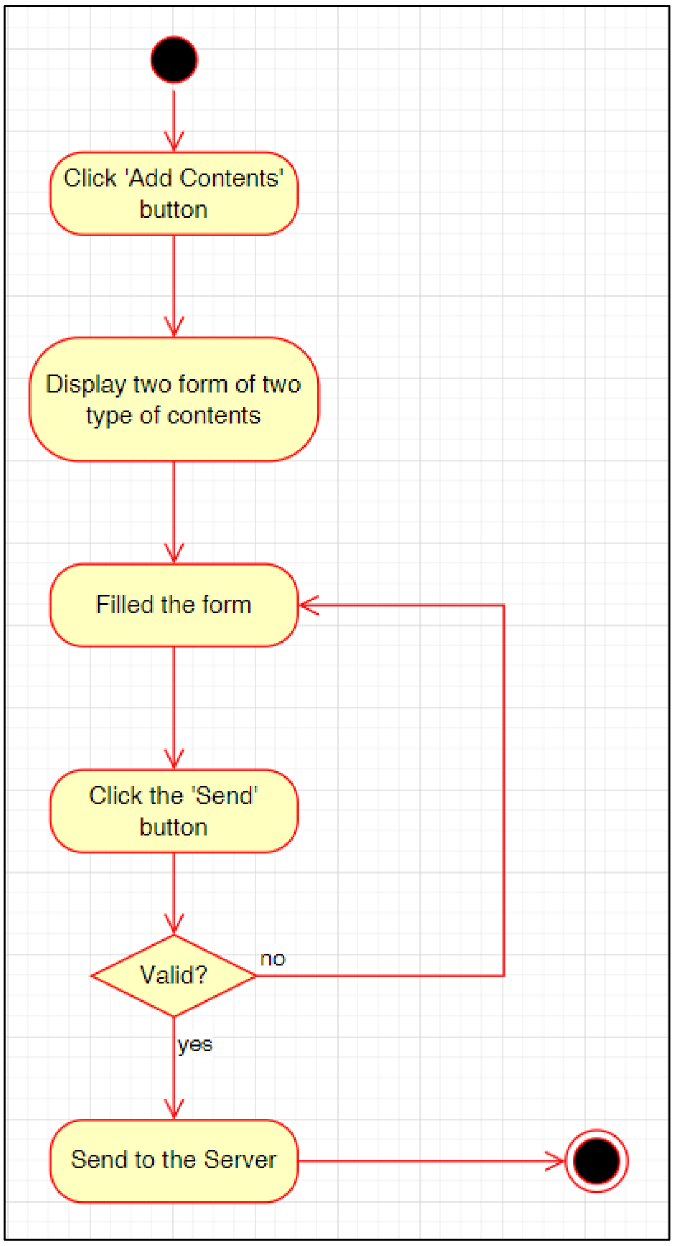

Click 'Add Contents' button
Display two form of two type of contents
Filled the form
Click the 'Send' button
Valid?
no
yes
Send to the Server

Figure 3.5.2: Activity Diagram for Add Contents[6]

[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.

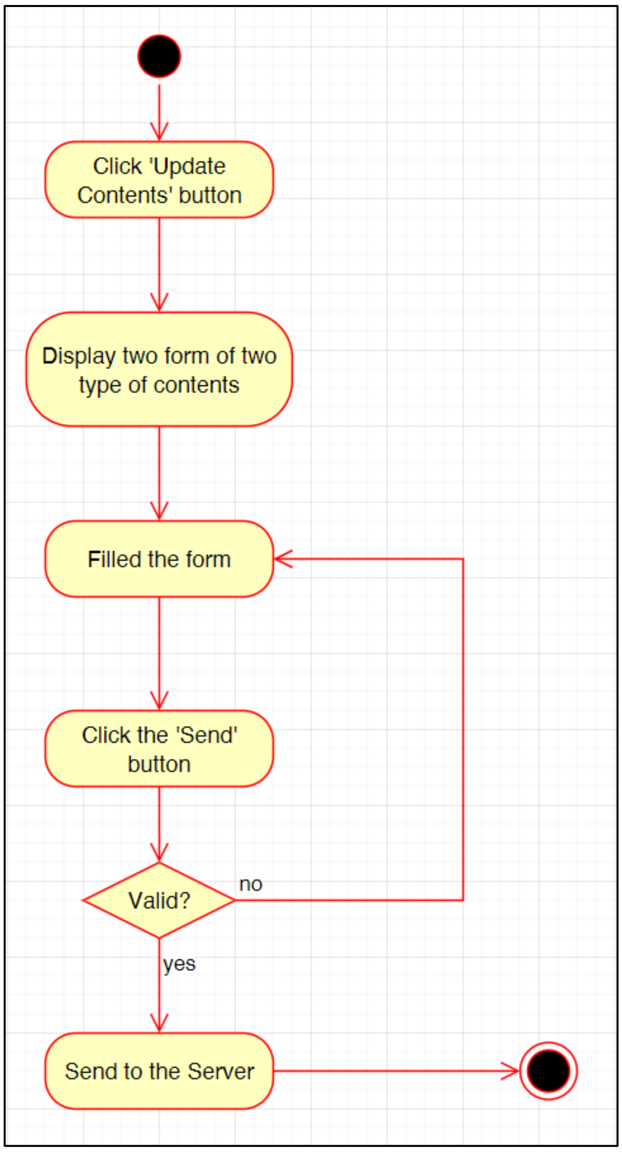

Click 'Update Contents' button
Display two form of two type of contents
Filled the form
Click the 'Send' button
Valid?
no
yes
Send to the Server

Figure 3.5.3: Activity Diagram for Update Contents[7]

[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.

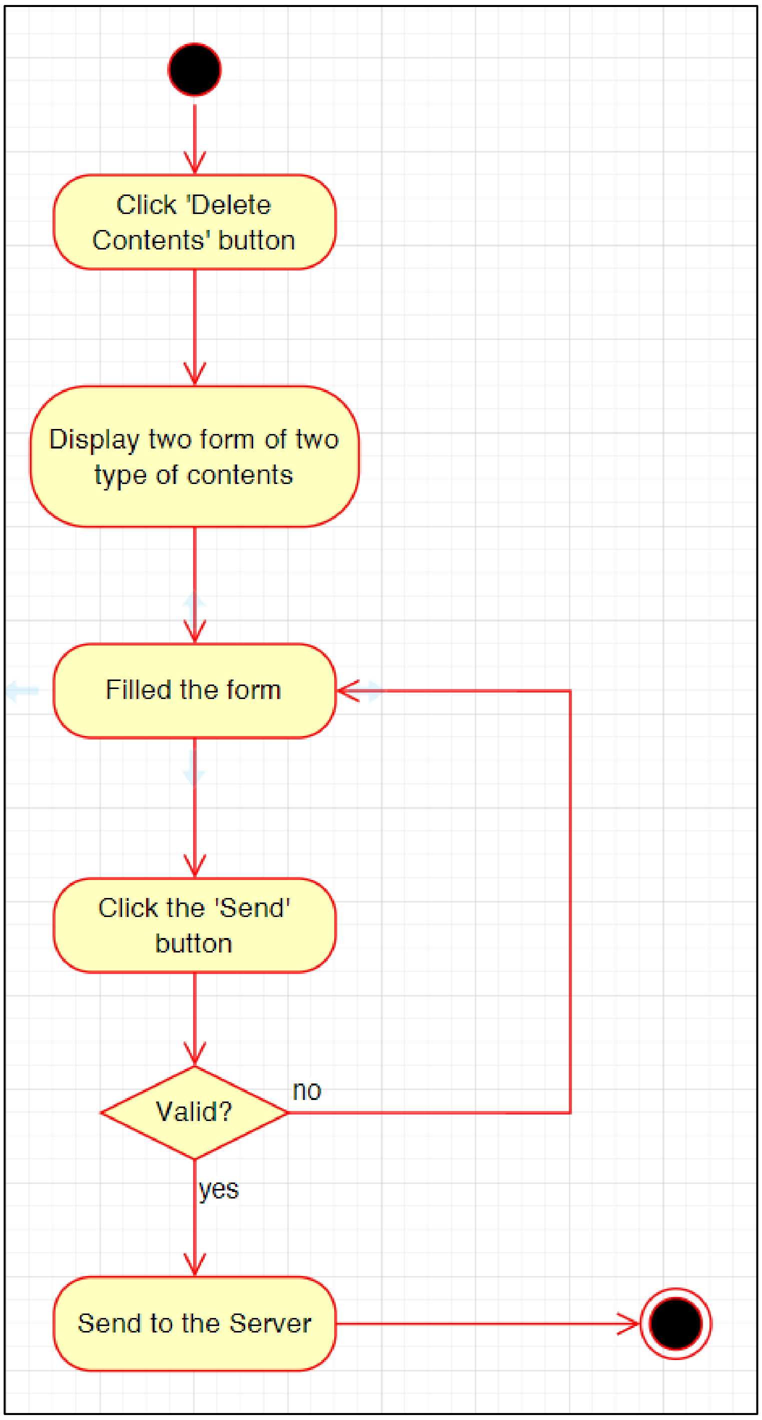

Click 'Delete Contents' button
Display two form of two type of contents
Filled the form
Click the 'Send' button
Valid?
no
yes
Send to the Server

Figure 3.5.4: Activity Diagram for Delete Contents[8]

[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.

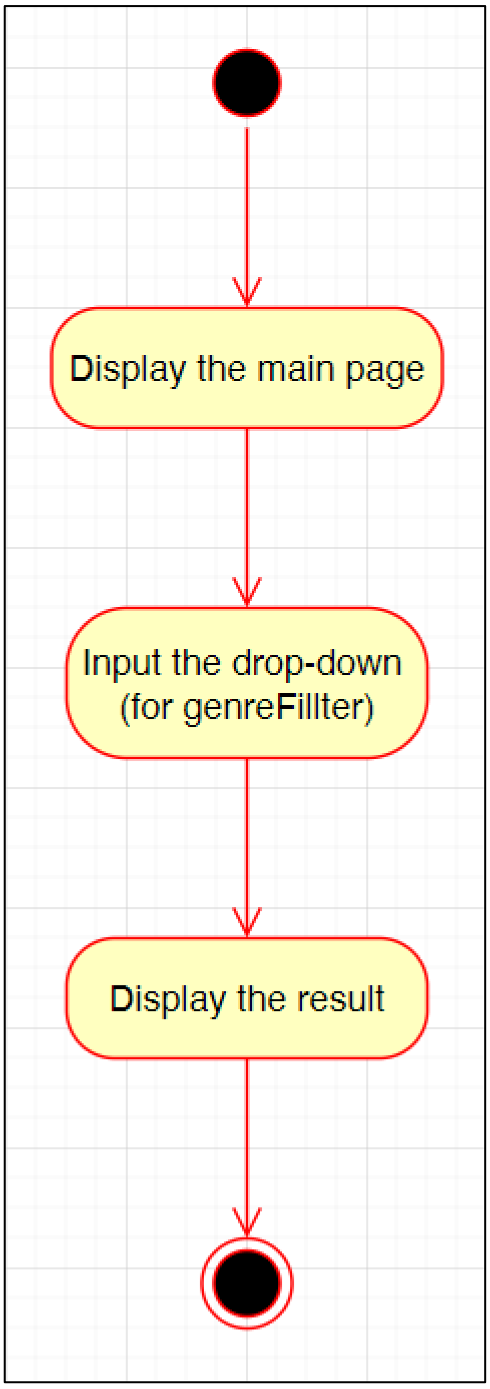


Figure 3.5.5: Activity Diagram for Searching by Movie's Genre[9]

[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.

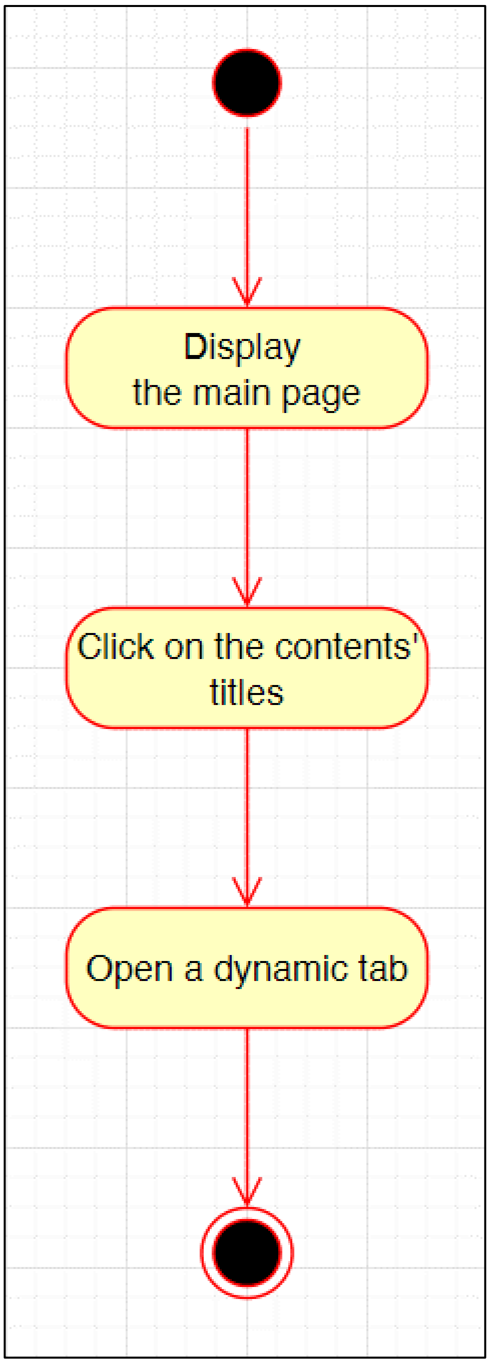


Figure 3.5.6: Activity Diagram for Opening 'Specific' Tab[10]

[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.

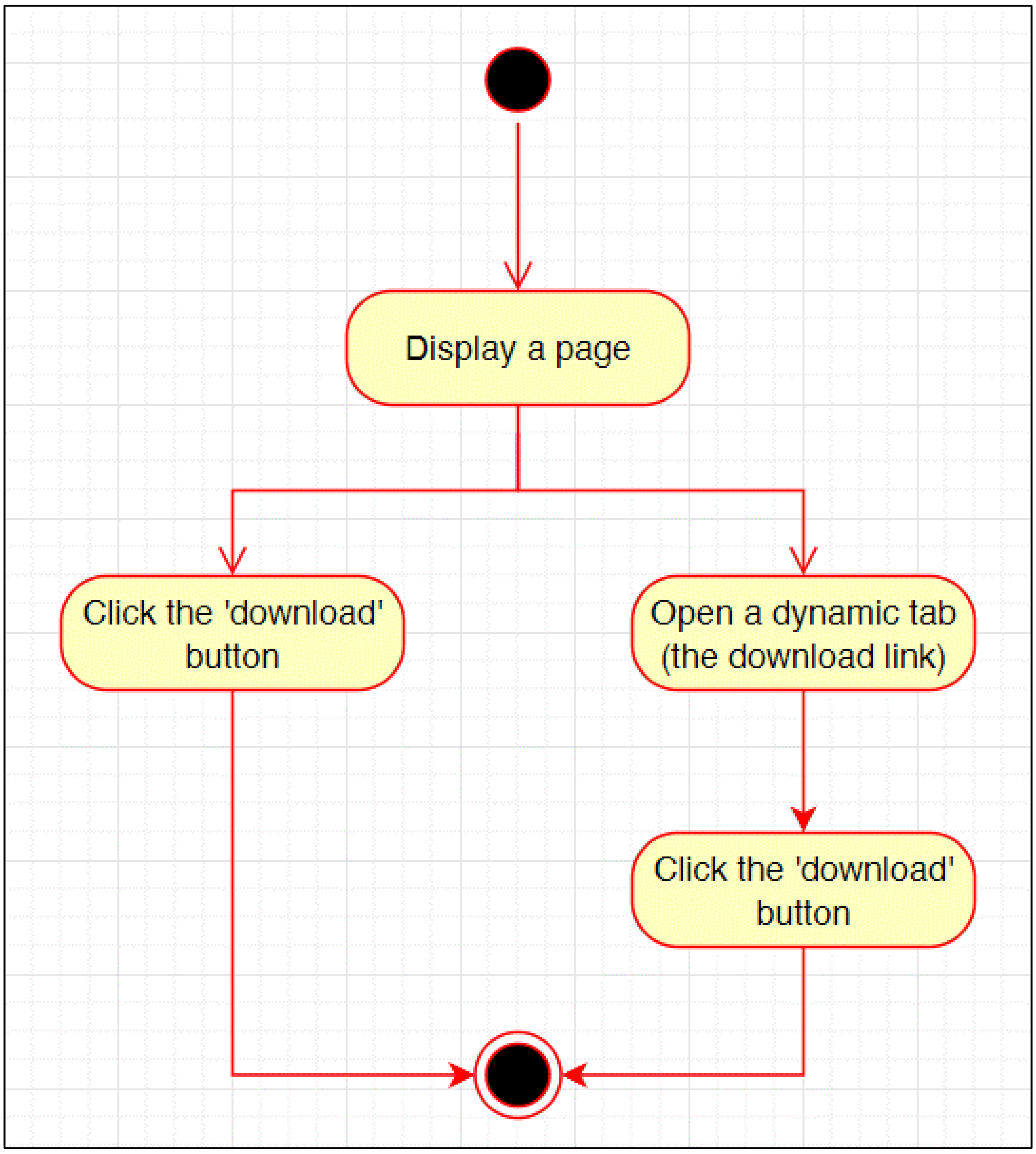
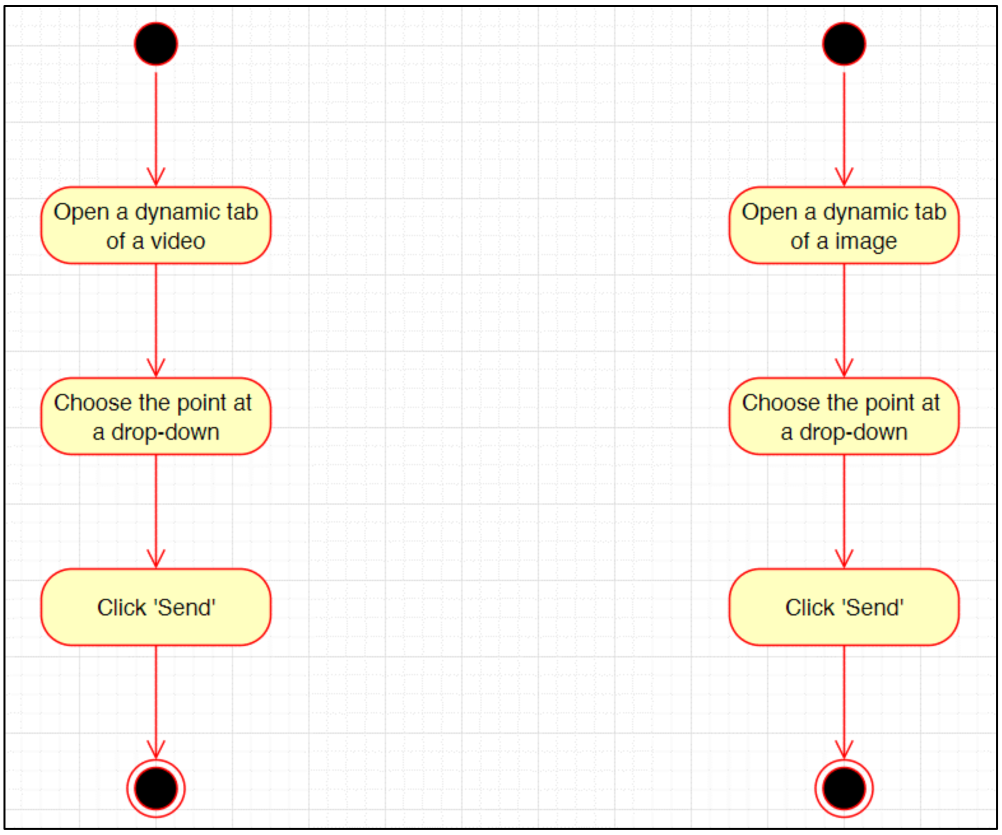

Display a page
Click the 'download' button
Open a dynamic tab (the download link)
Click the 'download' button

Figure 3.5.7: Activity Diagram for Download Contents[11]

[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.


Open a dynamic tab of a video
Choose the point at a drop-down
Click 'Send'
Open a dynamic tab of a image
Choose the point at a drop-down
Click 'Send'

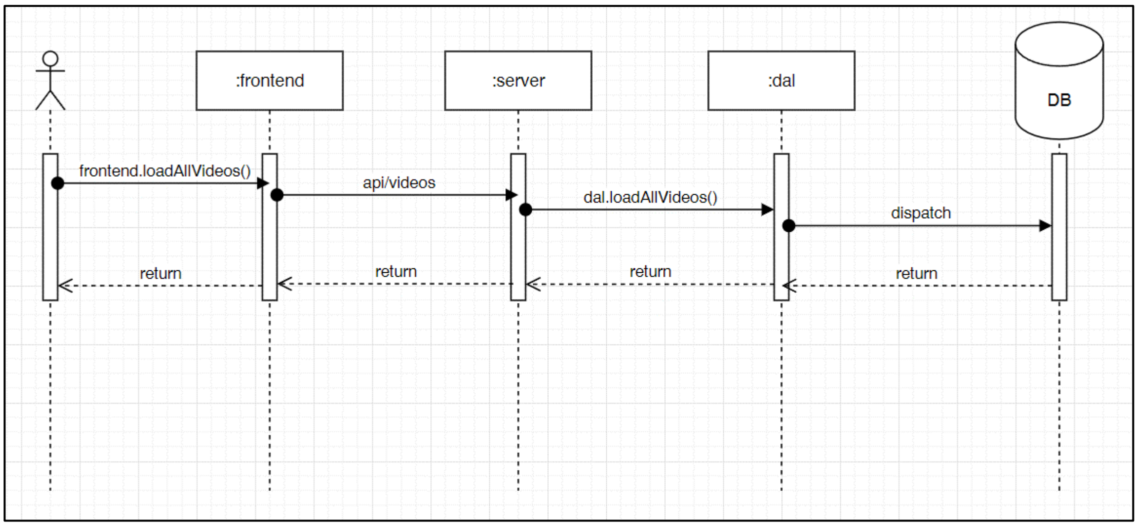
(a) *(b)*

Figure 3.5.8: Activity Diagram for Rating: (a)video (b)image[12]

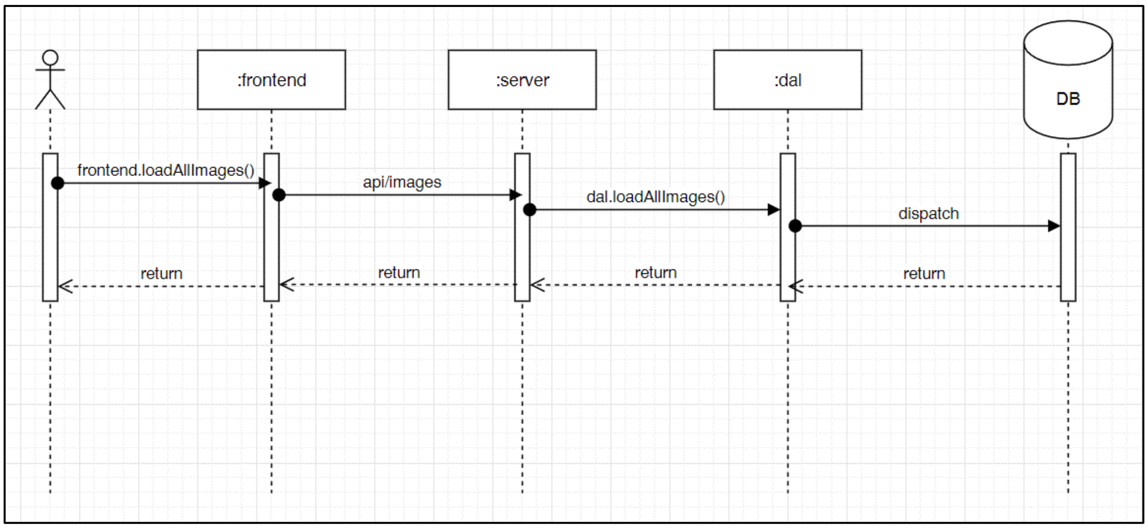
[12] The process occurs following these steps:
- In the main page, the user clicks on the title of a certain content to open the 'specific' tab.
- There is a drop-down at the bottom of the site, the user selects the point and click the 'Send' button, a notification will appear, states that the process is successful.

Note: the user must sign in before to use this feature.

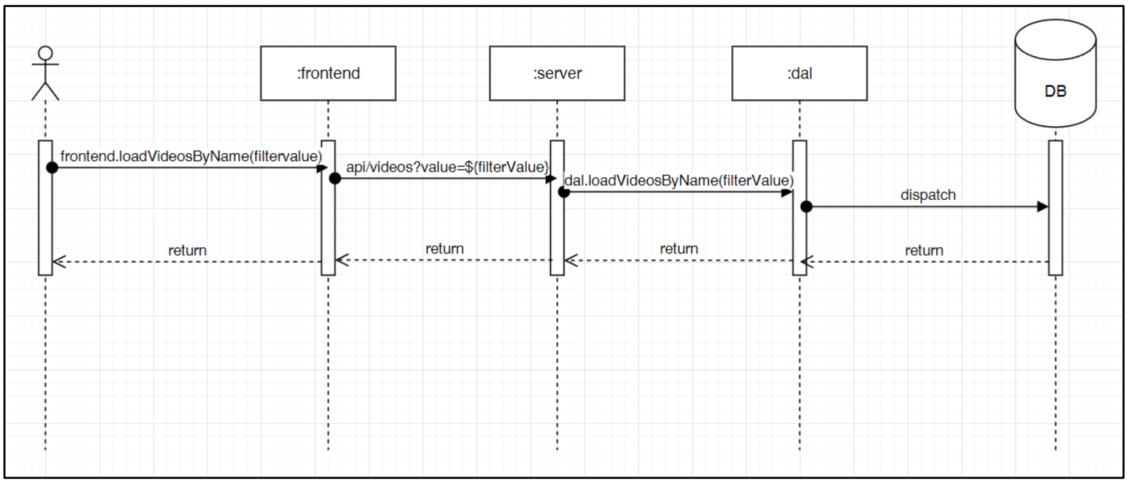
3.5.2. Sequence Diagram



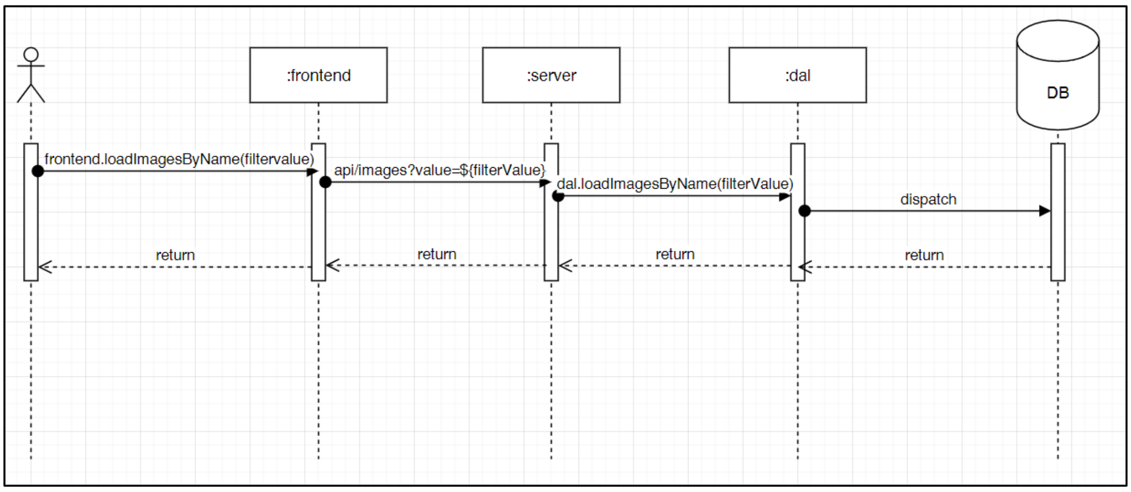
(a)



(b)


:frontend
:server
:dal
DB
frontend.loadVideosByName(filtervalue)
api/videos?value=${filterValue}
dal.loadVideosByName(filterValue)
dispatch
return
return
return
return

(c)

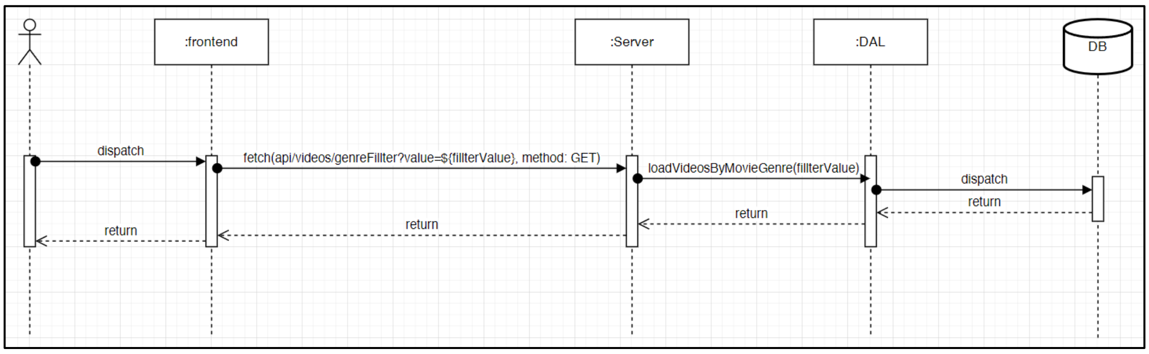

:frontend
:server
:dal
DB
frontend.loadImagesByName(filtervalue)
api/images?value=${filterValue}
dal.loadImagesByName(filterValue)
dispatch
return
return
return
return

(d)

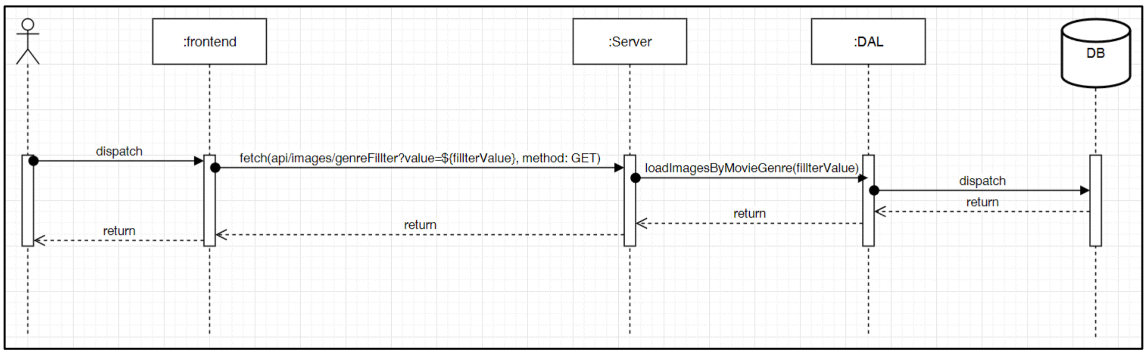
Figure 3.5.9: Sequence Diagram for Loading Contents:[13]
(a) load all videos,
(b) load all images,
(c) load videos by name, and
(d) load images by name.

[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.


:frontend
:Server
:DAL
DB
dispatch
fetch(api/videos/genreFillter?value=${fillterValue}, method: GET)
loadVideosByMovieGenre(fillterValue)
dispatch
return
return
return
return

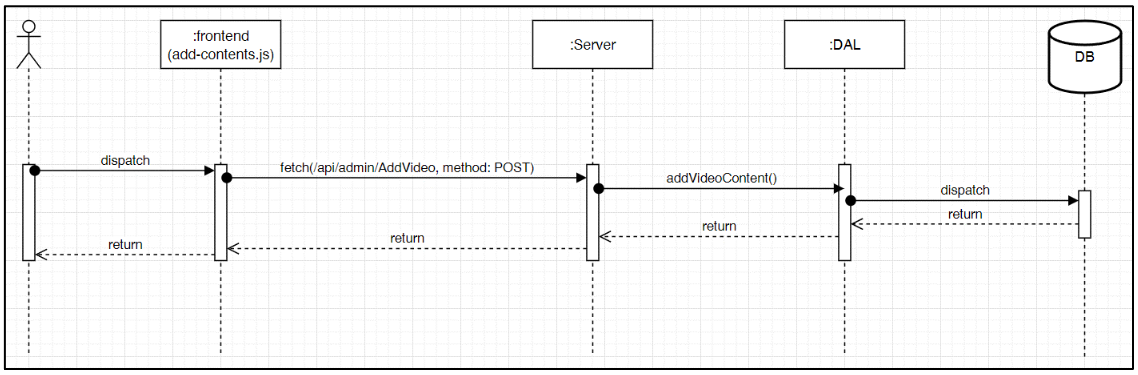
(a)


:frontend
:Server
:DAL
DB
dispatch
fetch(api/images/genreFillter?value=${fillterValue}, method: GET)
loadImagesByMovieGenre(fillterValue)
dispatch
return
return
return
return

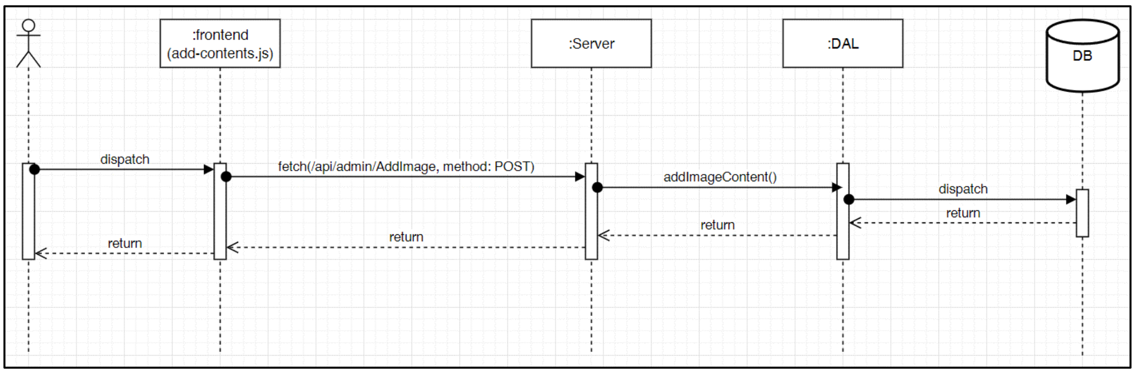
(b)

Figure 3. 5. 10: Sequence Diagram for[14]
(a) load videos by Movie Genre, and
(b) load images by Movie Genre.

[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.



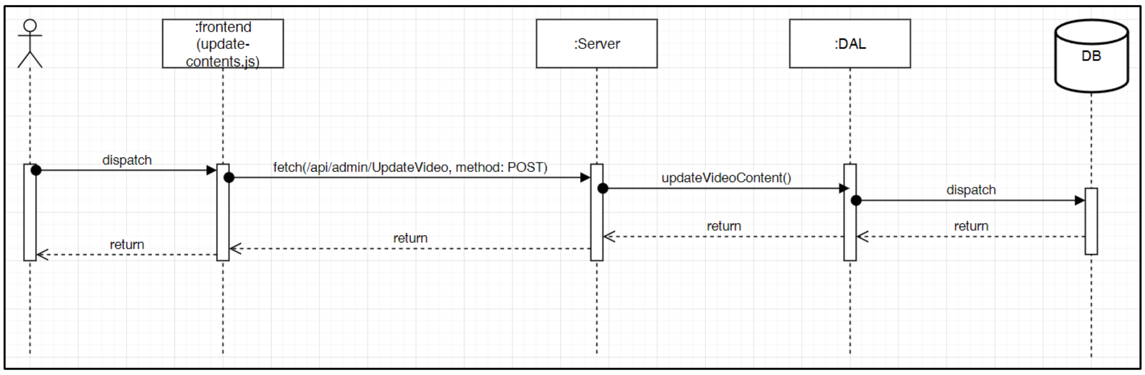
(a)



(b)

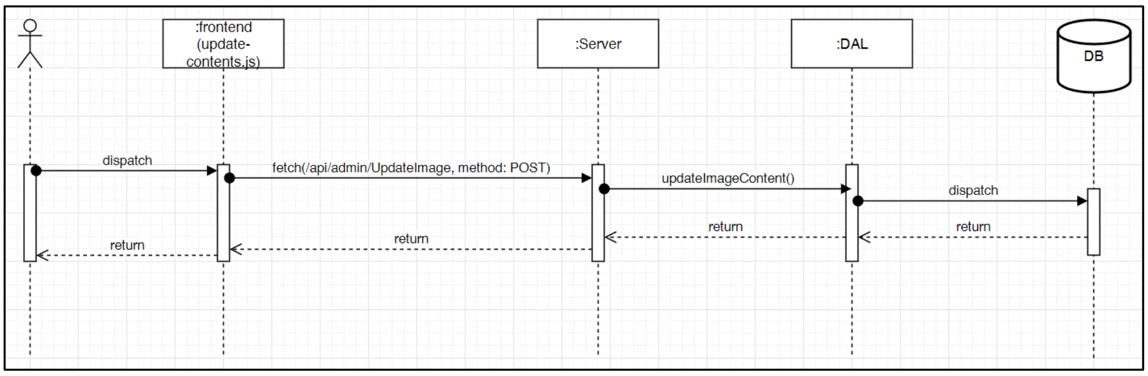
Figure 3. 5. 11: Sequence Diagram for Add Contents[15]
(a) videos, and (b) images.

[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.



:frontend (update-contents.js)
:Server
:DAL
DB
dispatch
fetch(/api/admin/UpdateVideo, method: POST)
updateVideoContent()
dispatch
return
return
return
return

(a)

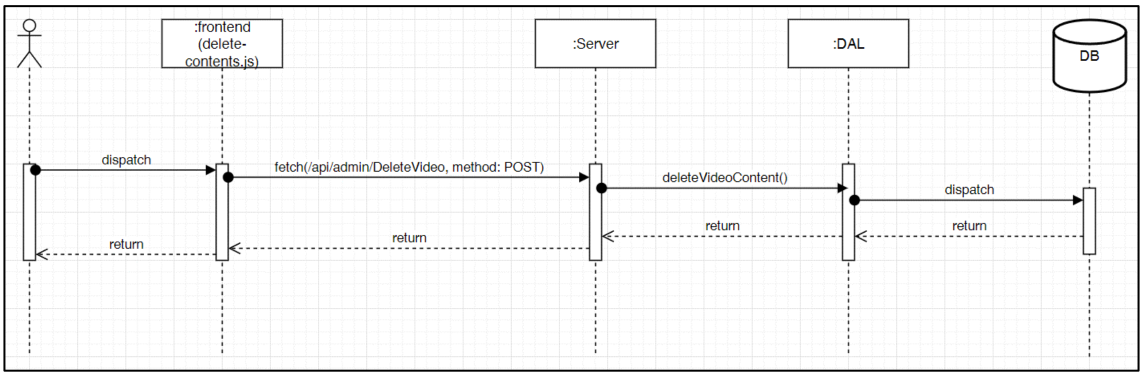


:frontend (update-contents.js)
:Server
:DAL
DB
dispatch
fetch(/api/admin/UpdateImage, method: POST)
updateImageContent()
dispatch
return
return
return
return

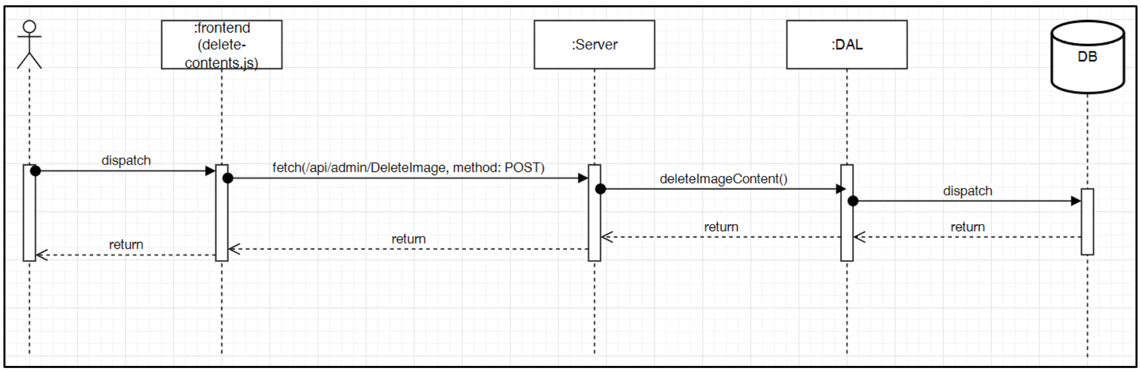
(b)

Figure 3.5.12: Sequence Diagram for Update Contents[16]
(a) videos, and (b) images.

[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.


:frontend (delete-contents.js)
:Server
:DAL
DB
dispatch
fetch(/api/admin/DeleteVideo, method: POST)
deleteVideoContent()
dispatch
return
return
return
return

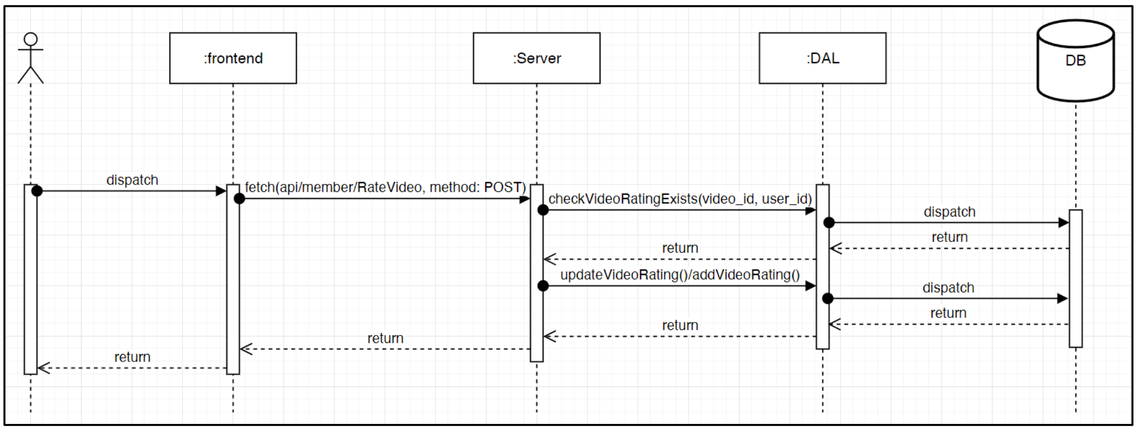
(a)


:frontend (delete-contents.js)
:Server
:DAL
DB
dispatch
fetch(/api/admin/DeleteImage, method: POST)
deleteImageContent()
dispatch
return
return
return
return

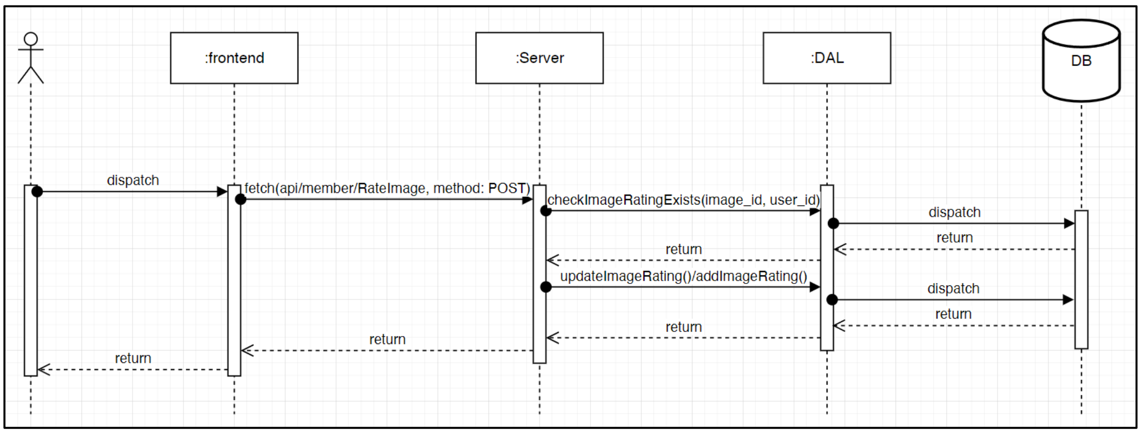
(b)

Figure 3. 5. 13: Sequence Diagram for Delete Contents[17]
(a) videos, and (b) images.

[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.


:frontend
:Server
:DAL
DB
dispatch
fetch(api/member/RateVideo, method: POST)
checkVideoRatingExists(video_id, user_id)
dispatch
return
return
updateVideoRating()/addVideoRating()
dispatch
return
return
return
return

(a)


:frontend
:Server
:DAL
DB
dispatch
fetch(api/member/RateImage, method: POST)
checkImageRatingExists(image_id, user_id)
dispatch
return
return
updateImageRating()/addImageRating()
dispatch
return
return
return
return

(b)

Figure 3. 5. 14: Sequence Diagram for Rating[18]
(a) videos, and (b) images.

[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.

3.6. Step 6: Best selection

Table iv: Comparison of the RMDBS[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

[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/

Table v: Comparison of the server's programing language [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

[20] https://learn.microsoft.com/en-us/dotnet/csharp/
https://developer.mozilla.org/en-US/docs/Web/JavaScript
https://www.python.org/doc/

Table vi: 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.

3.7. Step 7: Prototyping

Drawn by Goodnotes.

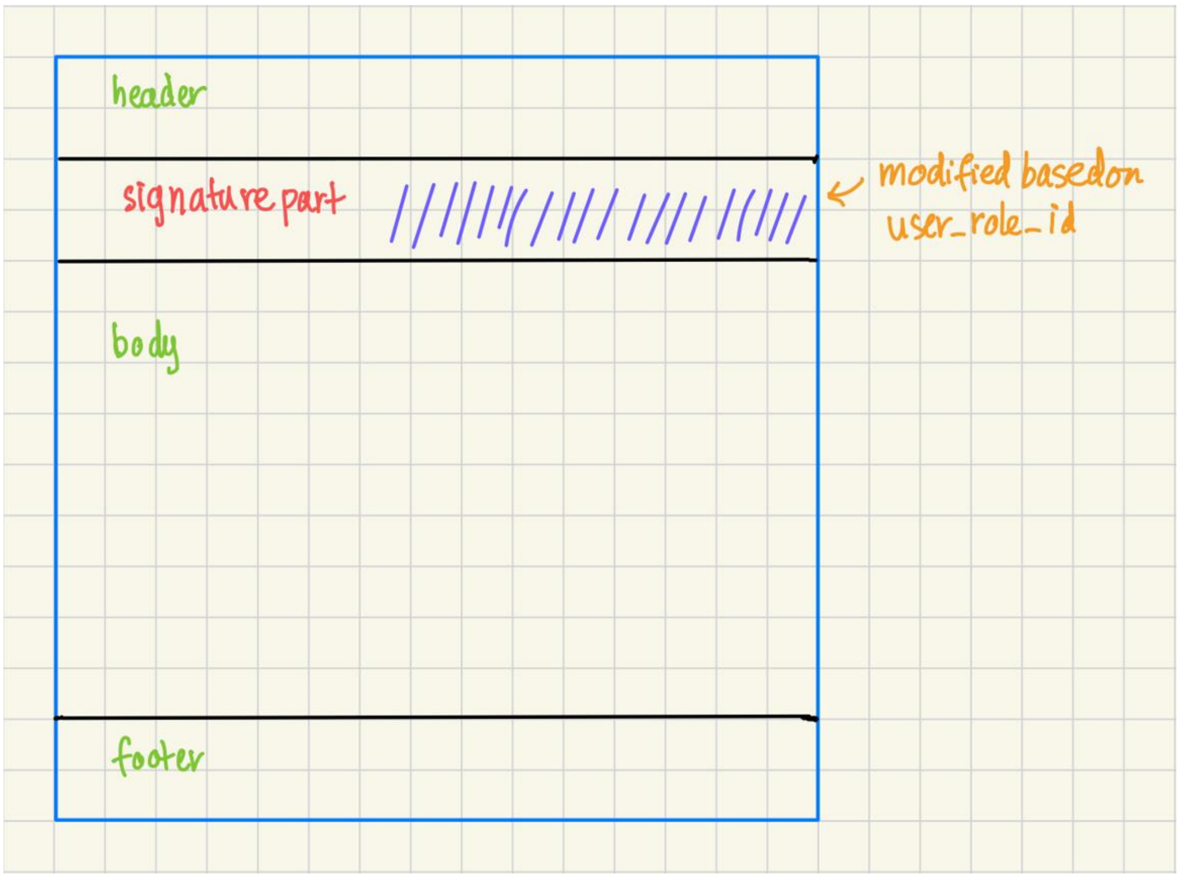


Figure 3.7.1: Preliminary Design for Main page

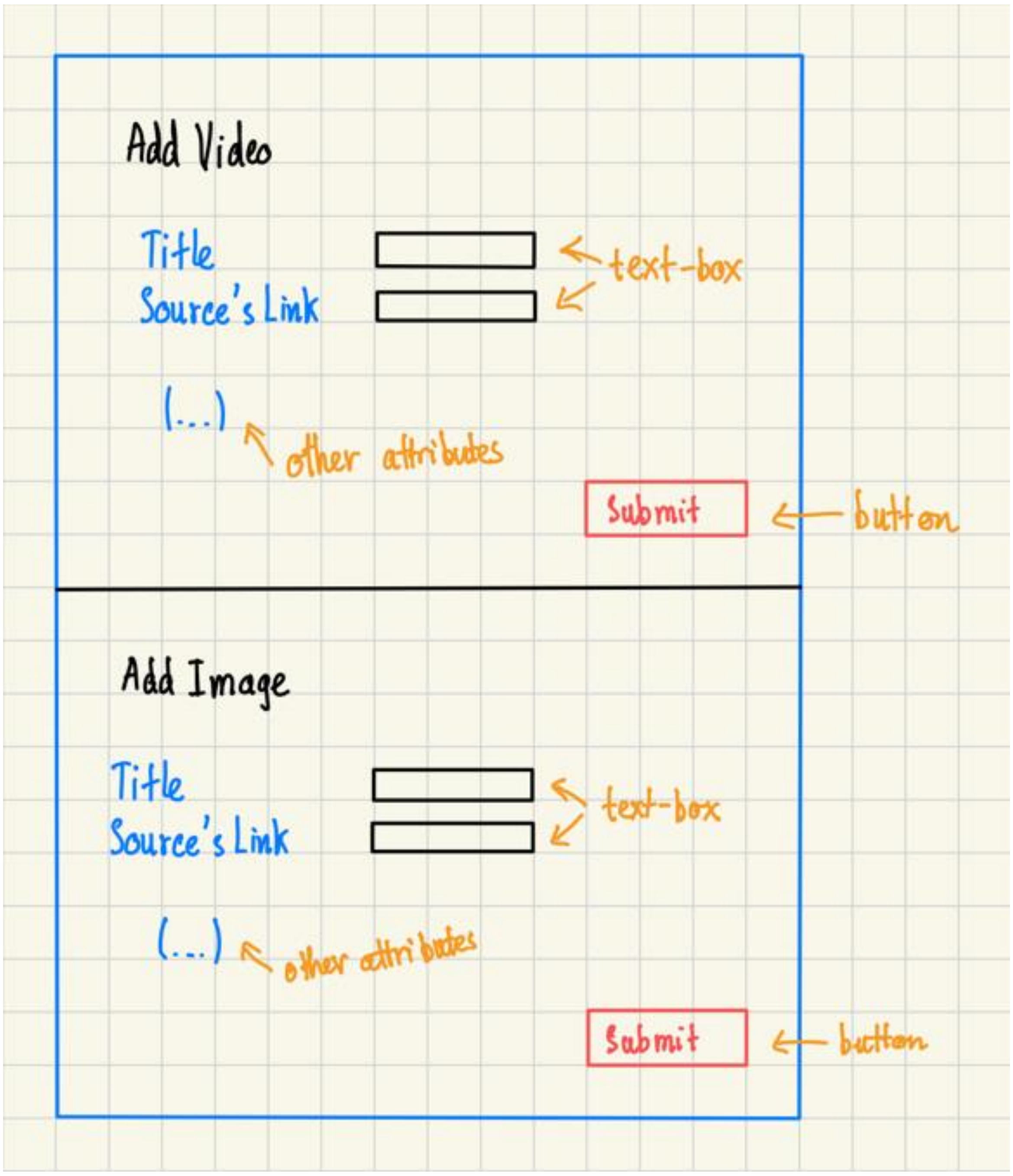


Figure 3.7.2: Preliminary Design for Add Contents site.

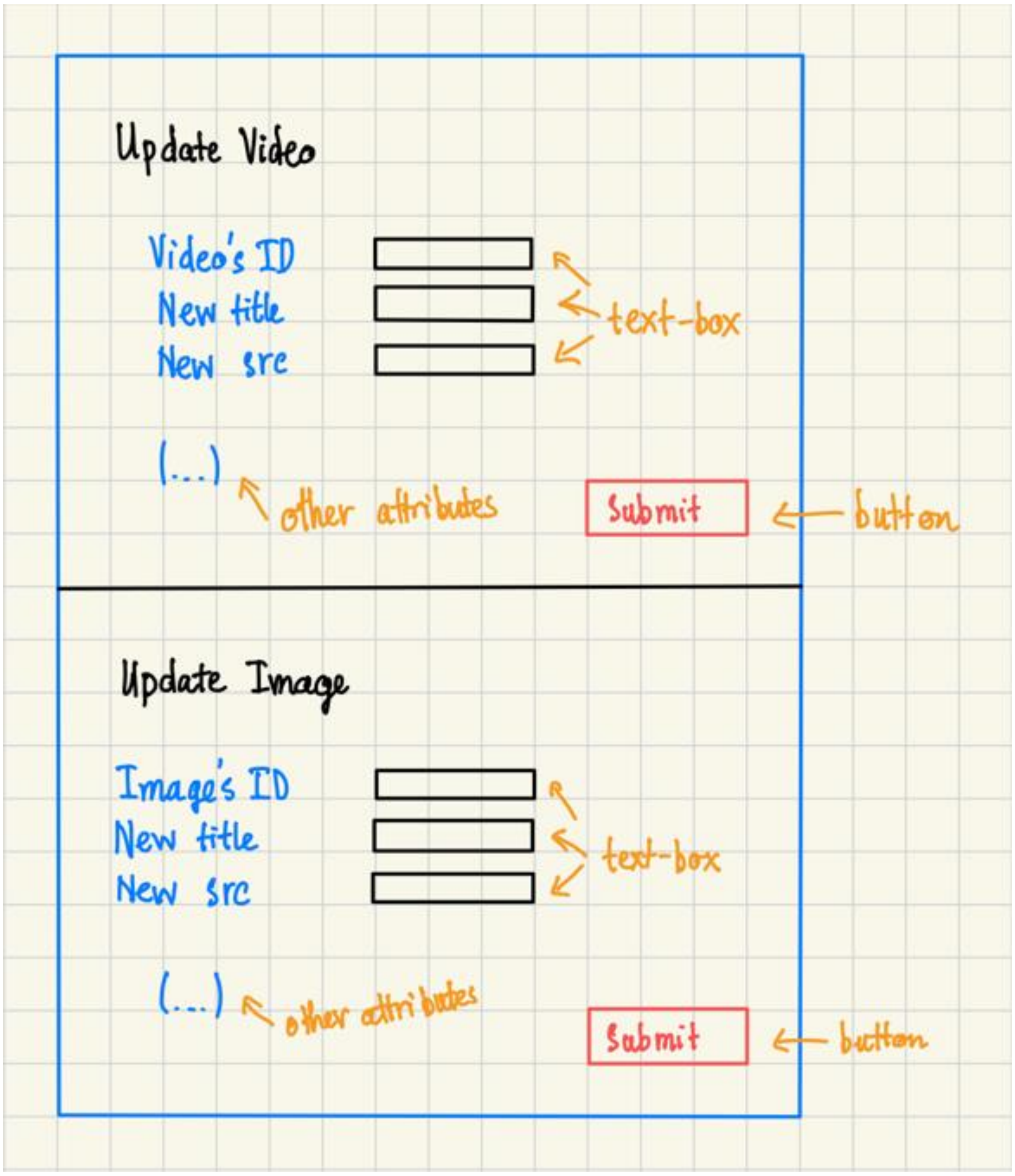


Figure 3.7.3: Preliminary Design for Update Contents site

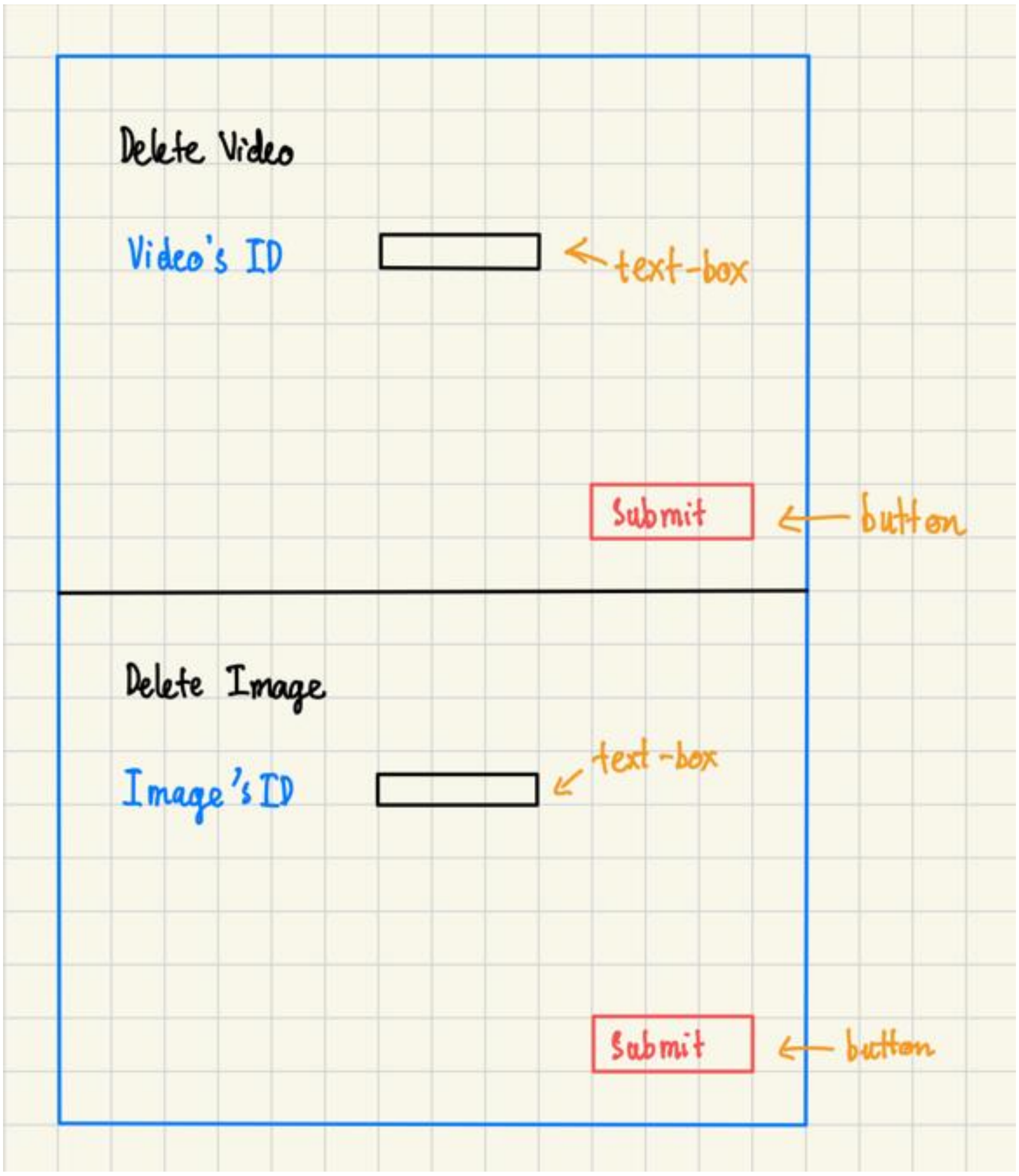


Figure 3.7.4: Preliminary Design for Delete Contents site.

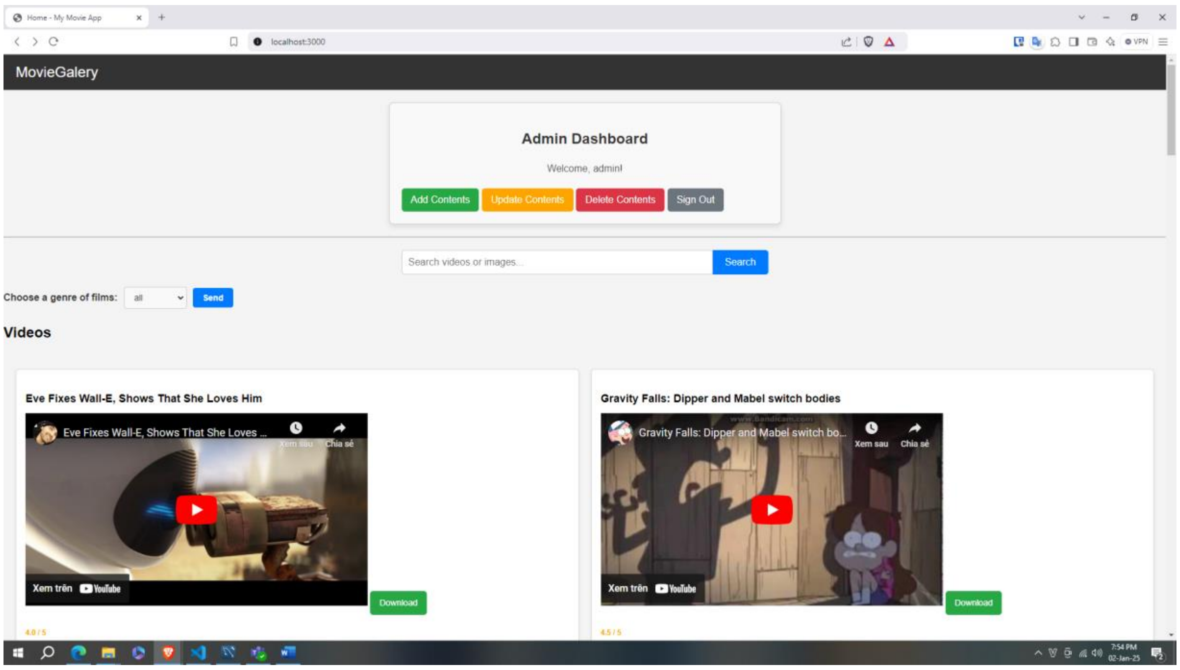


Figure 3.7.5: Final look of the Main page

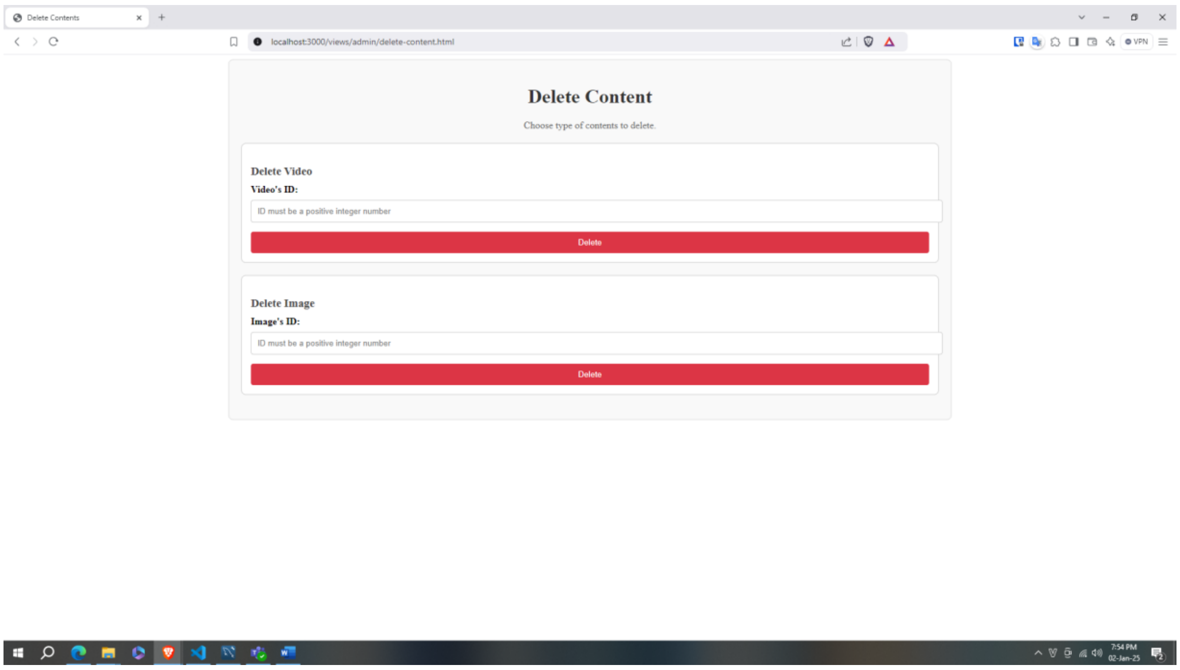

Delete Contents
localhost:3000/views/admin/delete-content.html
Delete Content
Choose type of contents to delete.
Delete Video
Video's ID:
ID must be a positive integer number
Delete
Delete Image
Image's ID:
ID must be a positive integer number
Delete

Figure 3.7.6: Final look of the Delete Content site

3.8. Step 8: Testing

Table vii: 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: o Invalid data: ▪ Let the id be negative or non-existing. ▪ Let the links not to start with http:// or https:// o Valid data
Analyze the test result	- Result: o Invalid data: response the error status o Valid data: response the successful status - Comments: The program realizes the invalid data and response to the user

Table viii: Rating Testing

<u>Affective factors</u>	- 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.
<u>Scenario for testing</u>	- 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.
<u>Conducting</u>	- Data: o Invalid data: ▪ Let the id be negative or non-existing. ▪ Let the point be not the integer number from 0 to 5. o Valid data - At the client's site, try that whether the feature works if the user has not signed in.
<u>Analyze the test result</u>	- Result: o 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). o Valid data: response the successful status and change the average point for each video and image too. o If users have not signed in yet, the program show out the problem. - Comments: The program works successfully.

Table ix: Sign up Testing.

<u>Affective factors</u>	- The username must be non-existing
<u>Scenario for testing</u>	- 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.

<u>Conducting</u>	- Data:
	  o Invalid data: The existing username
	  o Valid data

<u>Analyze the test result</u>	- Result: o Invalid data: When testing the server independently, response the error status. o Valid data: response the successful - Comments: The program works successfully.

Table x: Sign in Testing

<u>Affective factors</u>	- The username must be existing
<u>Scenario for testing</u>	- 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.

<u>Conducting</u>	- Data:
	  o Invalid data: The non-existing username
	  o Valid data

<u>Analyze the test result</u>	- Result: o Invalid data: When testing the server independently, response the error status. o Valid data: response the successful - Comments: The program works successfully.

4. 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.

5. 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.

6. 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

i. Test result for Update Image (valid data)

Method: POST	url: http://localhost:3000/api/admin/UpdateImage

Input

```
{
    "id": 17[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
    }
}
```

[21] In this case, we insert an image whose id is 17 before, so it is valid.

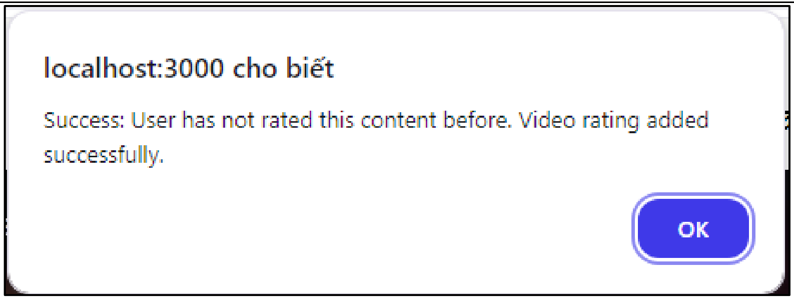
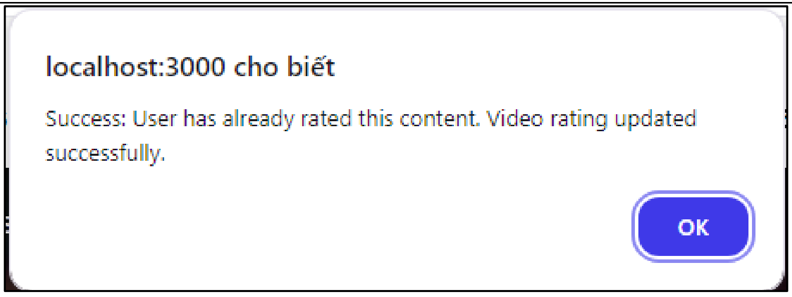
ii. Test result for Update Image (negative ID)

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"
	}
	```

iii. Test result for Rating

 localhost:3000 cho biết Success: User has not rated this content before. Video rating added successfully. 	The user has not rated the content before
 localhost:3000 cho biết Success: User has already rated this content. Video rating updated successfully. 	The user has already rated the content before