

---

**Functionality Description**

**For**

**[SP Movies 2.0]**

**Date: [05/02/2023]**

---

# Contents

<b>1</b>	<b>DOCUMENT MANAGEMENT.....</b>	<b>3</b>
<b>1.1</b>	<b><i>Contributors</i>.....</b>	<b>3</b>
<b>1.2</b>	<b><i>Version Control</i>.....</b>	<b>3</b>
<b>2</b>	<b>OVERVIEW .....</b>	<b>4</b>
<b>2.1</b>	<b>Function Description.....</b>	<b>4</b>
<b>2.2</b>	<b>Technology .....</b>	<b>15</b>
<b>2.3</b>	<b>Development Tools and Processes .....</b>	<b>15</b>
<b>2.4</b>	<b>Interfaces and services.....</b>	<b>17</b>
<b>3</b>	<b>APPENDIX .....</b>	<b>19</b>
<b>3.1</b>	<b>Advanced / Additional features .....</b>	<b>19</b>
<b>3.2</b>	<b>Reference Links.....</b>	<b>21</b>
<b>3.3</b>	<b>Credits .....</b>	<b>21</b>

# 1 Document Management

## 1.1 Contributors

*Details of all contributors to this document*

Function Integrated	Name
Functions #1,2,3	Ng Cheng Wai Melvin
Function #4	Kua Zi Lin
Function #5	Ng Chye Yong
Functions #6, #7	Ng Cheng Wai Melvin
Advanced/Additional functionalities	Ng Cheng Wai Melvin

## 1.2 Version Control

*Changes made to this document since creation.*

Date	Version	Author	Section	Amendment
01/02/23	1.0	Melvin	-	Initial Creation
03/02/23	1.1	Kua	2.1	Update Function 4 Function Description
03/02/23	1.2	NG CY	2.1	Update Function 5 Function Description
04/02/23	1.3	Kua	2.4	Update Function 4 Interface & Service
04/02/23	1.4	NG CY	2.4	Update Function 4 Interface & Service
04/02/23	1.5	Melvin	3.1, 3.2	Add new sections & Tidy up document
04/02/23	1.6	Melvin	2.1	Fix minor typos
05/02/23	1.7	Melvin	2.1	Update Functions 1-3 Function Description
05/02/23	1.8	Melvin	2.4	Update Functions 1-3 & 5 Interface & Service
05/02/23	1.9	Melvin	2.1	Update Functions 6-7 Function Description
05/02/23	2.0	Melvin	2.4	Update Functions 6-7 Interface & Service
05/02/23	2.1	Melvin	2.2, 2.3	Update Technology section Update Development Tools and Processes section
05/02/23	2.2	Melvin	3.1, 3.3	Update Additional/Advanced features Add new section 3.3 (Credits)

## 2 OVERVIEW

A brief description of what the overall system is about from a business perspective.

### 2.1 Function Description

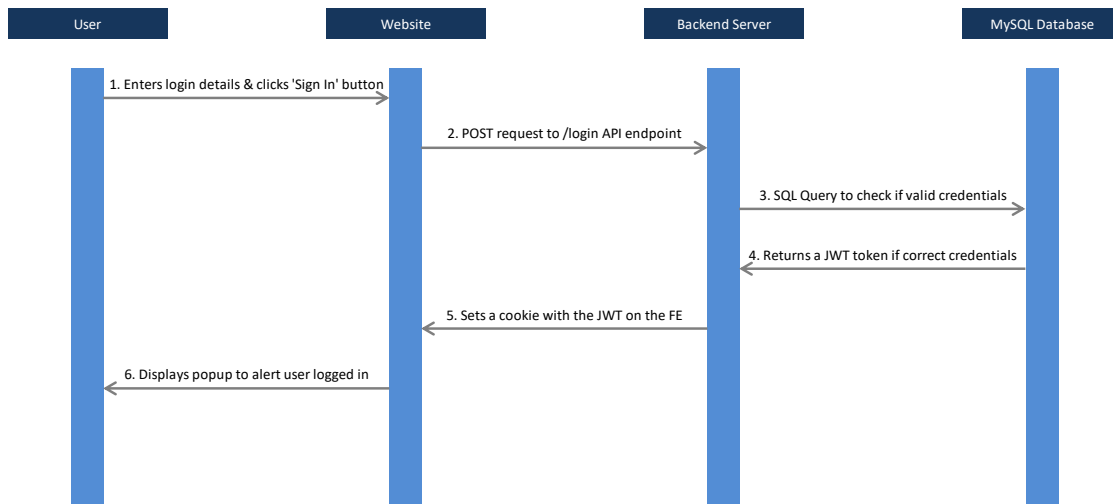
High level description of what functionality the system contains (a list of the components / functionalities that have been integrated (frontend ↔ backend)).

This should include

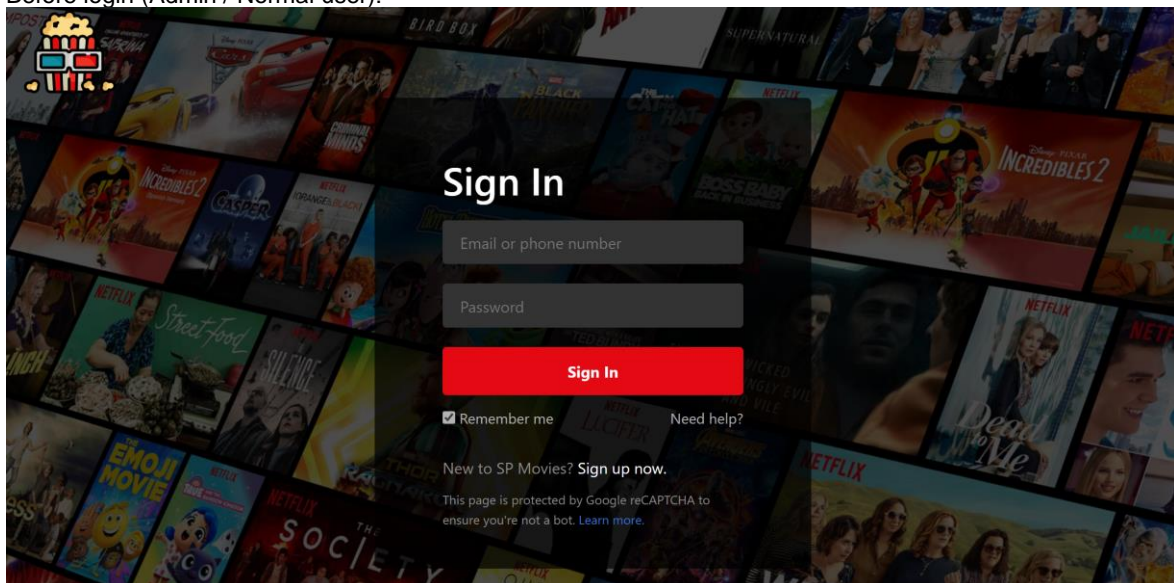
- i) a sequence diagram of the input / output between the User, Front End (i.e. the Website) and the Backend (i.e. the Server App).
- ii) screenshots of the end to end process (i.e. what would be observed by the User)

#### 1. Function #1 – Login and Access control (Admin user & Normal user)

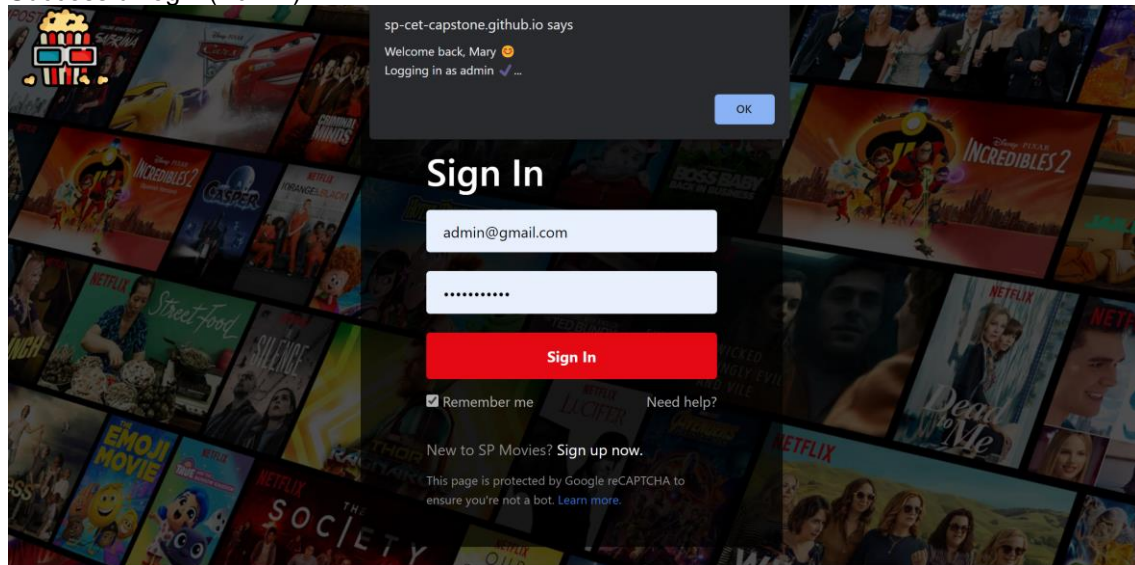
- When the web application is initially loaded, you will be initially shown a login screen. Using this particular login functionality, an admin or normal user will be able to login & if successful, they will be able to access the other web pages / functionalities of the web app. An error message will be shown instead if login fails.
- FYI:
  - o Admin user: Has ALL the privileges (able to access ALL functionalities of the application)
  - o Normal user: Has only LIMITED privileges (only able to access SOME functionalities)
  - o Not logged in admin/user: NOT ABLE to access any functionalities (only shown a login page by default)



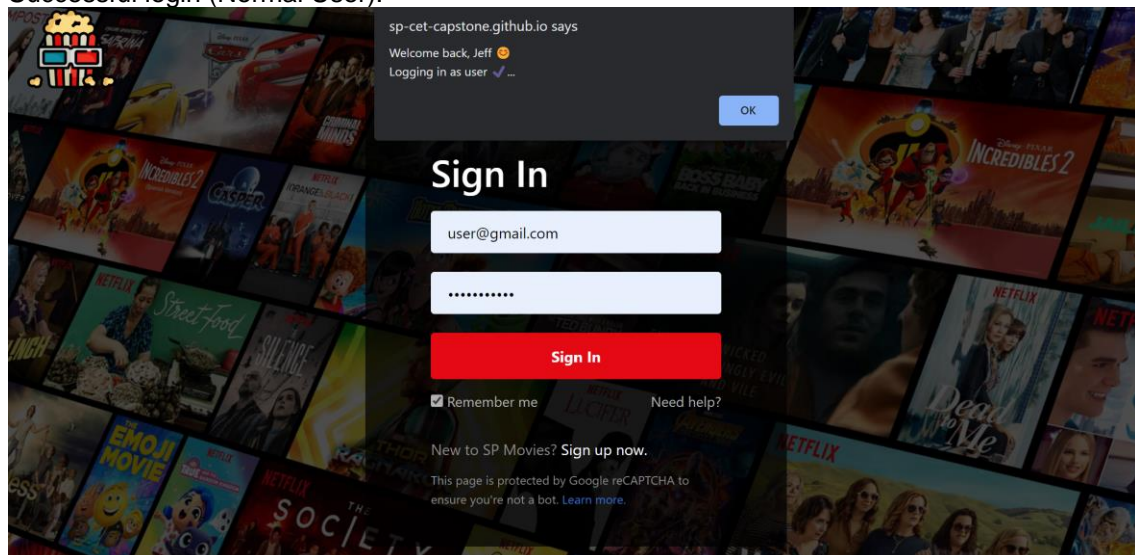
- Before login (Admin / Normal user):



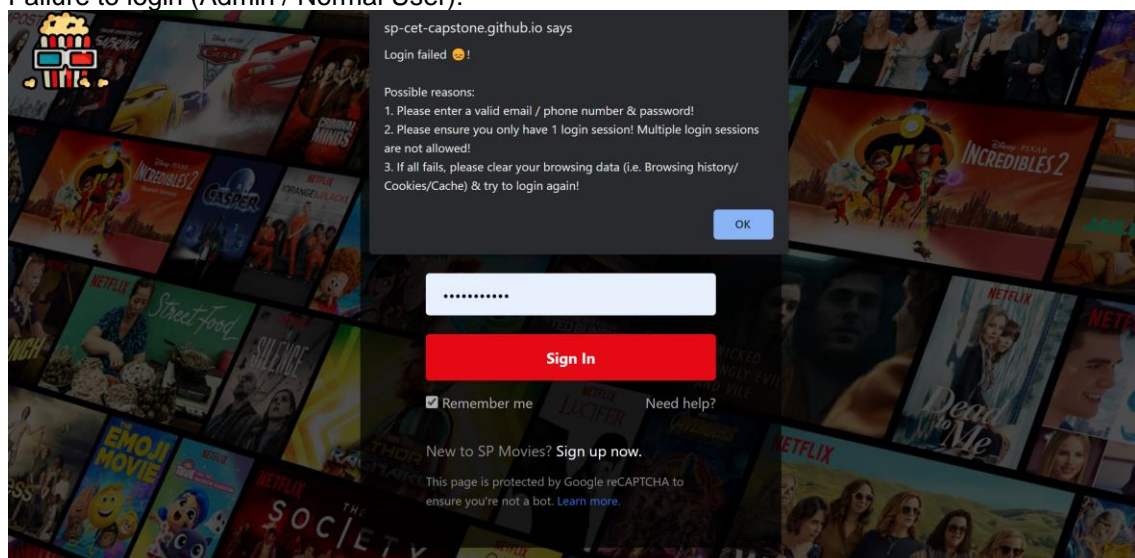
- Successful login (Admin):



- Successful login (Normal User):



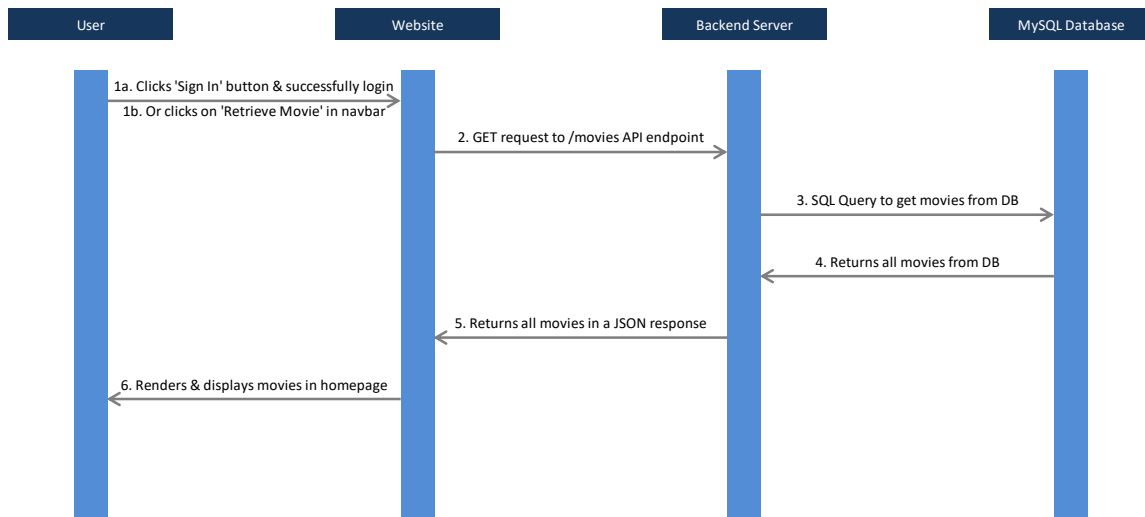
- Failure to login (Admin / Normal User):



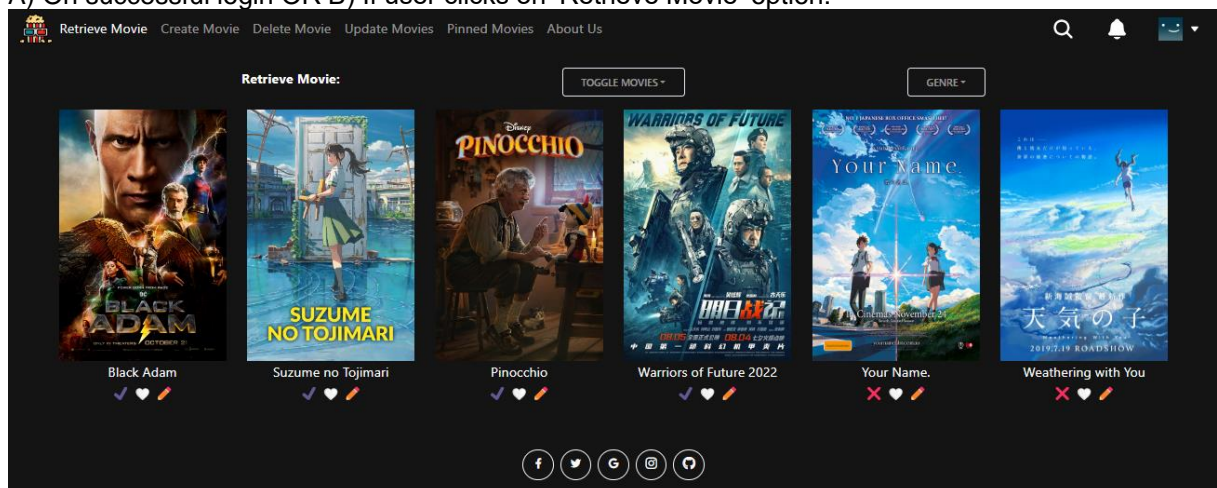


## 2. Function #2 – Retrieve and display a list of movies (Admin user & Normal user)

- Once the user is logged in, the user can use this functionality to retrieve the latest movies from the server / database and then display it. The order of the movies displayed is based on its own respective 'movieID' found in the 'movie' table of the database (ASC order).
- The user can also click on the 'Retrieve Movie' option in the navigation bar to execute the same functionality that fetches & displays a list of movies.



- A) On successful login OR B) If user clicks on 'Retrieve Movie' option:



- If the GET /movies API endpoint fails to retrieve & display a list of movies:

sp-cet-capstone.github.io says

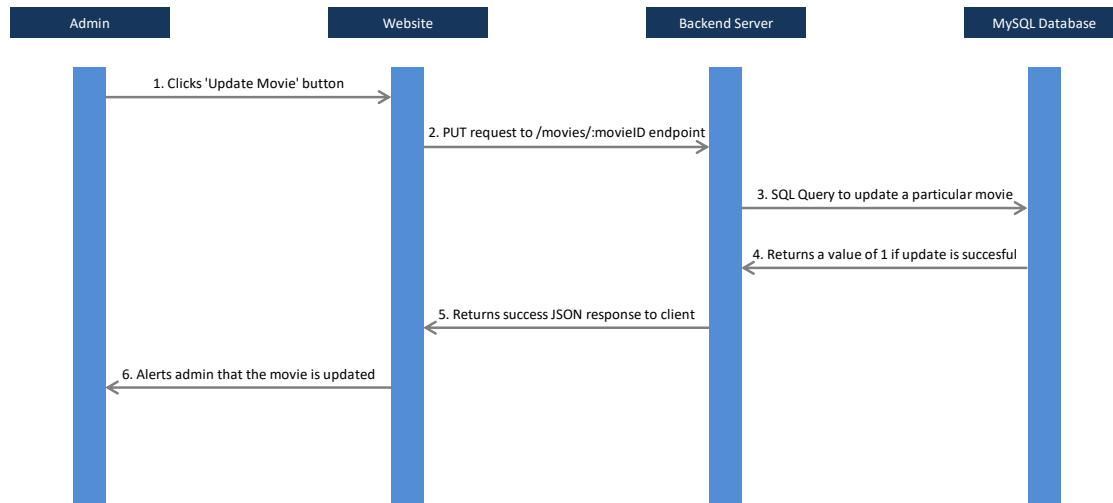
An error occurred while fetching the movies from the database!

Please try again later 😞

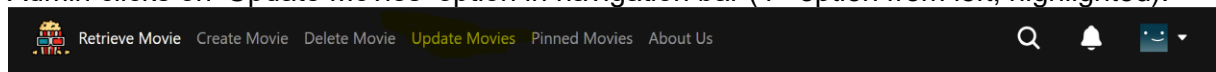
OK

### 3. Function #3 – Update Movie details (Only for Admin)

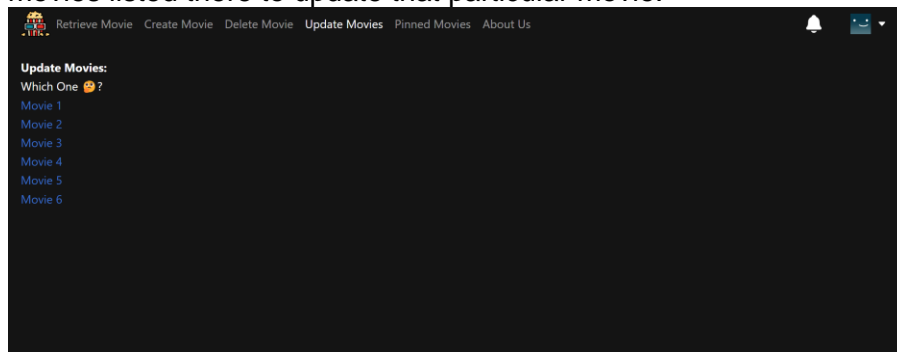
- Once the admin is logged in, the admin can use this function to update movie details for any particular movie.
- The updated movie details will be sent as a payload to the server and database for processing. If the update is successful, that particular movie will be updated in the database with the new movie details.
- When the user subsequently clicks on the 'Retrieve Movie' option in the navigation bar to go to the homepage, he/she will be able to see the updated movie details over there.



- Admin clicks on 'Update Movies' option in navigation bar (4<sup>th</sup> option from left; highlighted):

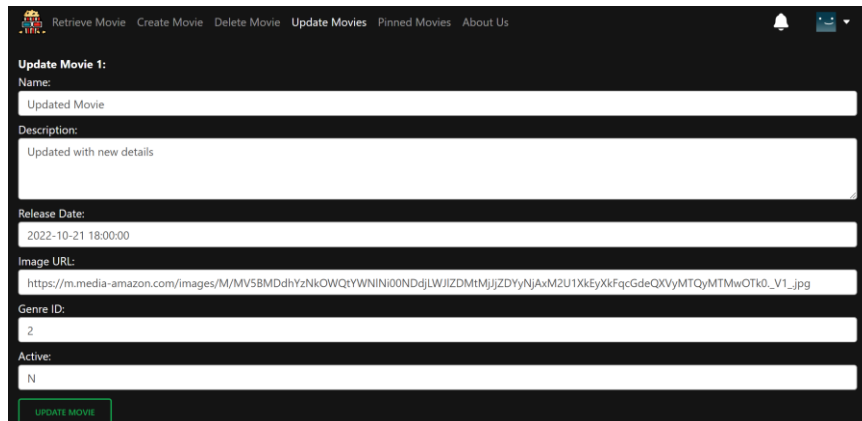


- The admin will then be directed to the 'Update Movies' webpage. Click on any of the movies listed there to update that particular movie:

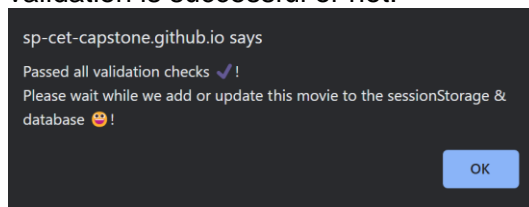


- For example, if we click on 'Movie 1' option (from the previous screen), we will be directed to a webpage that contains a form to update Movie 1 specifically:

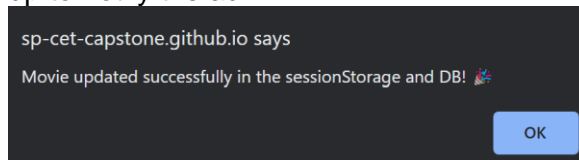
- Enter the new movie details you wish to update in the database. Once completed, press the 'Update Movie' button at the bottom of the form.



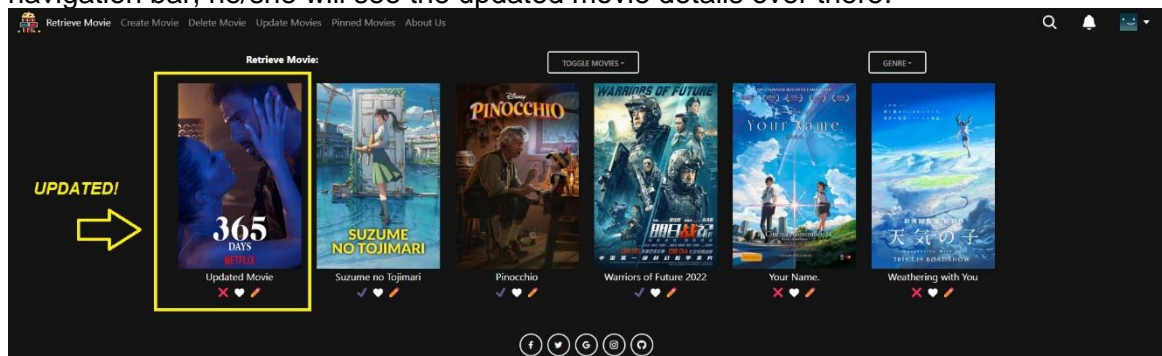
- The movie details payload will first be validated. Once validated, then we will send the payload to the server/database for processing. An alert message will first inform you if the validation is successful or not.



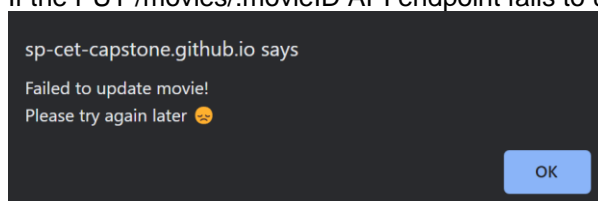
- Once the movie is successfully updated in the database, an alert message will then pop up to notify the admin.



- When the admin clicks on the 'Retrieve Movie' option or the SP Movies logo in the navigation bar, he/she will see the updated movie details over there:



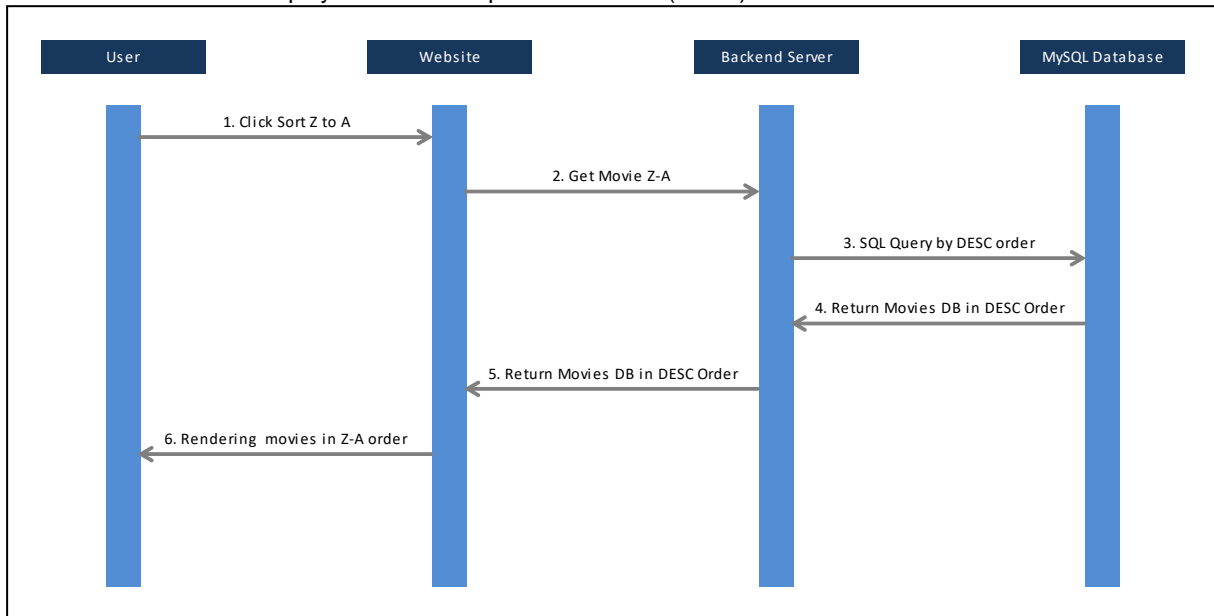
- If the PUT /movies/:movieID API endpoint fails to update the movie:



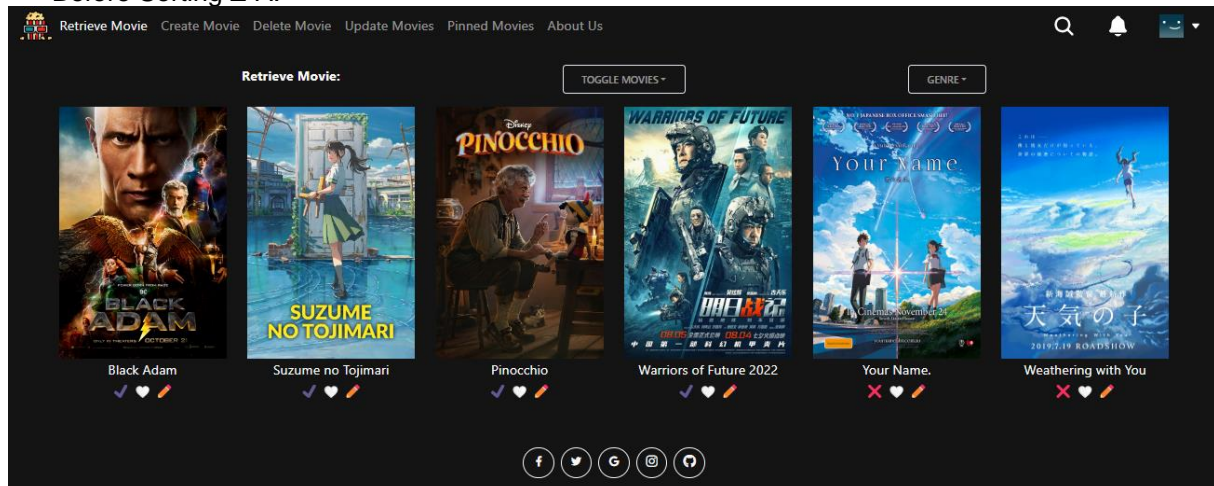


#### 4. Function #4 - Retrieve Movies and Sorting Z to A (Admin user & Normal user)

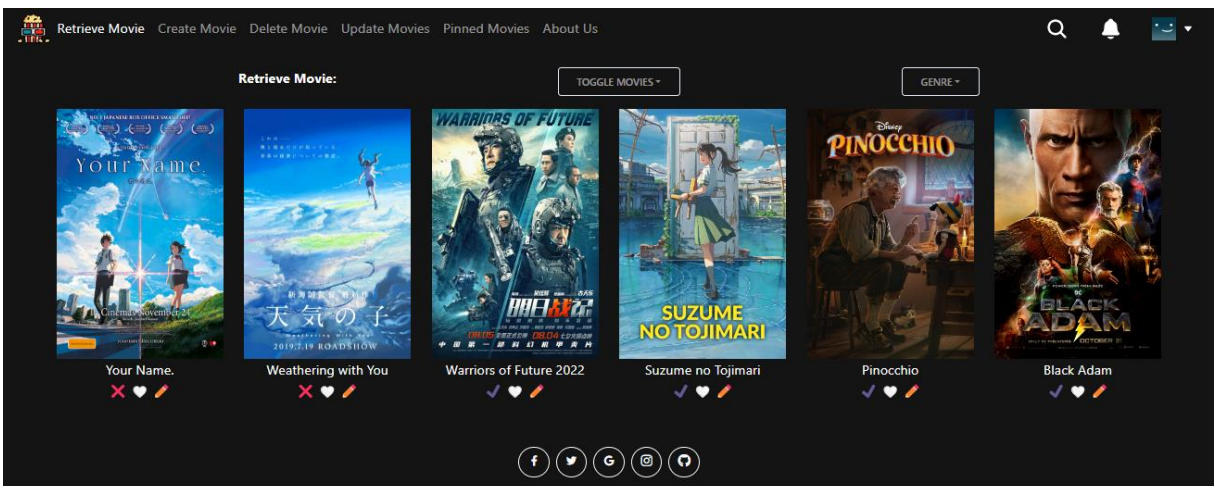
- Once the user is logged in, the user can use this function to retrieve the latest movies from the server / database and then display it in reverse alphabetical order (Z to A).



#### - Before Sorting Z-A:

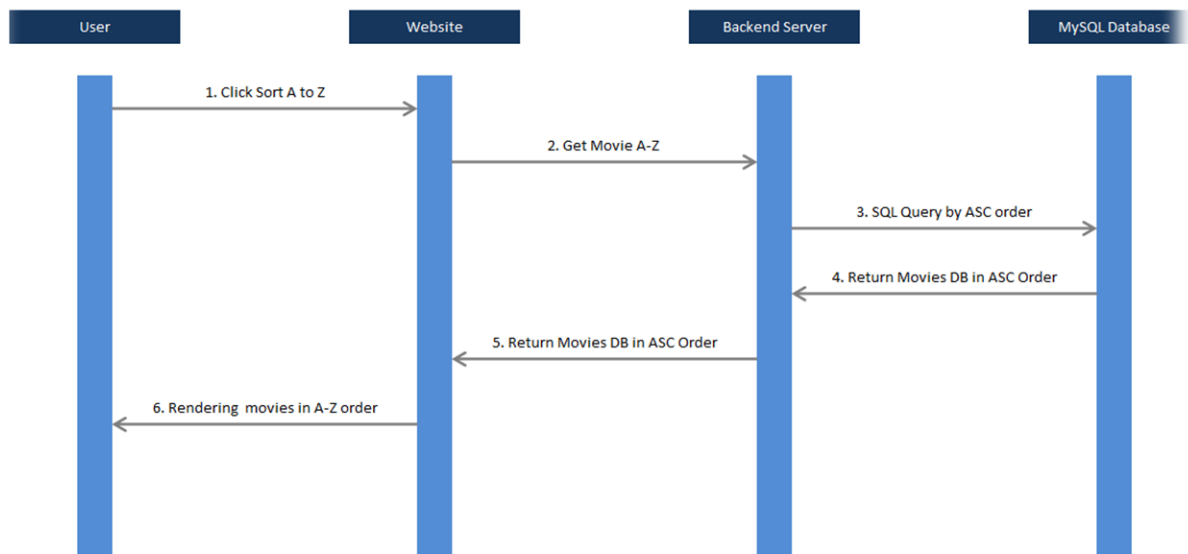


#### - After Sorting Z-A:

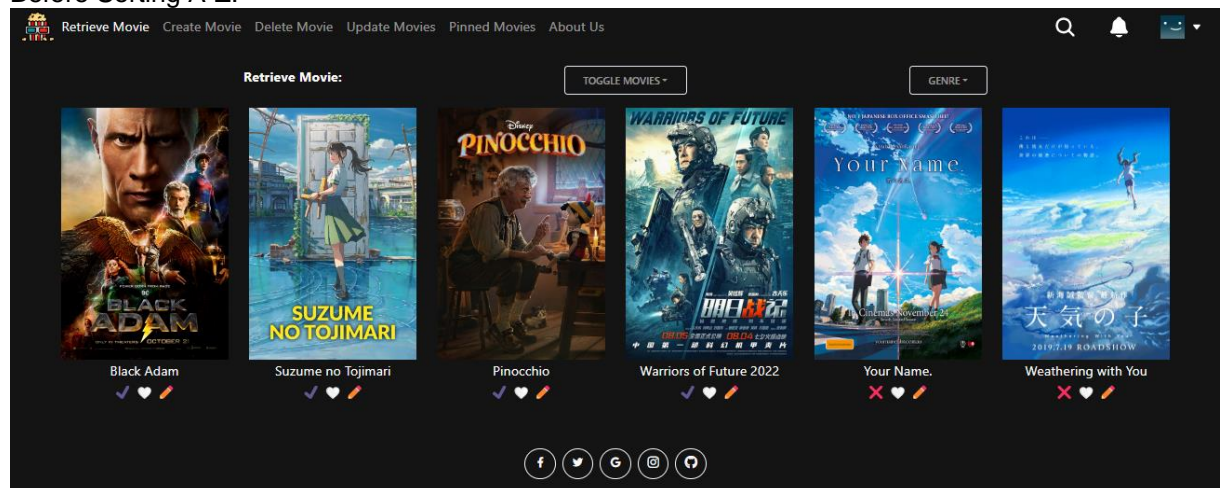


## 5. Function #5 - Retrieve Movies and Sorting A to Z (Admin user & Normal user)

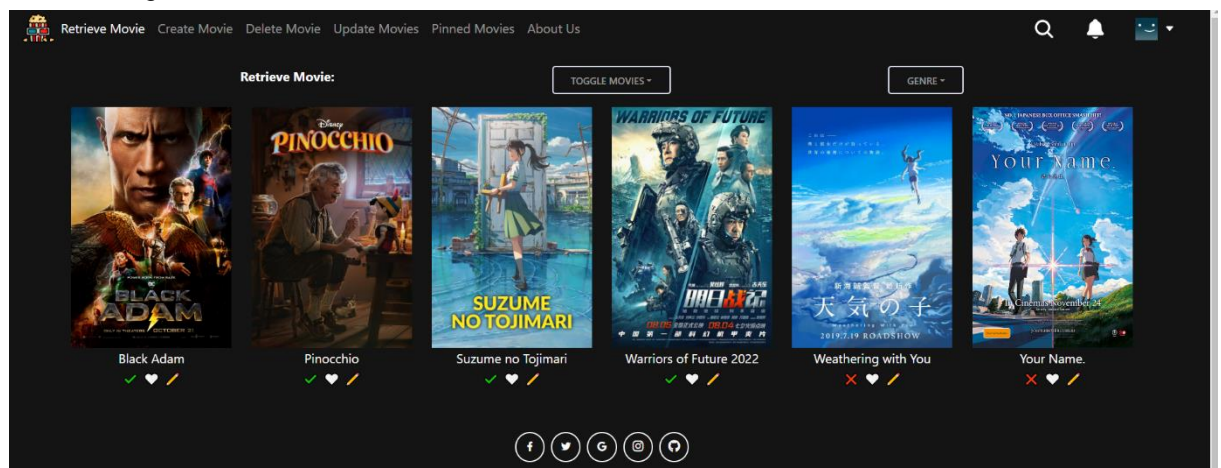
- Once the user is logged in, the user can use this function to retrieve the latest movies from the server / database and then display it in alphabetical order (A to Z).



- Before Sorting A-Z:

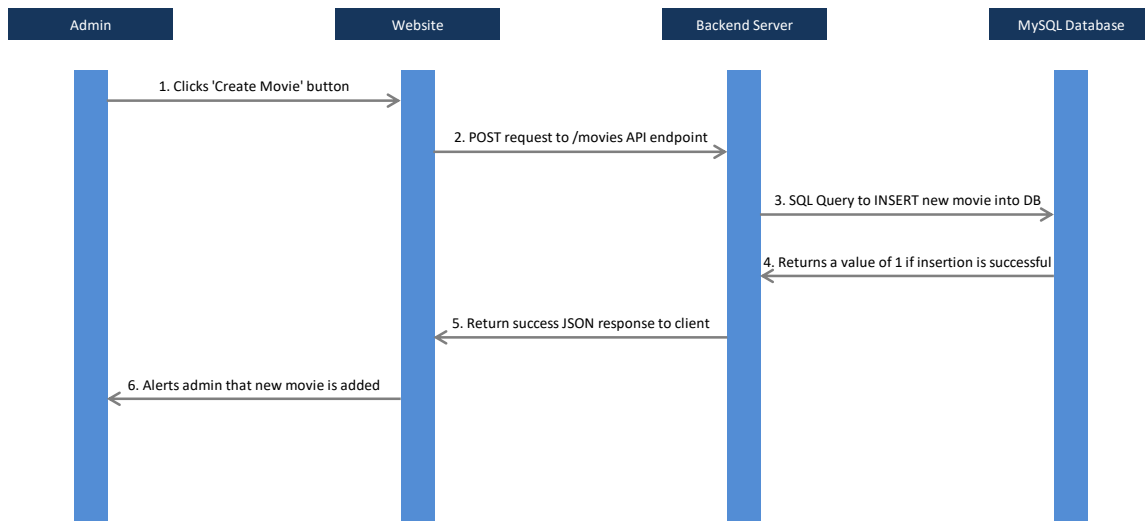


- After Sorting A-Z:



## 6. Function #6 – Create Movie (Only for Admin)

- Once the admin is logged in, the admin can use this functionality to create / add a movie to the database. If the insertion of the movie into the database is successful, he/she can then see the newly added movie in the 'Retrieve Movie' homepage.

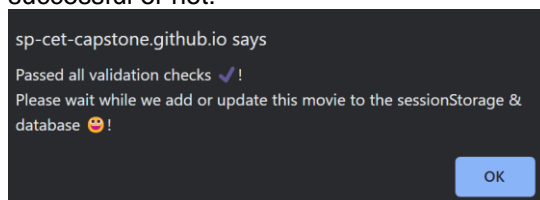


- Go to the 'Create Movie' page & enter your movie details. Once done, click 'Create Movie' button

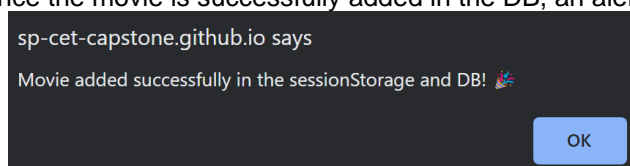
The screenshot shows the 'Create Movie' page with the following fields and values:

- Name:** New Movie
- Description:** New Movie Details
- Release Date:** 2022-12-15 00:00:00
- Image URL:** <https://media.istockphoto.com/id/1068817444/photo/cool-placeholder-for-your-picture-no-movie-screen-35mm-film-strip.jpg?s=170667a&w=0&k=20&c=IRwDC6j>
- Genre ID:** 1
- Active:** Y
- Button:** CREATE MOVIE

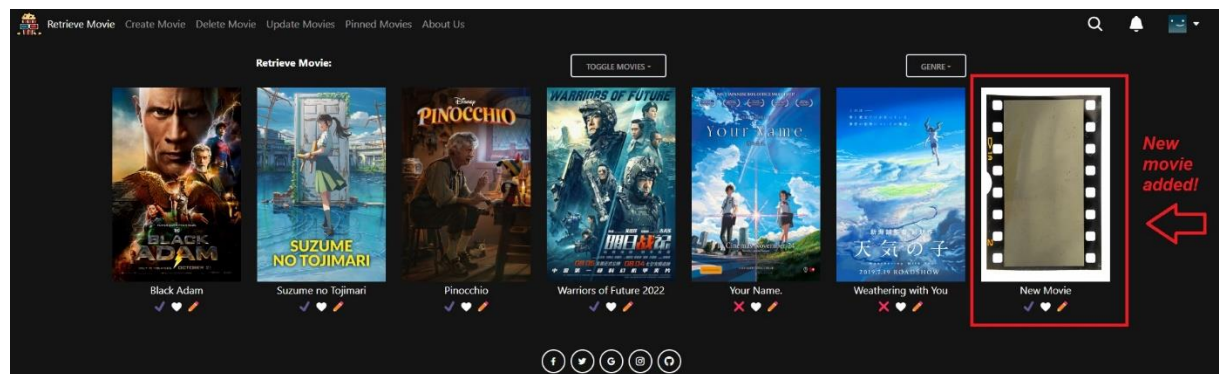
- The movie details payload will first be validated. Once validated, then we will send the payload to the server/database for processing. An alert message will first inform you if the validation is successful or not:



- Once the movie is successfully added in the DB, an alert message will pop up to notify the admin:

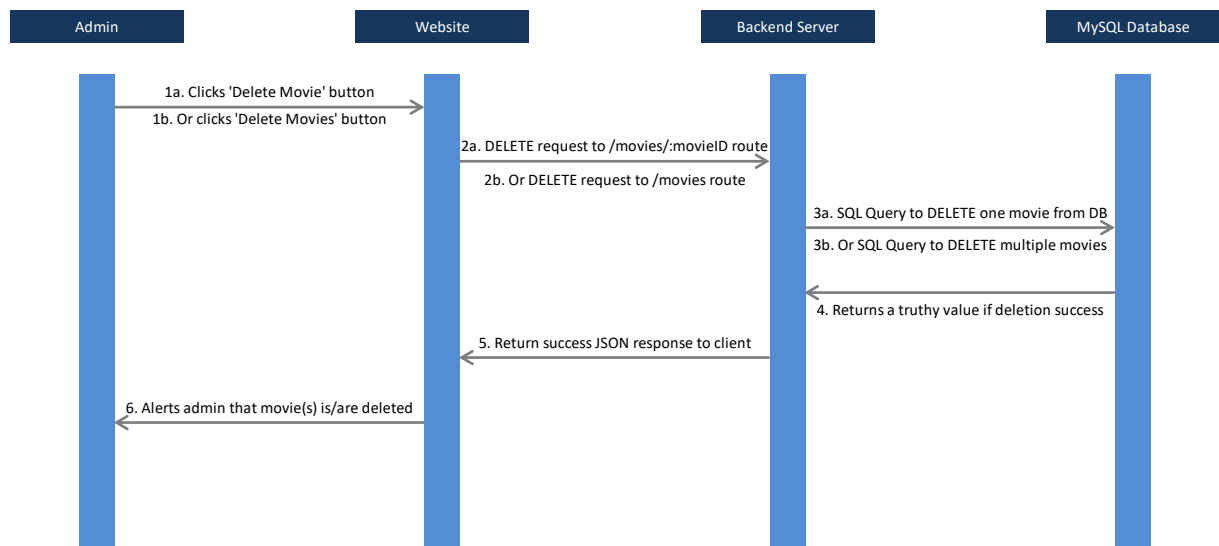


- When the admin clicks on the 'Retrieve Movie' option or the SP Movies logo in the navigation bar, he/she will see the newly added movie over there:

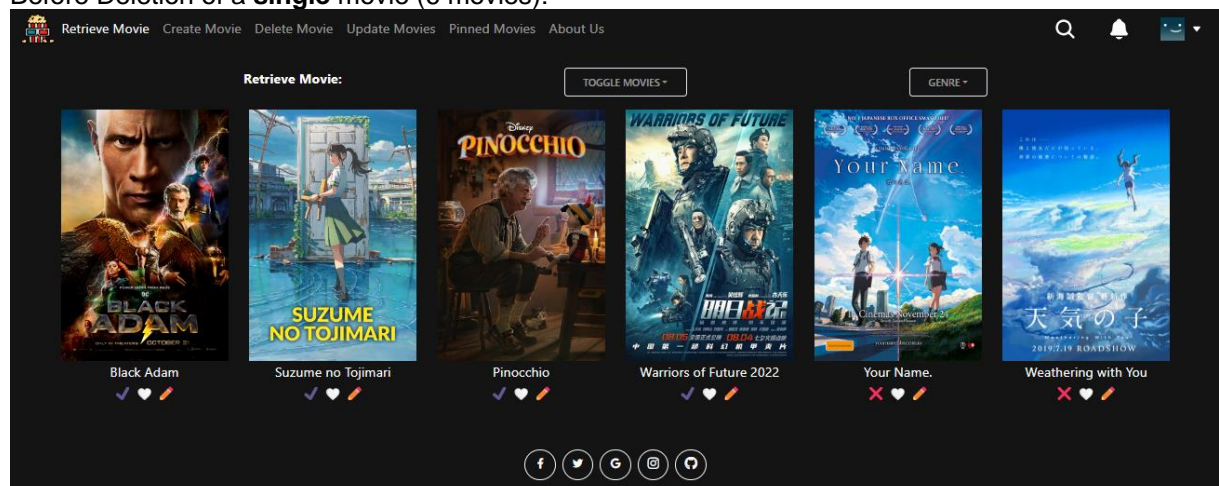


## 7. Function #7 – Delete One Movie or Delete Multiple Movies (Only for Admin)

- Once the admin is logged in, the admin can use this functionality to delete either a) ONE movie or b) MULTIPLE movies at a time from the database. If the deletion(s) is/are successful, the admin can navigate to the 'Retrieve Movies' homepage to see that the movie(s) have indeed been deleted.

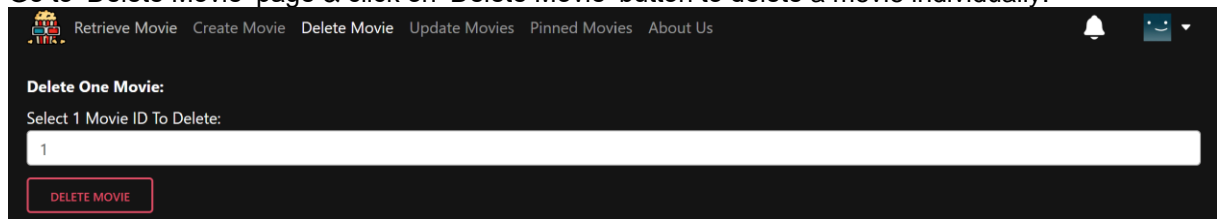


- Before Deletion of a **single** movie (6 movies):

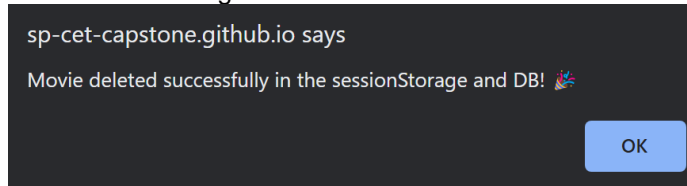




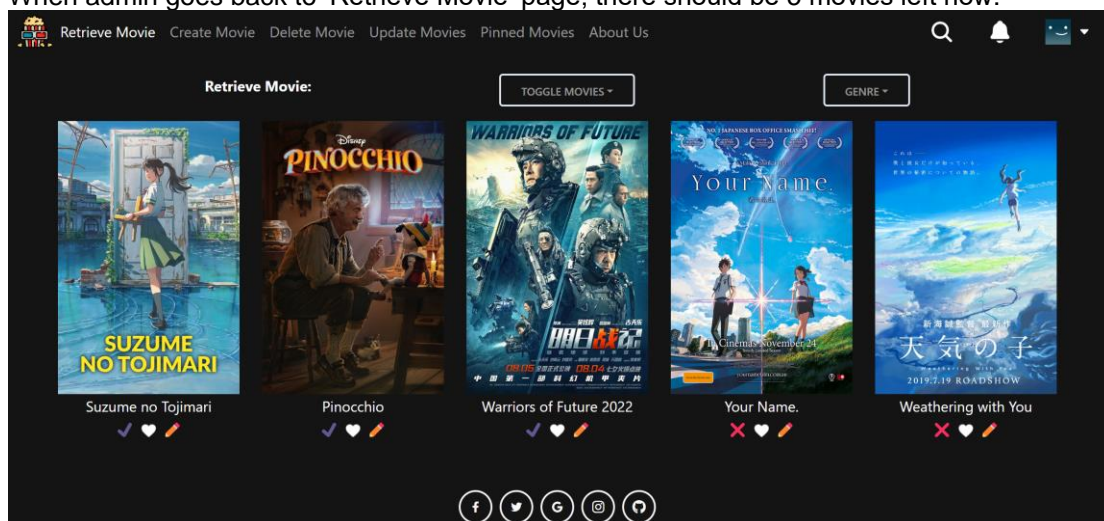
- Go to 'Delete Movie' page & click on 'Delete Movie' button to delete a movie individually:



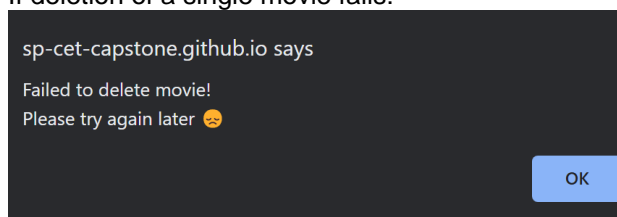
- If deletion of a single movie is successful:



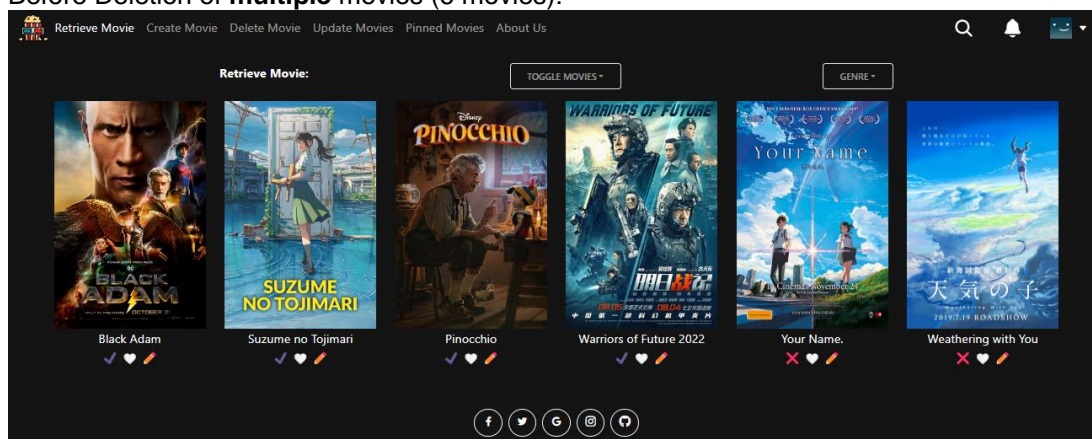
- When admin goes back to 'Retrieve Movie' page, there should be 5 movies left now:



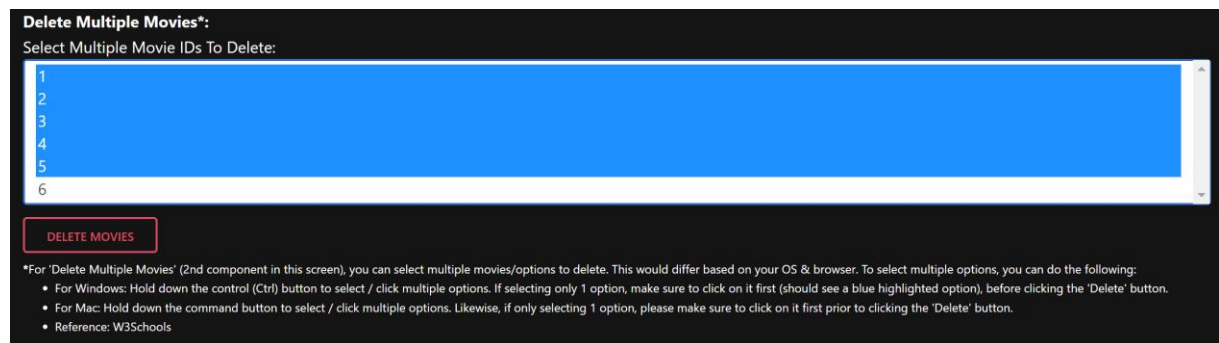
- If deletion of a single movie fails:



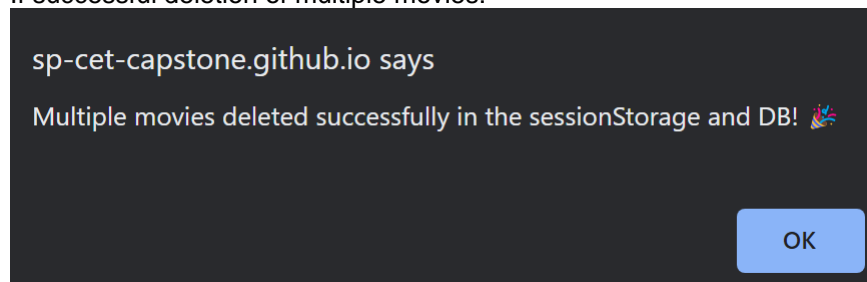
- Before Deletion of **multiple** movies (6 movies):



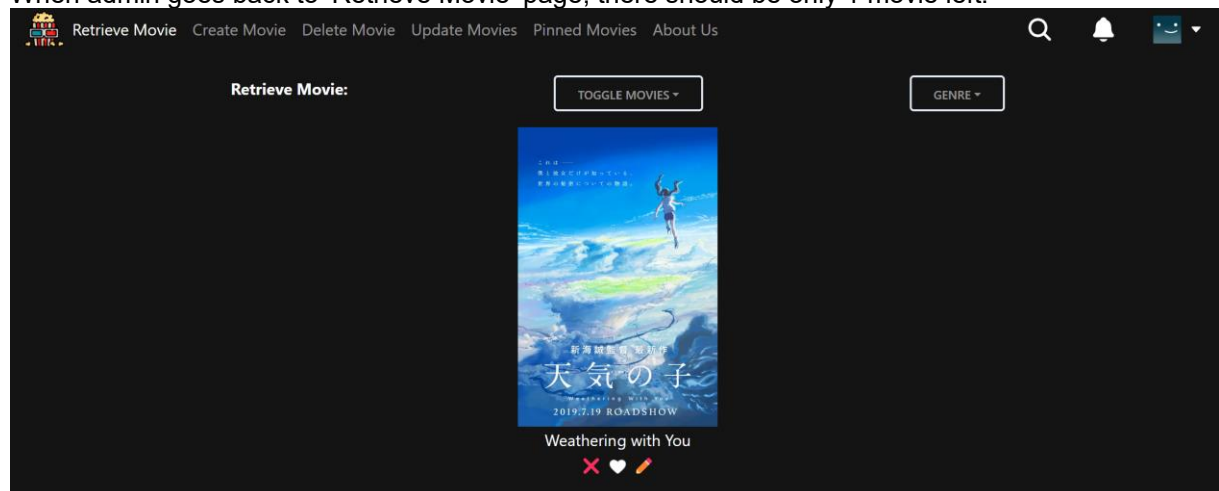
- Go to 'Delete Movie' page & click on 'Delete Movies' button to delete multiple movies at one go. We are deleting five movies (movies 1-5) for example:



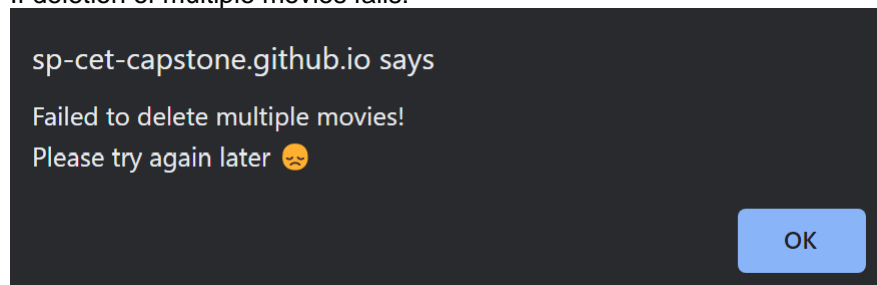
- If successful deletion of multiple movies:



- When admin goes back to 'Retrieve Movie' page, there should be only 1 movie left:



- If deletion of multiple movies fails:





## 2.2 Technology

*A listing and brief description of the underlying frameworks / libraries used by the application.*

### Frontend Tech Stack:

- HTML5, CSS3, JavaScript
- React.js, Redux, React-Router-DOM, Bootstrap 5, MDBootstrap 5, React-Icons, React-Spinners, Babel, Webpack, GitHub Pages (for hosting Frontend App)

### Backend Tech Stack:

- SQL
- Node.js, Express.js, MySQL, AWS (for hosting Backend App)

## 2.3 Development Tools and Processes

*This section should be used to document:*

- ☐ any specific development tools, compilers or suggested IDEs that are used for the application.
- ☐ Describe how software source control is performed (e.g. Git)

### Suggested Development Tools:

- Our team utilized the following development tools during the development of our application. Hence, we suggest the following development tools, should you wish to continue to work on developing the app even further:
  - o VSCode (suggested IDE)
  - o Live Server extension in VSCode (to start frontend app on <http://127.0.0.1:5500>)
  - o Nodemon (to start the backend app on <http://localhost:8081> – already integrated)
  - o Postman (if you wish to test/debug backend related API endpoints)
  - o Babel & Webpack (for compilation & bundling – already integrated into the project)
  - o GitHub Pages (for FE hosting)
  - o AWS (for BE hosting)
  - o Pls refer to this README.md before you develop/test this project. It contains numerous important details about the project & various debugging issues/solutions: <https://github.com/SP-CET-Capstone/capstone-assignment-class-1-team-1/blob/main/README.md>

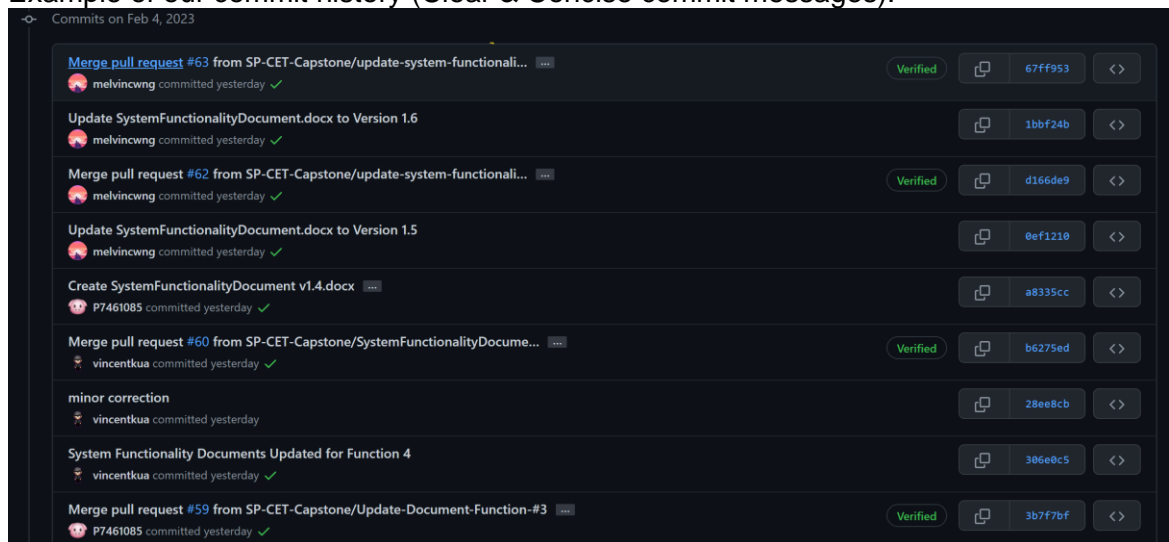
### How software source control is performed by our team:

- Our team understands the importance of source control/version control when working on software projects. Using source control, it is easy for team members to synchronize & get the latest updates to the codebase.
- Also, source control keeps things traceable – we will know what was changed, who made the change, when was the change made, and why changes were made – thus keeping things accountable in the software team.
- Source control also helps in conflict detection as well, and we can even “rollback” to a previous version if necessary (e.g. critical bugs in production → no choice but to rollback to previous working state).
- For this project, our team used Git & GitHub to perform source control.
- A GitHub repository was created & all the team members were added as collaborators.
- Some of us used the GitHub Desktop application to clone the repository to our local machines, while others used the command line to do clone the repository.
- Each of us ensured that we create a new branch for each feature that we were working on. In our local branch, we made sure to have small & meaningful commits.
- Once we were done with the feature, changes were committed to the branch and we pushed it to the remote repository.
- We then created a pull request (PR) & waited for the PR to be approved first, before merging the branch with the main branch. PRs have descriptions to indicate to the reviewer what were the main changes in the PR.
- The team-lead Melvin was the main reviewer & approver for the pull requests. For PRs with small changes, Zi Lin & Chye Yong also helped to review & approve them as well.

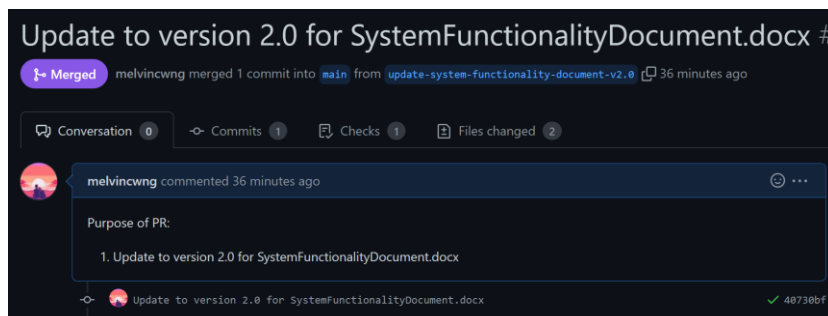
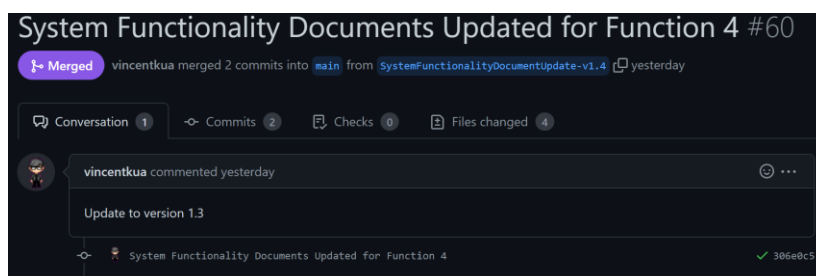
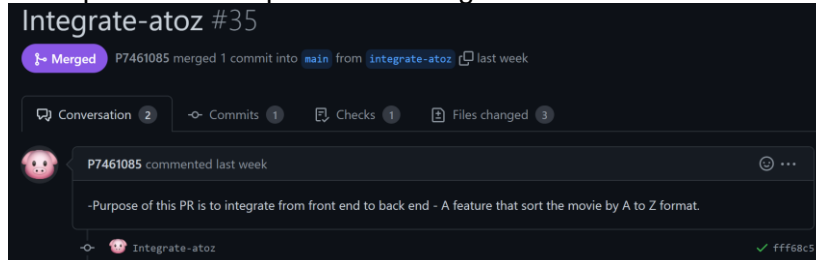
- To manage & track the progress of the project, we also used GitHub Projects - <https://github.com/orgs/SP-CET-Capstone/projects/11>
- The team-lead (Melvin) created the Kanban board, added and assigned all the tasks that needed to be done, to the respective team member.
- Each team member was responsible for handling the task that was assigned to them, and we regularly had calls/meetings/discussions ("standup"-like style) to indicate to one another on the progress of our assigned tasks.

### Screenshots indicating our team followed good source/version control:

- Example of our commit history (Clear & Concise commit messages):



- Examples of Pull Requests indicating a clear reason what the PR was about:



## 2.4 Interfaces and services

**Note:** As the group project codebase is relatively huge, we have listed only the main details of the front-end & back-end codes. Additional details can & will be elaborated during the project interview.

Function Integrated	Details of Front End Codes		Details of Back End Codes
Function #1 (Login & Access control)	src/components/LoginPage/index.jsx (lines 64-105, 157)	<->	controller/app.js (lines 527-559)
	src/components/RootPage/Index.jsx (lines 55, 243-302)		model/user.js (lines 151-238)
	src/components/RetrieveMoviePage/index.jsx (everything – all lines of code)		
Function #2 (Retrieve & display a list of movies)  1 <sup>st</sup> half (top) – Retrieve & display movies upon successful login  2 <sup>nd</sup> half (bottom) – Retrieve & display movies when clicking on 'Retrieve Movie' in navbar OR when clicking 'Show All Movies' option in the Toggle Movies button	src/redux/movieSlice.js (lines 15-31, 36)	<->	controller/app.js (lines 75-114)
	src/components/RootPage/Index.jsx (lines 79-81, 249, 259-260)		model/movie.js (lines 50-72)
	src/components/RetrieveMoviePage/index.jsx (everything – all lines of code)		
	src/utils/functions.js (lines 381-403)	<->	controller/app.js (lines 75-114)
	src/components/RootPage/index.jsx (lines 19-24, 90-92, 110-112, 184-190, 251-255, 259-260)		model/movie.js (lines 50-72)
	src/components/NavBar/index.jsx (lines 22-24, 66, 94)		
	src/redux/movieSlice.js (lines 88-90, 139-141)		
	src/components/RetrieveMoviePage/index.jsx (everything – all lines of code)		
Function #3 (Update movie details)	src/components/UpdateIndividualMoviePage/index.jsx (lines 37-66, 68-92, 141, 215-217)	<->	controller/app.js (lines 231-271)
	src/utils/functions.js (lines 72-192)		model/movie.js (lines 241-350)
	src/components/RootPage/index.jsx (lines 65, 110-112, 184, 188-189, 251-255, 259-260, 279-288)		
Function #4 (Retrieve & Sorting Z to A)	src/components/RootPage/Index.jsx (lines 19-24, 96-98, 116-118)	<->	controller/app.js (lines 183-193)
	src/redux/movieSlice.js (lines 62-79, 112-128)		model/movie.js (lines 180-202)
	src/utils/functions.js (lines 439-461)		
Function #5 (Retrieve & Sorting A to Z)	src/components/RootPage/Index.jsx (lines 19-24, 93-95, 113-115)	<->	controller/App.js (lines 171-181)
	src/redux/movieSlice.js (lines 44-61, 95-111)		model/movie.js (lines 157-179)
	src/utils/functions.js (lines 410-432)		

Function Integrated	Details of Front End Codes		Details of Back End Codes
Function #6 (Create Movie)	src/components/CreateMoviePage/index.jsx (lines 15-46, 48-72, 131-137)	<->	controller/app.js (lines 195-229)
	src/utis/functions.js (lines 72-192)		model/movie.js (lines 203-240)
	src/components/RootPage/index.jsx (lines 60, 110-112, 184-185, 189, 251-255, 259-269)		
Function #7 (Delete One Movie or Delete Multiple Movies)  1 <sup>st</sup> half (top) – Delete One Movie  2 <sup>nd</sup> half (bottom) – Delete Multiple Movies	src/components/DeleteMoviePage/index.jsx (lines 32-36, 38-42)	<->	controller/app.js (lines 273-292)
	src/components/DeleteMovieForm/index.jsx (lines 17-40, 42-68, 85-91)		model/movie.js (lines 351-374)
	src/utis/functions.js (lines 213-259)		
	src/components/RootPage/index.jsx (lines 61-62, 110-112, 184, 186, 189, 251-255, 259-260, 270-278)		
	src/components/DeleteMoviePage/index.jsx (lines 32-36, 38-42)	<->	controller/app.js (lines 294-313)
	src/components/DeleteMoviesForm/index.jsx (lines 17-40, 42-71, 95-101)		model/movie.js (lines 375-408)
	src/utis/functions.js (lines 267-330)		
	src/components/RootPage/index.jsx (lines 63-64, 110-112, 184, 187, 189, 251-255, 259-260, 270-278)		

### 3 APPENDIX

A brief description of what the additional / advanced features that were added in as well (beyond the scope of the CA1/CA2 requirements)

#### 3.1 Advanced / Additional features

High level description of what other advanced / additional functionalities the system contains.

This section includes

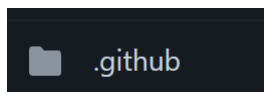
- i) a general description of the functionality in words.
- ii) screenshots demonstrating how the functionality works (i.e. what would be observed by the User)

As the team leader (Melvin), I wanted to explore beyond the syllabus and incorporated various advanced/additional features in the project.

The following below are the various advanced/additional features that I've also incorporated into the project:

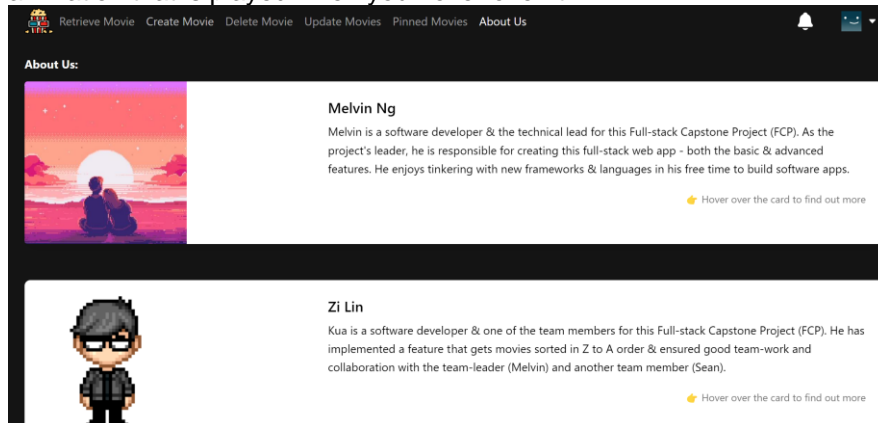
**a) Advanced / Additional code repository functionalities:**

- a. Wrote two yaml scripts to setup GitHub Actions that adds reviewers & assignees when pull requests are being raised. That way, when my team mates raise a PR, I will be automatically notified of their request. The 2 yaml scripts can be found in the .github folder.



**b) Advanced / Additional frontend functionalities:**

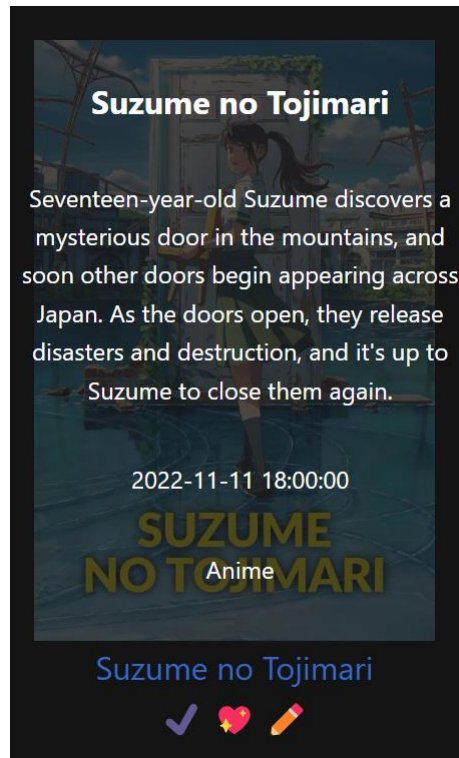
- a. I hosted our frontend React.js app on GitHub Pages - <https://sp-cet-capstone.github.io/capstone-assignment-class-1-team-1/website/index.html>
- b. I've created an 'About Us' page for people to see who were the contributors to our project. It contains 3 cards – one for each member, and each card has a custom CSS animation that is played when you hover over it.



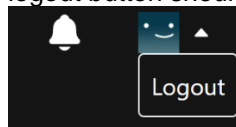
- c. Website/web app is mobile responsive utilizing media queries.
- d. Ability for the user or admin to search for movies by their name. This can be done by clicking the 'Search' icon in the Retrieve Movie homepage. Example:



- e. Ability for user/admin to pin and unpin movies by clicking on the 📌 emoji
- f. Ability for admin to edit movies directly by clicking on the ✎ emoji
- g. Hovering a movie image will play a custom CSS animation that showcases that particular movie's details:



- h. Ability to toggle movies based on their active/inactive status & ability to filter movies by genre in the 'Retrieve Movie' homepage
- i. Ability for user or admin to logout. To do this, hover over the smiley icon/picture & a logout button should appear:



**c) Advanced / Additional backend functionalities:**

- a. As a software developer, I understand the importance of writing good secure applications. I'm still learning with regards to enhancing the security of web applications (so I do look forward to your suggestions when grading our project 😊).
- b. Nonetheless, I wanted to incorporate security into our web applications. These was I have done using to enhance the security of our project's backend application:
- c. I hosted our backend Node.js/Express.js app on AWS, and added a SSL/TLS certificate to it to make it HTTPS using a variety of tools (e.g. .tech domain, AWS Route 53, AWS Certificate Manager, Application Load Balancer, EC2 Target Group settings, AWS Security Groups settings, Redirection of route requests).
- d. Created and added a custom protective middleware to protect against CSRF (Cross-site Request Forgery) attacks. The middleware is called 'verifyAgainstCSRFAttacks' and can be found in the `server/auth/verificationLib.js` file (lines 19-94).
- e. Our `/login` API mechanism involves validating a user's credentials (email and password), and if validated, the server will send and set a cookie on the client which contains the JWT token. To prevent the cookie from being stolen by hackers/attackers via various attack methods such as XSS or Man-In-The-Middle attacks, I have ensured that the cookies are sent to the client securely. An example of the relevant code snippets can be found in `server/controller/app.js` (lines 545-550) where I am sending the cookie (containing the JWT token) over `httpOnly: true` & `secure: true` (thus preventing XSS & Man-in-the-middle attacks). You can refer to the comments at line 547 & 548 of `app.js` & [this link](#) for more information as well.
- f. Implemented 2 additional API routes (i.e. `DELETE /movies/:movieID` & `DELETE /movies`) which were not part of the original BDD CA1/CA2 requirements.



## 3.2 Reference Links

For additional information, please kindly refer to the following links below:

- GitHub Repository Link : <https://github.com/SP-CET-Capstone/capstone-assignment-class-1-team-1>
- README.md (\*\*Important\*\* for understanding the web app & login credentials are also listed there): <https://github.com/SP-CET-Capstone/capstone-assignment-class-1-team-1/blob/main/README.md>
- Our group's Kanban Board: <https://github.com/orgs/SP-CET-Capstone/projects/11>
- Roles & Responsibilities of each team member (we worked together like a Scrum team): <https://github.com/orgs/SP-CET-Capstone/projects/11?pane=issue&itemId=19145533>

## 3.3 Credits

- This project was proudly done by:
  - o Ng Cheng Wai Melvin (P7411407) 🧐 - **Team Leader**
  - o Kua Zi Lin (P7461142) 🐱 - **Team Member**
  - o Ng Chye Yong (P7461085) 🤖 - **Team Member**

© 2023 IT8907-FCP Class 1 Team 1