



SAINT LOUIS UNIVERSITY
SCHOOL OF ACCOUNTANCY, MANAGEMENT, COMPUTING, AND
INFORMATION STUDIES



IT 312
Web Technologies
Final Activity

Instructor:
Kasima Rose Mendoza

Group Members:
Alingay, Jillah Marie
Chan, Duane Matthew
Efe, Justin Lenard
Madison, Jhoie Amber
Santiago, Denzel Khyle
Siapno, Renzo Romulo

DOCUMENTATION



I. Inventory of Web Resources

The web resources of the project consist of two folders: the “admin” folder and the “user” folder.

Admin module/folders

The “admin” files are incorporated with the scripting language PHP. The “admin” folder will be responsible for utilizing the administrator processes in the system.

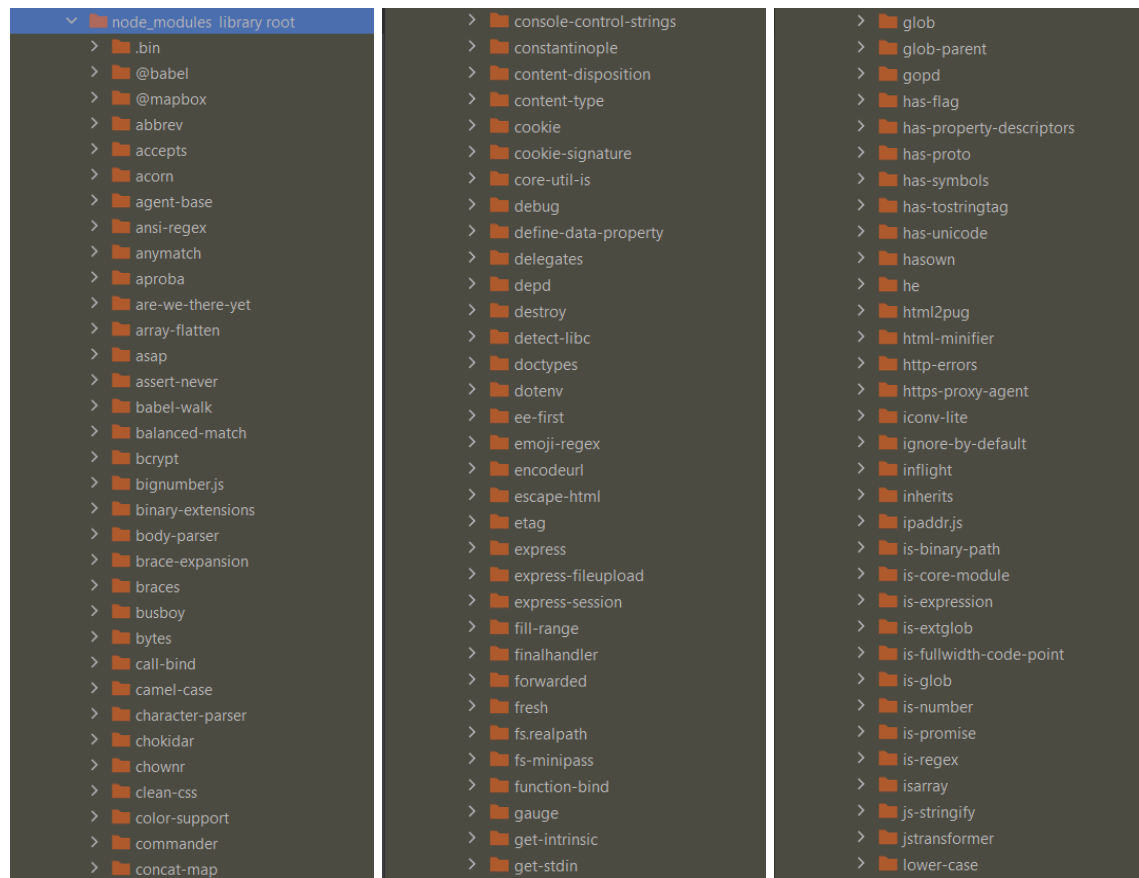
File/s	Description
add-user.php	This code will allow the admin to add another manager to the database. The admin can input the new user’s username and password.
dashboard.php	This webpage is the admin dashboard that will welcome the user. It provides a button to add a new manager, access Super Admin features, and display user data. It uses modals for adding managers and accessing Super Admin features.
database.php	This code establishes the connection to the database in MySQL.
delete-user.php	This code handles the deletion of a user from the database. It will display a success or error message and redirect the user to the "read-user.php" page.
demote-to-manager.php	This code allows demoting a user from the “admin” role to the “manager” role.
elevate-privilege.php	This code allows updating, deleting and demoting admin users to managers with additional validation for the default admin user “admin.”
Favicon.png	This is the favicon image of the website.

	
index.php	This file helps manage user sessions. If someone isn't logged in, it shows a login form. If they're logged in, it takes them to the dashboard.
login.php	This file will verify the entered credentials and either log in as an admin, show an error for non-admin users or show an error for non-existing users. If the entered credentials are an admin, it will redirect to the dashboard page.
login_form.html	This webpage includes text fields for the username and password wherein the functionalities of the login.php are utilized.
Logo.png 	This is the logo image of the website.
logout.php	This file logs out the user by removing their login details and will redirect them to the index.php page.
promote-to-admin.php	This code handles the process of promoting a user from a "manager" to an "admin" role. This is possible by updating the user's role in the database and then redirects the user to the dashboard page.
read-user.php	This webpage lets the user read all the users from the database displayed in a table; it will also allow users to filter, sort, and search for specific users. It also provides buttons to

	update and delete users with special handling for admin users.
update.php	<p>This file updates user information depending on the username and password. It will update the database accordingly based on the input and will be redirected to the dashboard page.</p> <p>Additionally, it handles specific actions like promoting a user to admin or demoting a user to manager.</p>
update-admin-password.php	This file handles the changing of password for the user with the user role and username of 'admin.'

node_modules

- folders and files that were implemented after nodejs were installed. And other installations such as multer, etc.





User module/folders

- The “user” files are incorporated with JavaScript and NodeJS. The “user” folder will be responsible for the live streaming on the viewer side and login/logout system for the content manager. The content manager will also be responsible for organizing the live streaming that will be seen on the viewer's side.
- The “user” folder consists of five (5) sub-folders—and five (5) files without a folder.

1.0 “model” folder

1.1 auth

File/s	Description
auth.js	This code functions as controlling the access

	<p>of the dashboard page and login page depending on the user's authentication status.</p> <p>'checkUnauthorized' ensures that a user is logged in before allowing access to certain routes, redirecting to the login page if not.</p> <p>On the other hand, 'checkAuthorized' ensures that a user is not already logged in before allowing access to certain routes, redirecting to the dashboard page if they are.</p>
--	--

1.2 encryption

File/s	Description
encrypt.js	This code functions as a middleware for encrypting a user's password before processing a request.

1.3 error

File/s	Description
error.js	This code will send a "404 Not Found" page when the user tries to access a page or resource that doesn't exist on the website.

1.4 log

File/s	Description
logger.js	This code makes a request to the website, and this code logs some details about that request.

1.5 schedule

File/s	Description
schedule.js	This code will be fetching a schedule and the video data in that schedule. The result will be the array of video data corresponding to the schedule.

1.6 session

File/s	Description
session.js	This code ensures that the user's session remains consistent with the stored status in the database. If an error is detected, it logs the user out and redirects them to the login page. This file serves to prevent unauthorized access or session tampering.

1.7 sql

File/s	Description
database.js	This code will handle the following: <ul style="list-style-type: none">- user authentication- interacting with the database to manage data- stores and retrieves information about media files- deals with scheduling
lola_flora.sql	This file is where the details of the database are stored.

“public” folders

- The public folders consist of the static websites of the user and content-manager side.

3.0 “res” folder

- This file consists of the images of each team member of Lola Flora.

File/s	Description
Denzel.png	This file is a photo of lola flora team member: DENZEL KHYLE SANTIAGO



Duane.png



This file is a photo of lola flora team member:

DUANE MATTHEW CHAN

Jhoie.png



This file is a photo of lola flora team member:

JHOIE AMBER MADISON

Jillah.png




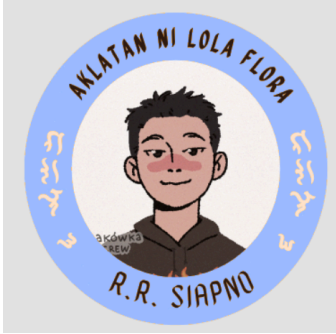
This file is a photo of lola flora team member:

JILLAH MARIE ALINGAY


justin.png


This is a photo of lola flora team member:

JUSTIN LENARD EFE

	
<p>renzo.png</p> 	<p>This is a photo of lola flora team member: RENZO ROMULO SIAPNO</p>

3.1 “public” folder

File/s	Description
<p>about.html</p>	<p>This is the webpage of the about us page. It will provide insights to what is the objective of the website’s content.</p>
<p>Favicon.png</p> 	<p>This is the favicon image of the website.</p>
<p>footer.html</p>	<p>This is the webpage's footer used for the admin side.</p>
<p>header.html</p>	<p>This is the webpage’s header used for the admin side.</p>
<p>home.html</p>	<p>This is the webpage of what the user/viewer</p>

	will see.
Logo.png 	This is the logo image of the website.
style.css	This is where all of the website designs are integrated.
viewer-page-script.js	This code will handle real-time updates through WebSockets. It will also handle playing the scheduled videos and will manage the different views and buttons for live streaming which are the screensharing and camera, and the past streams as well. Additionally, it keeps the date and time updated and logs information to the console.

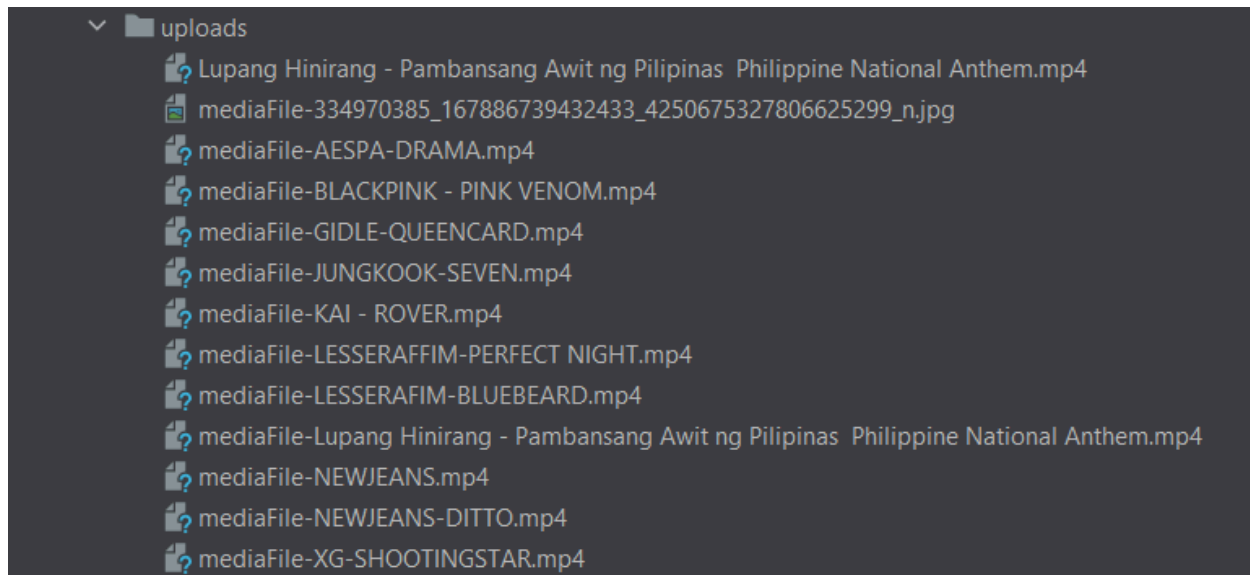
4.0 “routes” folder

File/s	Description
login.js	This file serves as a server-side route handling user login. It will validate user credentials, manage user sessions and will redirect the users to the dashboard web page upon successful login.
logout.js	This file will log out a user. If the user requests to log out, it will be updated in the database, will end their session, and will redirect them to the login page.
schedule.js	This code will set up routes for adding and retrieving schedule data on a web application. It checks the authorization for adding schedule and fetching the newly added schedule unto the schedule table.
upload.js	This code handles file uploads and retrieval of the uploads in the content manager. It will use ‘multer’ as a middleware to handle the file

	<p>uploads and will save the uploaded files in the folder 'uploads.'</p> <p>Additionally, it will store the information of these uploaded files in the database, this information will then be viewed on the webpage of the content manager.</p> <p>Error messages are also applied in this code and will be reflected in the console when errors are detected.</p>
video.js	<p>This code will serve as a server for streaming videos and schedules automatic video that will be switching at regular intervals.</p> <p>This code will provide routes for streaming the current video and retrieving specific videos based on their mediald.</p>

5.0 “uploads” folder

- Where the uploaded media files will be stored.



6.0 View folder

6.1 auth-login

File/s	Description
--------	-------------

login.pug	This webpage is the login form. The form will take the username and password. There is an integrated exception error if an error occurs during logging in.
login-failed.pug	This file will indicate the webpage if the user failed to log in. In short, this is the failed login message.
login-successful.pug	This is the login successful message.
logout.pug	This webpage is the logout form where the user will be asked if they are sure to logout their credentials.

6.2 content-manager

File/s	Description
dashboard.pug	This code will create a webpage in the content manager to handle information for the live stream which are the screensharing livestream, and pastreams.
upload.pug	This code creates a webpage in the content manager where the content managers will upload media files in the form. The page will also present a table for the uploaded media files.
schedule.pug	This pug file enables the content managers to add schedules. This will also create a table for the created schedules.

6.3 error

File/s	Description
404.pug	This is a web page designed for a "404 Not Found" error.

6.4 viewer

File/s	Description
viewer.page.pug	This creates a webpage that only contains the video frame for the viewers.

No folder under “view” folder files

File/s	Description
about.pug	This pug file shows the webpage of the about page.
footer.pug	This pug file shows the footer of the website.
header.pug	This pug file shows the header of the website.
index.pug	This file will show the webpage structure of the home website. It is integrated with the header and footer sections. The login button is shown (if no user is logged in), and the main content on which the viewer-page.pug is called upon. It also includes the button for Upload page and Schedule page.
video.pug	This code displays the screen sharing live stream, camera stream and pastreams for dashboard.pug and viewer-page.pug.

No folder under “user” folder

File/s	Description
.env	This file contains secret keys and tokens used for secure communication and session management in a web application.
package.json	This file shows the information on the Node.js project. The name, version, dependencies, and scripts are shown. This will manage project configuration and dependencies.
package-lock.json	This file keeps track of the versions of all the dependencies used in the project. It will ensure that every member that is working on the project installs the same version of the said dependencies, to avoid compatibility issues.
server.js	This code is responsible for setting up the following: <ul style="list-style-type: none">- a web server using Express- handling different routes- interacting with the database- providing user authentication and fetching media files

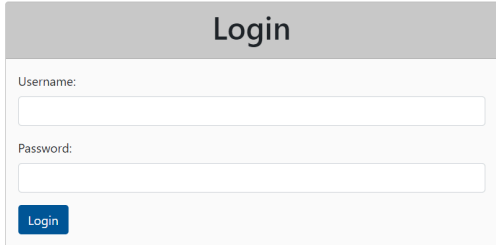
II. Relational Schema

USER(id, role, username, password, login)
AK username

MEDIAFILES(mediaId, userId, fileName, fileData, fileType, size, uploadDate)
AK filename
FK userId References USER Nulls Not Allowed
Delete Restrict, Update Restrict

SCHEDULE(scheduledId, videoId, day, start, end)
FK videoId References MEDIAFILES Nulls Not Allowed
Delete Restrict, Update Restrict
userId References USER Nulls Not Allowed
Delete Restrict, Update Restrict

III. List of essential features description of the UI

ADMIN SIDE	
Essential feature of the UI	Description
Admin Login 	The page wherein the user will log in. This allows the user to input the admin's username and password.
Admin Login failed message	The failed message will show when the user fails to input the right credentials for the admin.

Login

Username:

Password:

Login

Login failed! Invalid username or password.

Admin Dashboard

ESELYU

LIVE

[Home](#)
[About](#)
[Logout](#)

Welcome back ADMIN!

Add & New Manager

Update Settings

Filter By:

All Users

Admins

Managers

Sort By:

Sort Ascending

Sort Descending

Search Users

Search

List of Users

#	Role	Username	Last Name	First Name	Middle Name	Login	Update	Delete	Promote
1	admin	admin				1	Update	Delete	Promote
2	manager	duane	CHAN	DUANE MATTHEW	MATTHEW	0	Update	Delete	Promote
3	manager	Zorero	SAPHO	RENZO	ROMULO	0	Update	Delete	Promote
4	admin	jaya	ALINGAY	JILLAH	FRANCISCO	0	Update	Delete	Promote

The admin dashboard will show upon successful login.

Add User

Add A Manager

Last Name

Minimum of 1 character and a maximum of 50 characters.

First Name

Minimum of 1 character and a maximum of 50 characters.

Middle Name

Minimum of 1 character and a maximum of 50 characters.

Username

Minimum of 4 characters and a maximum of 8 characters.

Password

Minimum of 4 characters and a maximum of 6 characters.

Cancel

Add Manager

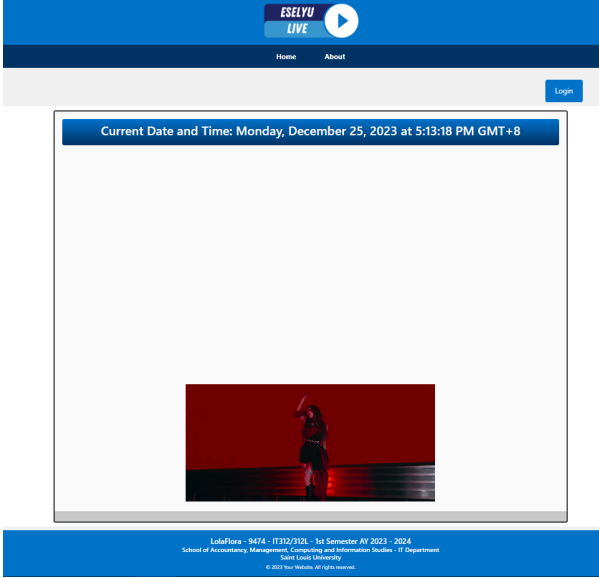
The add user functionality serves for adding a new manager. The user/admin will input the new user's information such as last name, first name, middle name, username and the password. After adding this will be directly reflected on the dashboard's list of users and in the database.

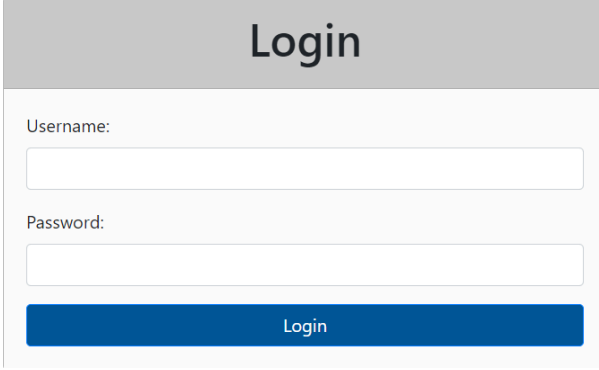
Elevate Privileges

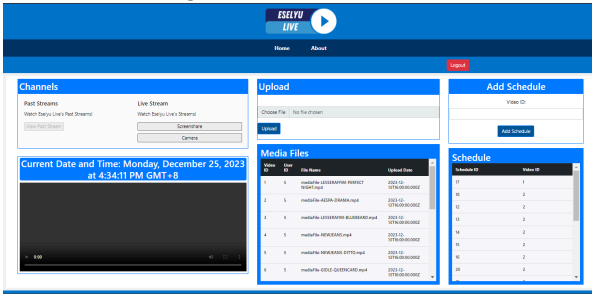

The elevate privileges functionality will serve as a separate modal for editing the

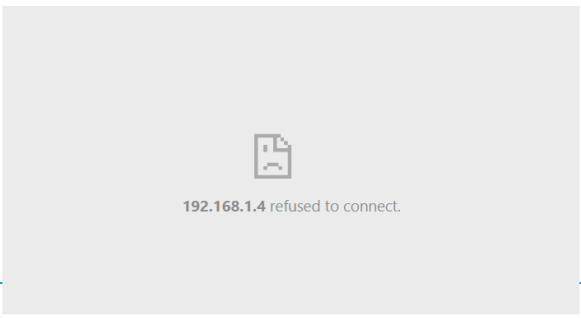
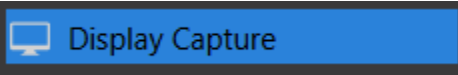
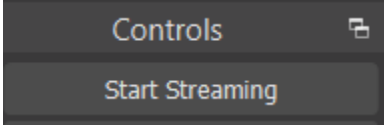

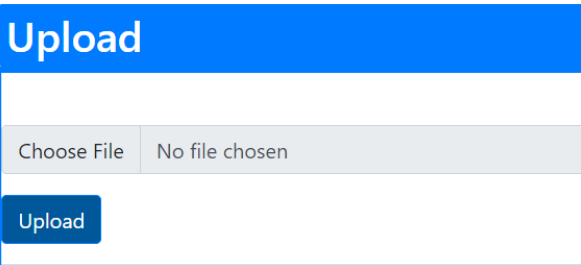
<div><div>Edit Admin Users</div><div><div>List of Admin Users</div><table><thead><tr><th>#</th><th>Role</th><th>Username</th><th>Update</th><th>Delete</th><th>Demote</th></tr></thead><tbody><tr><td>1</td><td>admin</td><td>admin</td><td><button>Update</button></td><td><button>Delete</button></td><td><button>Demote to Manager</button></td></tr><tr><td>2</td><td>admin</td><td>duane</td><td><button>Update</button></td><td><button>Delete</button></td><td><button>Demote to Manager</button></td></tr></tbody></table><div>End of Table</div><div><button>Close</button></div></div></div>	#	Role	Username	Update	Delete	Demote	1	admin	admin	<button>Update</button>	<button>Delete</button>	<button>Demote to Manager</button>	2	admin	duane	<button>Update</button>	<button>Delete</button>	<button>Demote to Manager</button>	<p>credentials of users with admin roles. Here the user has the freedom to update the admin’s password, delete an admin and demote them to manager.</p>
#	Role	Username	Update	Delete	Demote														
1	admin	admin	<button>Update</button>	<button>Delete</button>	<button>Demote to Manager</button>														
2	admin	duane	<button>Update</button>	<button>Delete</button>	<button>Demote to Manager</button>														
<div><div>Filter</div><div><div>Filter By:</div><div><button>All Users</button><button>Admins</button><button>Managers</button></div></div></div>	<p>The filter function will allow the user to filter the list of users into ‘all users’, ‘admins’ and ‘managers’.</p>																		
<div><div>Sort By</div><div><div>Sort By:</div><div><button>Sort Ascending</button><button>Sort Descending</button></div></div></div>	<p>The sort function will allow the user to sort the list of users in an ascending and descending manner.</p>																		
<div><div>Search Users</div><div><div>Search Users</div><div><button>Search</button></div></div></div>	<p>The search function will only be allowed to search the usernames of the list of users.</p>																		
<div><div>Update User</div></div>	<p>The update user function will allow the user/admin to update the user’s last name, first name, middle name, username and password.</p>																		

<div data-bbox="207 210 803 945"> <div>Update User</div> <div>×</div> <div>Last Name</div> <div>SIAPNO</div> <div>Minimum of 1 character and a maximum of 50 characters.</div> <div>First Name</div> <div>RENZO</div> <div>Minimum of 1 character and a maximum of 50 characters.</div> <div>Middle Name</div> <div>ROMULO</div> <div>Minimum of 1 character and a maximum of 50 characters. Please put 'N/A' if not applicable.</div> <div>Username</div> <div>Zoren</div> <div>Minimum of 4 characters and a maximum of 8 characters.</div> <div>Password</div> <div></div> <div>Minimum of 4 characters and a maximum of 6 characters.</div> <div>Cancel</div> <div>Save changes</div> </div>	
<div data-bbox="207 1035 803 1302"> <div>Delete User</div> <div>×</div> <div>Delete User</div> <div>Are you sure you want to delete this user?</div> <div>Cancel</div> <div>Delete</div> </div>	<p>The delete user function will allow the user/admin to delete a user. The modal will then appear when the delete user button has been clicked to ask the user if they are sure to delete this user.</p>
<div data-bbox="207 1350 803 1480"> <div>Promote to Admin</div> <div>Update</div> <div>Delete</div> <div>Promote to Admin</div> </div>	<p>The admin can simply click a manager's row to promote them to admin.</p>

USER/VIEWER SIDE	
Essential feature of the UI	Description
<p>Viewer Dashboard</p> 	<p>The viewer dashboard will show the user what is now being shown by the content manager. In the screenshot, the past streams are playing.</p>

CONTENT MANAGER	
Essential feature of the UI	Description
<p>Content Manager Login</p> 	<p>The webpage wherein the content manager will log in with their credentials.</p>
<p>Content Manager Logout</p>	<p>The webpage where the manager will be redirected to when they request to be logged out. The system will then ask the manager if they are sure to logout.</p>

<div data-bbox="207 205 795 468"> <h2>Logout</h2> <p>Are you sure you want to logout, Amber?</p> <div> Logout Cancel </div> </div>	
<div data-bbox="207 499 795 877"> <h3>Content Manager Dashboard</h3>  </div>	<p>The content manager dashboard displays the commands of the manager which are: past streams, camera, and screenshare buttons. It will also show the functionalities which are the upload media files and adding of schedules. The uploaded media files will also be shown in the table 'Media Files', the table will be updated without refreshing the button, this is because AJAX is applied. It is the same for the schedule table which we would see the queueing of videos in that table.</p>
<div data-bbox="207 898 795 1192"> <h3>Content Manager Commands</h3> <h4>Channels</h4> <div> <div> <h5>Past Streams</h5> <p>Watch Eselyu Live's Past Streams!</p> View Past Stream </div> <div> <h5>Live Stream</h5> <p>Watch Eselyu Live's Streams!</p> <div> Screenshare Camera </div> </div> </div> </div>	<p>This shows the command buttons of the content manager that vary into three categories: past streams, screen share, and camera.</p>
<div data-bbox="207 1213 795 1717"> <h3>Past Stream Button Functionality</h3> <div> <p>Current Date and Time: Monday, December 25, 2023 at 5:23:50 PM GMT+8</p>  </div> </div>	<p>The past stream functionality will be shown on the video frame and the videos that will be shown and fetched here are from the submitted video IDs of the content manager. Each of those video IDs that have been submitted will automatically be in queue to be shown in that video frame.</p>
<div data-bbox="207 1738 795 1879"> <h3>Screenshare Button Functionality</h3> </div>	<p>The screen sharing will be shown here by using an API of owncast. The content manager needs to set up this API with the Ubuntu server and the screen-sharing</p>

<div data-bbox="215 216 792 338"> <p>Current Date and Time: Monday, December 25, 2023 at 5:24:30 PM GMT+8</p> </div> <div data-bbox="215 338 792 653">  </div>	<p>functionality will be triggered by using the OBS screen-sharing functionality. Using 'Display Capture',</p> <div data-bbox="833 310 1287 384">  </div> <p>and the 'Start Streaming' button.</p> <div data-bbox="828 430 1209 554">  </div>
<p>Camera Button Functionality</p> <div data-bbox="215 783 792 905"> <p>Current Date and Time: Monday, December 25, 2023 at 5:25:11 PM GMT+8</p> </div> <div data-bbox="215 905 792 1251">  </div>	<p>The camera button functionality will access the camera of the content manager and this will be shown on the video frame.</p>
<p>Upload button</p> <div data-bbox="215 1325 792 1587">  </div>	<p>The content manager will be able to upload videos that will be stored in the database.</p>
<p>Upload Success Message</p>	<p>The success message will show when the admin successfully uploads a media file and stores it in the database.</p>

Upload

Choose File No file chosen

Upload

File uploaded and stored in the database successfully!

Media Files Table

Media Files

Video ID	User ID	File Name
1	5	mediaFile-LESSERAFIM-PERFECT NIGHT.mp4
2	5	mediaFile-AESPA-DRAMA.mp4
3	5	mediaFile-LESSERAFIM-BLUEBEARD.mp4
4	5	mediaFile-NEWJEANS.mp4
5	5	mediaFile-NEWJEANS-DITTO.mp4
6	5	mediaFile-GIDLE-QUEENCARD.mp4

The media files table will allow the manager to see the list of the videos stored in the database. Once the manager has also uploaded a video, it will be redirected to the media files table.

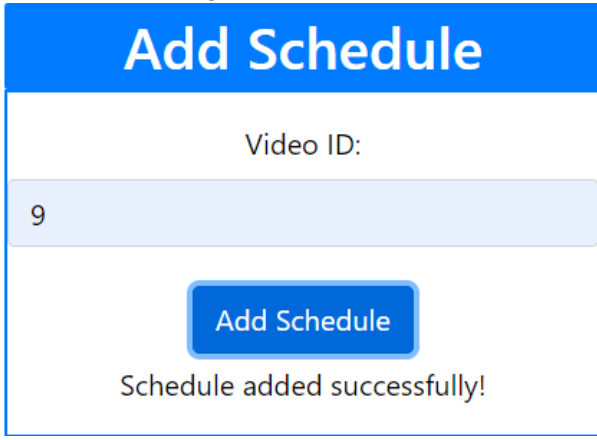
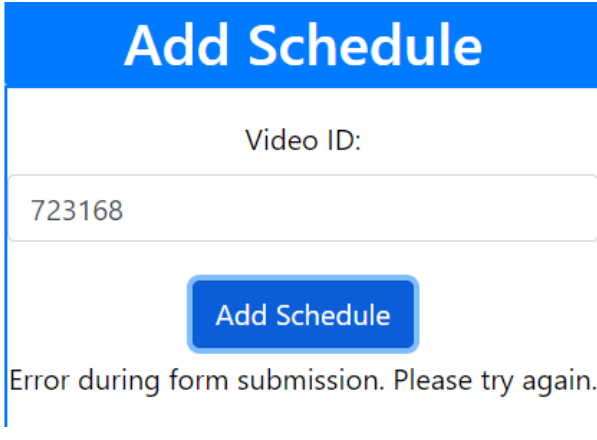
Schedule Form

Add Schedule

Video ID:

Add Schedule

The content manager will be able to create a new Schedule by inputting the Video ID of the uploaded media files.

<p>Success Message</p> 	<p>The success message will show when the admin is able to put valid inputs in adding a schedule.</p>
<p>Error Message</p> 	<p>The failed message will show when the admin fails to put valid inputs in adding a schedule.</p>
<p>Schedule Table</p>	<p>Once the manager has created a schedule it will be redirected to the schedule table.</p>

Schedule

Schedule ID	Video ID
20	1
19	2
18	3
17	4
14	5
12	6
21	6
13	7
16	8