



Open AmDram Portal- A Lightweight Website & Web Portal for Small Theatre Groups

Matthew Larby

C1933765

CM3203 - One Semester Individual Project

Credits: 40

School of Computer Science
Cardiff University, 2022

Author: Matthew Larby
Supervisor: Dr Daniela Tsaneva

Abstract

A small Amateur Dramatics group reached out needing a new website including image galleries that fetched from their existing photo storage service, a full archive of the details surrounding their previous productions, and a secure portal with which to share resources and announcements regarding upcoming shows. This project also works on the challenge of keeping webpages loading quickly while still delivering hundreds of photos in a single gallery. This paper covers the initial research into existing solutions followed by the development of a new website and portal, along with the challenges faced and the approaches used when implementing all the requested features.

Acknowledgements

Thank you to Dr Daniela Tsaneva for all of your encouragement, guidance, and support during this project.

Many thanks to the Silchester Players theatre group for being the client of this project.

I would also like to thank Gregor Webster and my parents, Mandy and Mark Larby, for their feedback on the report writing.

Table of Contents

Abstract	2
Acknowledgements	2
Table of Contents	2
Table of Figures	4
1. Introduction	5
1.1 Project Aim	5
1.2 Project Needs	5
1.3 Intended Users	6
1.3.1 Public Site Visitor	6
1.3.2 Theatre Group Member	6
1.3.3 Show Director/Producer	6
1.3.4 Blog Writer	6
1.3.5 Webmaster/Admin	7
1.4 Project Approach	7
1.5 Assumptions	7
1.6 Outcome Summary	8
1.6.1 Project Architecture	8
1.6.2 Feature List	8
2. Background	9
2.1 The Wider Context of the Project	9

2.2 Project Constraints	9
2.3 Technical Capabilities	10
2.4 Similar Existing Solutions.	11
2.4.1 WordPress	11
2.4.2 Blackboard Learn	11
2.4.3 IMDb - The Internet Movie Database	12
3. Specification & Design	13
3.1 User Personas	13
3.1.1 Public Site Visitor	13
3.1.2 Theatre Group Member	14
3.1.3 Show Director/Producer	15
3.1.4 Blog Writer	16
3.1.5 Webmaster/Admin	17
3.2 Use Cases	17
3.3 Technology Choice	23
3.3.1 Backend	23
3.3.2 Database	24
3.4 Site Accessibility	25
3.4.1 Text Contrast	25
3.4.2 Font Size	25
3.4.3 Site Theme (Positive vs Negative Contrast)	25
3.4.4 Device Compatibility	25
4. Implementation	26
4.1 Web Portal Authentication	26
4.2 Database	27
4.3 Google Photos API	29
4.3.1 API Authentication with OAuth	29
4.3.2 BaseURL Expiry Documentation	29
4.3.3 Re-Authenticating without an OAuth Pop-up	30
4.3.4 The Final Implementation	31
4.4 Fast Photo Gallery	33
5. Results and Evaluation	35
5.1 Results	35
5.1.1 The Prototype	35
5.1.2 Measurable Results	35
5.1.3 Testing	37
5.1.4 Project Accomplishments	46

5.2 Evaluation	46
6. Future Work	47
6.1 Consolidation of Similar Database Tables	47
6.2 Full Screen Photo Gallery	47
6.3 General Styling of the Web Portal.....	47
6.4 User Account Registration and Management	47
6.5 Any New Requested Features	48
7. Conclusions.....	48
8. Reflection on Learning	48
8.1 Learning OAuth.....	48
8.2 Working with a Real-World Client	48
8.3 Timescales	49
Glossary	50
Table of Abbreviations.....	50
Appendix	50
References.....	51

Table of Figures

Figure 1. System Architecture Diagram.....	8
Figure 2: Facebook Post Engagement Jan-Apr 2022	10
Figure 3: Google Business Profile - Website Clicks Jan-Apr 2022	11
Figure 4: Website Traffic by Device Type (Mobile, Desktop) 2009-2022 [11].....	26
Figure 5. The route that handles the 2FA code verification.	27
Figure 6: Database Relationship Diagram.....	27
Figure 7. Cast Member Name Change in IMDb	28
Figure 8. Cast Member Name Change in this project.....	29
Figure 9. The route redirecting the user to the OAuth login screen.....	30
Figure 10. Authorisation code exchange.	31
Figure 11. Image Collection	32
Figure 12. Logic for efficiently adding photos to the database.	33
Figure 13: Visual representation of the gallery.	34
Figure 14. The JavaScript gallery element.	35
Figure 15. Sites Speed Comparison (Generated with WebPageTest.org)	36
Figure 16. Still from Load Speed Test (in seconds)	37
Figure 17. Test Case 1 Step 3 Screenshot	38
Figure 18. Test Case 2 Step 1 Screenshot	39
Figure 19. Test Case 4 Step 3 Screenshot	41
Figure 20 Test Case 5 Step 1 Screenshot	42
Figure 21 Test Case 6 Step 3 Screenshot	43
Figure 22. Test Case 7 Step 3 Screenshot	44
Figure 23. Test Case 8 Step 1 Screenshot	45

1. Introduction

1.1 Project Aim

The aim of the project is to develop a custom website for a small amateur theatre group, along with a web portal to help them share the resources and information needed when putting on a show.

The website will have a few custom features that are not found in off the shelf CMS alternatives, including a section for a Past Shows Archive, automatically updating announcements on the “Home” page and on the “Auditions” page, and photo galleries that fetch the images from the group’s Google Photos account.

1.2 Project Needs

The ‘Past Shows’ archive will function similarly to sites like IMDb with the ability to see shows that a particular individual has been involved in along with the details of any past show, including when it was put on, the genre and the author alongside the full cast and crew lists taken from the programmes.

The photo galleries will also be included within the ‘Past Shows’ archive. The main function of this will be collating the images for each show from Google Photos and storing it in a way that the front-end gallery element can use to display them. The actual gallery element will display the photos in an easily accessible way whilst not taking too long to load in.

The “Home” and “Auditions” pages will need to be capable of automatically updating their content when a new public update is posted regarding an upcoming show. The “Auditions” page will only ever show the most recent post that is specifically tagged as being about an auditions event, whilst the “Home” page will always show the most recent public announcement (including auditions) about the next show to be performed.

The web portal side of the project handles anything that the theatre group will need to keep private and accessible to only their members including show updates, file sharing, updating the details of a show, adding photos to the gallery, and updating the group’s blog.

The show updates and file sharing features will have a page for each show where any and all updates will be shown in a feed of posts alongside a section with all the shared files. These pages will only be updateable by the director and producers for a particular show or by a site admin role.

The blog posting feature of the site will also be a restricted feature, accessible to only those who have an "author" or "admin" roles.

1.3 Intended Users

The project will have five main user roles.

1.3.1 Public Site Visitor

This user class encompasses any member of the public who is visiting the website looking for updates on upcoming shows, reading the blog, or looking through the show archives. This user class will not need to be authenticated with the site.

Available Features:

- Browse the site for updates.
- Read the blog.
- Look through the 'Past Shows' archive.

1.3.2 Theatre Group Member

This user class includes anyone involved in the theatre group. This is the first user class to be required to authenticate to access certain features.

Available Features:

- Everything from previous classes.
- See Private updates from within the web portal.
- Access files in the shared area for each show.
- Update their user profile found in the 'Past Shows' archive.

1.3.3 Show Director/Producer

This user class includes the members involved in the running of a specific show and allows them to share resources via file upload and to publish announcements to the web portal and the public website. This permission level should ensure that a director or producer for one show does not gain the permission to make changes to another show that they are not a director or producer of.

Available Features:

- Everything from previous classes.
- Upload documents to the shared area (e.g., scripts, music files).
- Post private announcements that can only be seen by logged in members.
- Post public announcements about their upcoming show.

1.3.4 Blog Writer

This user class is used to grant access to the blog writing side using a specific role in the web portal.

Available Features:

- Everything from the Theatre Group Member class.
- Access to the blog writing dashboard and editing and publishing tools.

1.3.5 Webmaster/Admin

This user class is assigned to those responsible for the website's maintenance and includes access to certain functions involved in deploying, updating, and maintaining the website.

Available Features:

- Has access to all previous features.
- Update site settings including title, about us page, and social media links.
- Generate sign-up links to prevent unauthorised sign-ups.

1.4 Project Approach

The approach used for developing this project is described as "Incremental". [1]

This approach allows for focusing on one feature at a time until all the features have been built. By working on the features in order of priority, it allows the project to be as close to the projects needs as at the end of the limited project timescale with only minor features or niceties missing.

1.5 Assumptions

A few assumptions must be made about the scope of the project in order to focus on development.

One of these assumptions made for this project is the technical ability of the theatre group's members. The project assumes that they will be able to use login forms and intuit how to navigate the web portal. Some level of competence must be assumed in order to focus on the functionality instead of having to build instructional elements into the web portal front-end.

Another assumption that has been made is that the members of the theatre group who create announcements and blog posts will be able to use Markdown for formatting any text content they create. The project will utilise some form of graphical element for entering the Markdown-styled content. This will have buttons for managing formatting in a similar style to that of a traditional word processor which will reduce the learning curve by using the members transferable knowledge. Any Markdown entry forms will be alongside a help button with either a pop-up or link to an external help page to further reduce the learning curve.

1.6 Outcome Summary

1.6.1 Project Architecture

The project will consist of a front end and back end which will be deployed on a Linux-based server along with a database for storage.

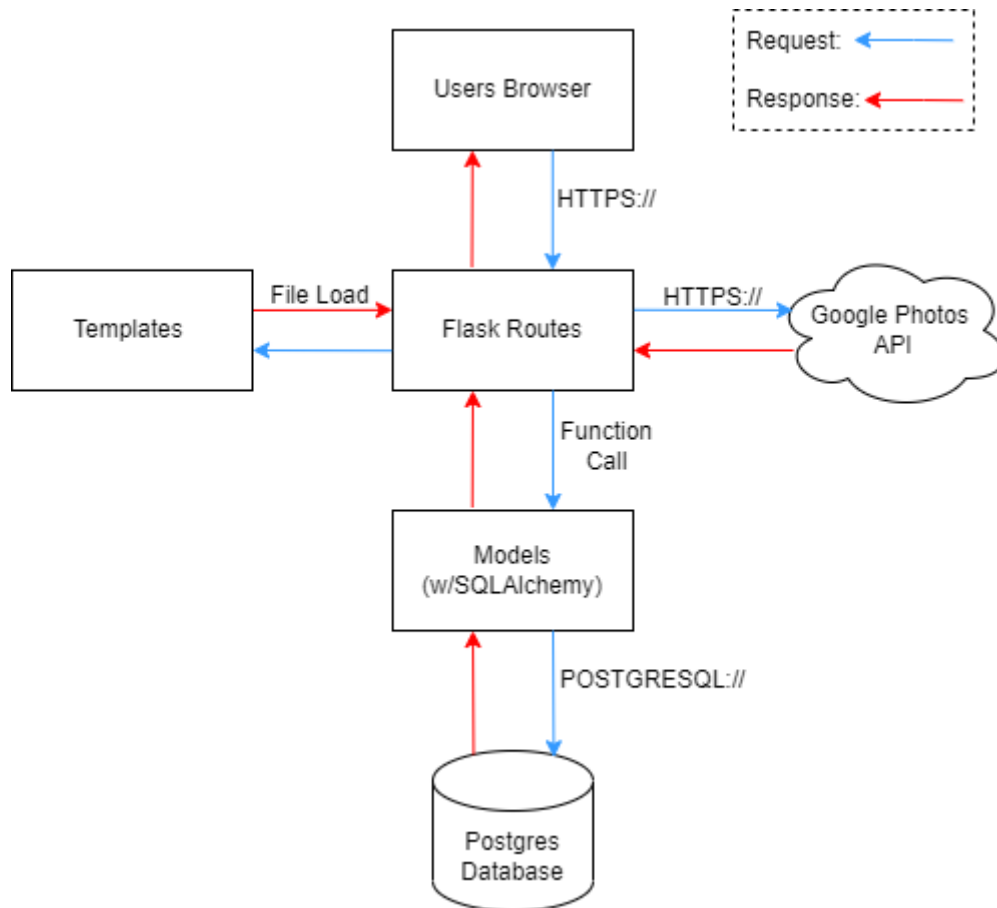


Figure 1. System Architecture Diagram.

(Note: Data requests are represented by the blue arrows and data responses by the red arrows)

The website will have a usable front end with clearly defined buttons for navigation and separate layouts for both desktop and mobile browsers. The look of the site will address any accessibility issues in the User Interface and ensure that it is usable by as many people as possible. The web portal should be clearly laid out in a manner that is intuitive and provide all the features that are needed by the client.

1.6.2 Feature List

- Public and Private show updates / announcements
- Blog with editor and Word document upload
- Archive of all past shows with photos loaded from Google Photos
- Web Portal for access control
- Shared files with restricted upload in an area for each show

- Member profile for each user that can be updated by them in the web portal
- Admin-only site settings including title, about us page and social media links
- Invitation only sign-up system to prevent unauthorised access

2. Background

2.1 The Wider Context of the Project

The client is a local Amateur Dramatics group who find their current website, based on WordPress, to be lacking features and tedious to update as all changes have to be made by the group's website admin.

The first issue to address is the tediousness of updating the public facing pages of the site. This includes notices about auditions and upcoming shows, and events that the public are invited to. As previously said, the changes are currently managed by the site admin but, in order to reduce their workload, the ability to update those parts of the site should be available to the director and producer of the show currently in production.

The theatre group also have trouble sharing files such as scripts amongst themselves as frequent updates to the script mean that it is somewhat impractical to keep requesting the website be updated by the admin. The current solution of emailing updated copies to the cast and crew results in many differing versions of script which often makes it difficult to find the most up-to-date copy. Sharing these files via the website, particularly in a password-protected section, is a better solution to this particular problem and that will be implemented in the project.

One of the issues the client has with their current site is the discontinuation of Picasa (a photo storage service which was bought by Google in 2004 and kept running until 2016 [2]) which broke the embedded photo galleries showcasing stills from the group's previous shows. When Picasa was shutdown, all the photos were safely transferred to albums stored on Google Photos, but the gallery feature was never fixed, as such, the client's site stopped displaying the majority of its photos. In the project, this will be rectified with a way to integrate the Google Photos API into a simple gallery element on each page.

As well as the issues with photos, the 'Past Shows' pages also lack a lot of information about each show, often times having no page in place at all. The pages that do exist are all static pages on the WordPress CMS which makes maintaining the content on each page an arduous process. The project will consolidate all the information about each show, including records of cast and crew, into a database for dynamically generating pages from a template which reduces the effort required to maintain and helps with integrating the photos back onto the site.

2.2 Project Constraints

One constraint is how the photos are stored. Currently, the group stores all the photos for each show on Google Photos with an established workflow for uploading new photos. This means that the project will have to integrate the

Google Photos API in the new implementation of a gallery while still being accessible to the public.

Another constraint is the accessibility of the site. The wide demographic of the theatre groups patrons requires considerations be made in ensuring sure the site is usable on all modern devices (i.e. computers, phones and tablets) and that content is suitably clear for everyone to read regarding contrast levels and font size.

2.3 Technical Capabilities

For deployment, the client needs the site to be capable of keeping up with the same level of traffic as their current WordPress based site.

Unfortunately, the client's current website provider does not have any analytics available for getting the traffic rates. However, a baseline from their Google Business Profile and Facebook page redirects can be used to make a rough estimate for what traffic the project must be capable of handling.

Looking at the statistics from the 1st of January 2022 to the 30th of April 2022, Facebook posts and Google Business Profile have 502 clicks (not including search results). This gives a baseline of about four clicks per day, but this is just average traffic from only two sources. In the lead up to a show, site traffic is expected to surge with the best performing shows run selling over six hundred tickets all of which is directed through the website.



Figure 2: Facebook Post Engagement Jan-Apr 2022

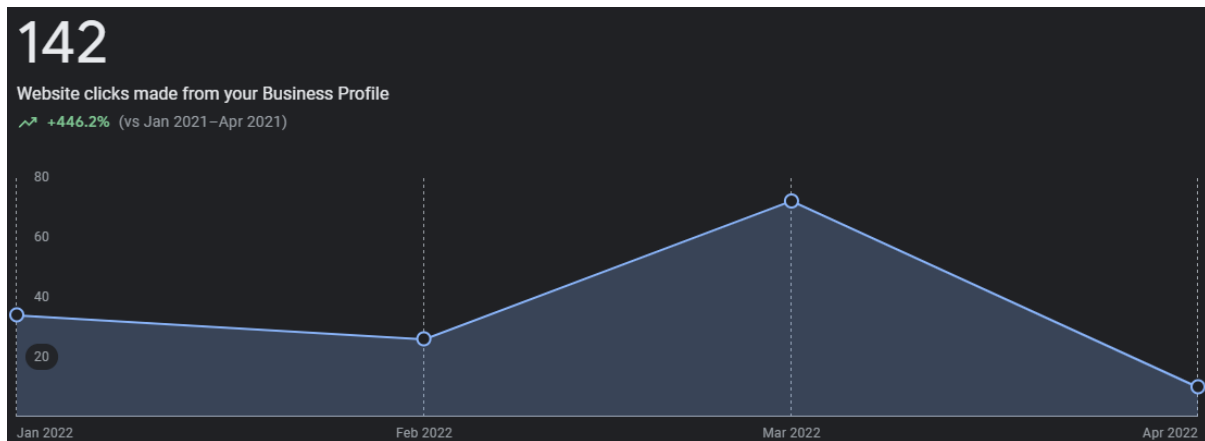


Figure 3: Google Business Profile - Website Clicks Jan-Apr 2022

With the new photos feature, the actual webserver is likely to experience at least two hundred requests per second during peak periods which needs to be considered when deciding on technologies to use. Most webserver, frameworks and reverse-proxies are built to handle at least these loads, especially within a production meaning, this is not a significantly limiting factor. Other than making a particularly poor decision, the choice can be based upon other factors like prior experience and ease of use.

The site also needs to load quickly. The current website has a noticeable delay when loading of a page which detracts from the overall usability. This requirement is a limiting factor which the development tools and technology choices need consider in order to result in as short a delay as possible.

2.4 Similar Existing Solutions.

2.4.1 WordPress

The client's current website is running on a WordPress backend which is great for handling mostly static content with the functionality built-in for a blog, however, having over 130 similar static pages is a very inefficient way to display the data from the past shows when this is an ideal case for storing in a database with a single template.

An unmodified WordPress installation also has no support for a restricted access area nor a file sharing feature without storing the files as static media element.

Also, modifying a WordPress page requires knowledge of HTML and CSS which limits the number of people with the capability to make changes. There is also the issue of regulating access if everyone in the group has access to the WordPress edit feature when only the director and producer of the upcoming/current shows need to be able to post announcements about it.

2.4.2 Blackboard Learn

Another solution is Blackboard Learn, a "web-based virtual learning environment and learning management system" [3]. This software is used by many universities for delivering learning materials with all content separated by module as well as providing announcements (e.g., lecture dates and times).

The way content is shared is similar to the file sharing feature that will be included in this project; however, their implementation is intended for handling a much larger amount of content. Since it is targeted at students, this solution has a higher margin for interface complexity whilst still maintaining usability. Theatre groups have members from a variety of backgrounds with many levels of technical ability so the interface for finding and downloading a file must be kept simple.

The way content is organised by modules in Blackboard Learn is very similar to how this project will separate resources by show. Their implementation is largely what inspired that organisational aspect of this project.

Whilst similar to this project, Blackboard Learn focuses on catering to large universities meaning there are many features that the client has no use for and its price ("£9,500 USD per year" [4]) puts it well out of reach to the client. Blackboard Learn also lacks much of the unrestricted site functionality that the client needs for the public facing side of their website meaning they would need a second product.

2.4.3 IMDb - The Internet Movie Database

IMDb stores data from "over 8 million movies, TV series, and video game titles" and connects it to the roles that "over 11 million cast and crew" [5] had in each including photos for the shows, descriptions, and more. One of the features the client wants is a complete overhaul of their old 'Past Shows' archive page which utilised static pages for each show. IMDb is a great comparison just on a different scale and dataset.


IMDb has been a great source of inspiration for this project as it provides an example for the user interface and how browsing shows, cast, and crew could be handled in this project. It also gives a good list of attributes that should be stored about any individual show, particularly with which to sort, group or filter the shows by.

Unfortunately, IMDb is a closed source, proprietary site which means it cannot even be deployed on a small scale for the 'Past Shows' archive. Even if it were available to deploy, it would lack the restricted web-portal aspect of this project which is one of the major features requested by the client.


3. Specification & Design

3.1 User Personas


3.1.1 Public Site Visitor

Persona	Content
Status Designation	Primary
Photograph	
Name:	Louise Penny
Age:	46
Occupation:	Librarian
Quote:	"I enjoy watching the performances from my local theatre group"
Narrative:	Louise is a forty-six-year-old librarian. She enjoys going to watch shows by the local theatre group and keeping up with the group's activities between shows. She also has a child who took part in some of the group's productions and likes to reminisce on those past shows by looking at photos.
Goals:	<ul style="list-style-type: none"> • Browse the site for updates. • Read the blog. • Look at photos in the 'Past Shows' archive.


3.1.2 Theatre Group Member

Persona	Content
Status Designation	Primary
Photograph	
Name:	Faith Montgomery
Age:	23
Occupation:	Student
Quote:	"I enjoy performing in shows produced by my theatre group"
Narrative:	Faith is a twenty-three-year-old student. She is a member of the theatre group and often performs in their shows, particularly pantomimes. As part of performing in shows, she needs to always have the latest copy of the script but struggles to find the right email with the correct version.
Goals:	<ul style="list-style-type: none"> • See Private type updates from within the web portal. • Access files in the shared area for each show.


3.1.3 Show Director/Producer

Persona	Content
Status Designation	Primary
Photograph	
Name:	Sam Oliver
Age:	62
Occupation:	Retired
Quote:	"I enjoy directing and producing shows for my theatre group"
Narrative:	Sam is a sixty-two-year-old retiree. He is a member of the theatre group and sometimes takes the role of director or producer in their shows. As part of producing or directing shows, he needs to keep the cast and crew up to date with scripts and rehearsal schedules.
Goals:	<ul style="list-style-type: none"> • Post updates of any type regarding their shows. • Upload files to the shared area for each show.

3.1.4 Blog Writer

Status Designation	Primary
Photograph	
Name:	Stephen Roberts
Age:	38
Occupation:	IT Support Technician
Quote:	"I like to write blog posts about what I do in my life"
Narrative:	Stephen is a thirty-eight-year-old IT technician. He is a member of the theatre group and likes to write blog posts about everything, including the theatre group. As part of the theatre group, he would like to publish these to their website.
Goals:	<ul style="list-style-type: none"> • Access to the blog writing dashboard and editing and publishing tools.

3.1.5 Webmaster/Admin

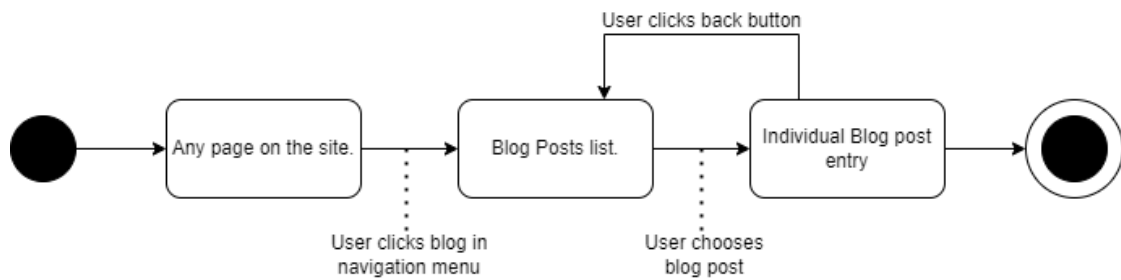
Status Designation	Primary
Photograph	
Name:	Roy Sheppard
Age:	23
Occupation:	Freelance Web Developer
Quote:	"Have you tried turning it off and on again?"
Narrative:	Roy is a twenty-three-year-old web developer. In the theatre group, he runs the lights for shows and manages their current website. He would like to spend less time keeping the website up to date with the latest information.
Goals:	<ul style="list-style-type: none"> • Update site settings including title, about us page and social media links. • Generate sign-up links for each member of the theatre group to prevent unauthorised sign-ups.

3.2 Use Cases

3.2.1a Use Case 1

Use Case No.1	Use Case Name: Read Blog	Rating: Must
Purpose: Read the blog.		
Main actor: Public Site Visitor		Secondary Actor: N/A
Pre-conditions: User is viewing the website on any page.		
Trigger: The user clicks the Blog link in the navigation bar.		
Description: <ol style="list-style-type: none"> 1. The user is shown a list of all blogposts that are stored in the database. 2. The user can click on a blog post and read it. 		
Extension: <ol style="list-style-type: none"> 1. If the user wishes to read another post, they can use the back button integrated into the blog post display. 		
Post-Conditions: User clicks Home to return to the site's homepage.		
Author: Matthew Larby		Date: 24.02.2022

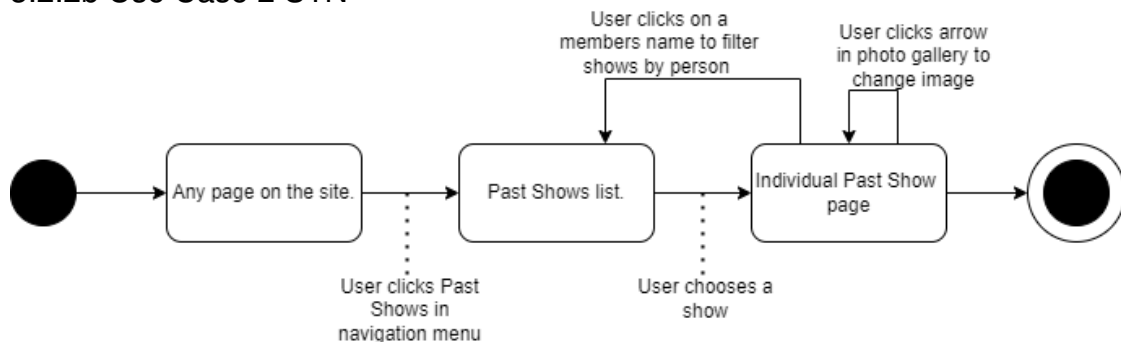
3.2.1b Use Case 1 STN



3.2.2a Use Case 2

Use Case No.2	Use Case Name: Past Shows Archive	Rating: Must
Purpose: Browse the Past Shows Archive		
Main actor: Public Site Visitor		Secondary Actor: N/A
Pre-conditions: User is viewing the website on any page.		
Trigger: The user clicks the Past Shows link in the navigation bar.		
Description: <ol style="list-style-type: none"> 1. The user is shown a list of all the Past Shows that are stored in the database. 2. The user can click on a show to see more details. 3. The user can click through the gallery to see every photo. 		
Extension: <ol style="list-style-type: none"> 1. The user can click on a member's name from the Cast or Crew lists to see a list of shows that person has been involved in. 2. If video content is available for a show, the user can play the video. 		
Post-Conditions: User clicks Home to return to the site's homepage.		
Author: Matthew Larby		Date: 24.02.2022

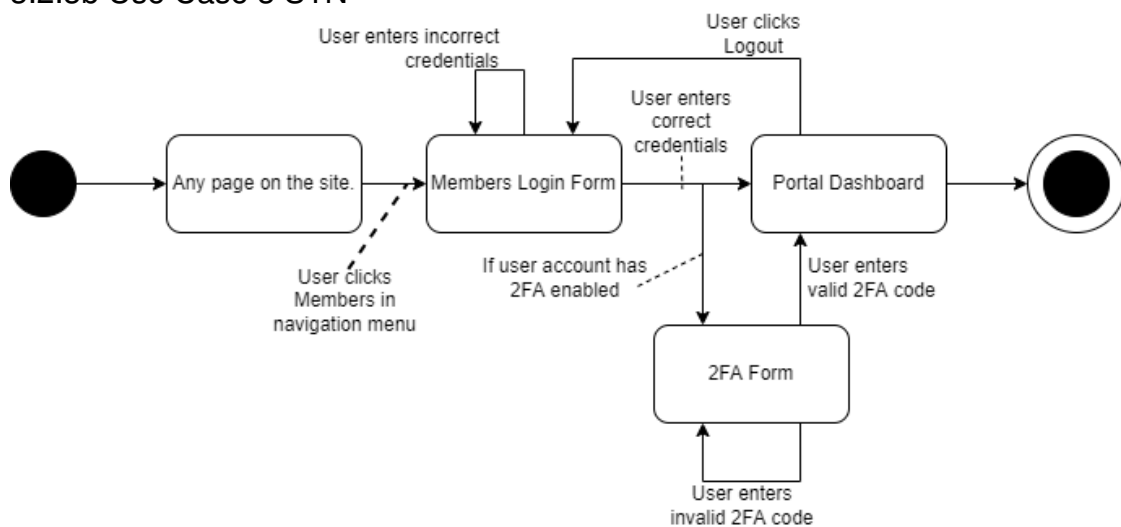
3.2.2b Use Case 2 STN



3.2.3a Use Case 3

Use Case No.3	Use Case Name: Log into Web Portal	Rating: Must
Purpose: A group member must be able to log into their account		
Main actor: Theatre Group Member	Secondary Actor: N/A	
Pre-conditions: An unauthenticated user is viewing the website on any page.		
Trigger: The user clicks the Members link in the navigation bar.		
Description: <ul style="list-style-type: none">1. The user is shown the log in form.2. The user enters their email and password.3. If credentials are correct, the user is logged into their account and is redirected to the dashboard.		
Extension: <ul style="list-style-type: none">1. If the user enters incorrect credentials, they are returned to the login page.2. If the users account has Two Factor Authentication enabled, they are shown a field to enter their 2FA code.<ul style="list-style-type: none">1. If the code is valid, they are logged in.2. If the code is not valid, they are redirected back to the code entry field.3. User can log out from dashboard and be redirected to login page.		
Post-Conditions: User is redirected to their Portal Dashboard.		
Author: Matthew Larby	Date: 24.02.2022	

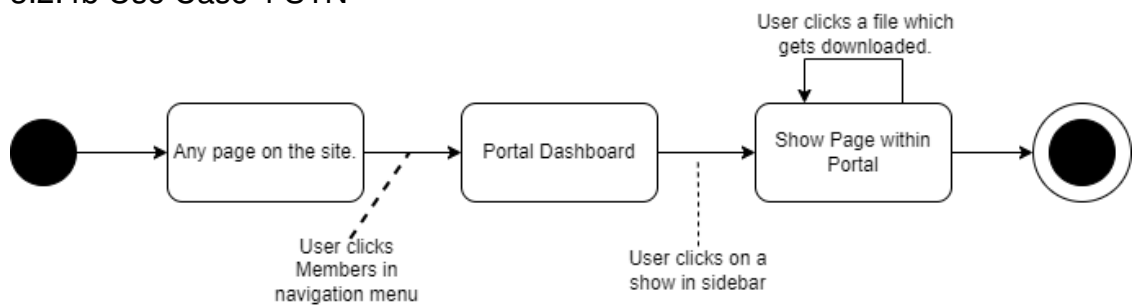
3.2.3b Use Case 3 STN



3.2.4a Use Case 4

Use Case No.4	Use Case Name: Access Shared Files	Rating: Must
Purpose: A group member must be able to access shared files for a show.		
Main actor: Theatre Group Member	Secondary Actor: N/A	
Pre-conditions: An authenticated user is on any page of the site.		
Trigger: The user clicks the on the Members link in the navigation bar.		
Description: <div><div>1. The user is shown their dashboard.</div><div>2. The user clicks on a show in the side bar.</div><div>3. The show page is displayed with a list of available files.</div><div>4. The user can download a file by clicking on it.</div></div>		
Extension: <div><div>1. N/A</div></div>		
Post-Conditions: <div>File is downloaded, user is redirected back to the show page.</div>		
Author: Matthew Larby	Date: 24.02.2022	

3.2.4b Use Case 4 STN

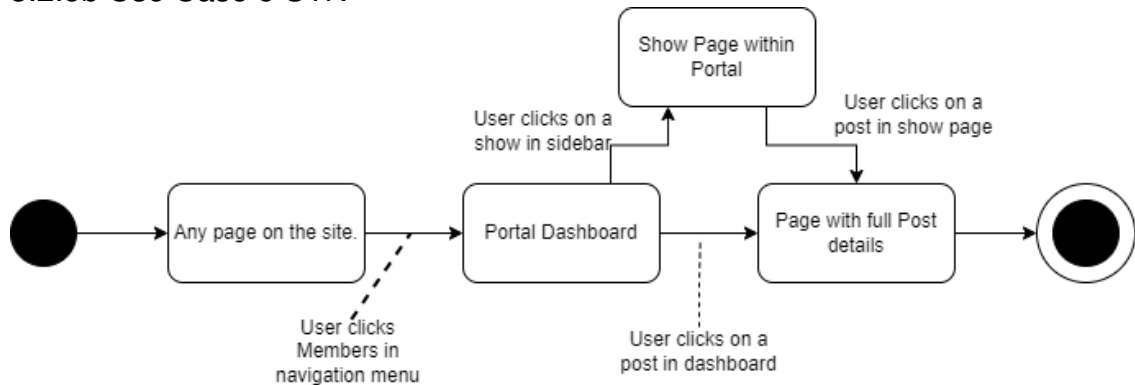


3.2.5a Use Case 5

Use Case No.5	Use Case Name: See Private Announcements	Rating: Must
Purpose: A group member must be able to see private announcements for a show.		
Main actor: Theatre Group Member	Secondary Actor: N/A	
Pre-conditions: An authenticated user is on any page of the site.		
Trigger: The user clicks the on the Members link in the navigation bar.		
Description: <ul style="list-style-type: none">1. The user is shown their dashboard.2. The user can click on a recent post from their dashboard to view the full post.		
Extension: <ul style="list-style-type: none">1. The user can click on a show in the side bar and see just the posts regarding a specific show.		
Post-Conditions: Post is shown in its own page.		
Author: Matthew Larby	Date: 24.02.2022	

Open AmDram Portal

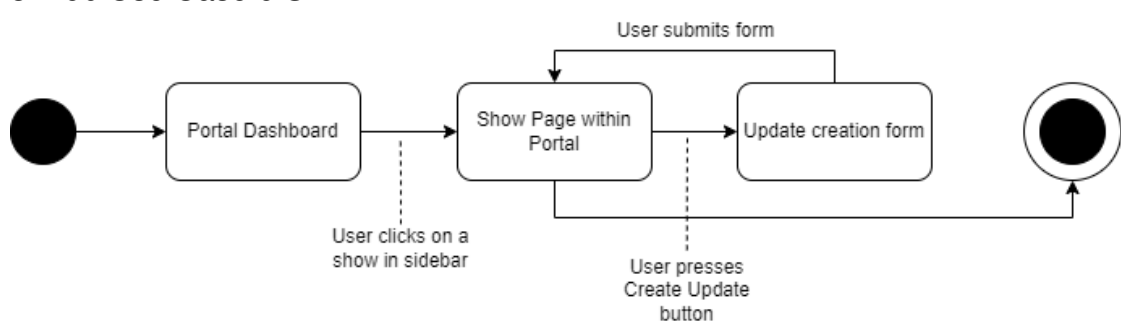
3.2.5b Use Case 5 STN



3.2.6a Use Case 6

Use Case No.6	Use Case Name: Post Updates	Rating: Must
Purpose: A director or producer member must be able to post announcements for a show.		
Main actor: Show Director/Producer	Secondary Actor: N/A	
Pre-conditions: A show director or producer is on their dashboard.		
Trigger: The user clicks the on the show that they wish to post an update about.		
Description: <ol style="list-style-type: none">1. The user is shown the show page.2. The user clicks on a Create Post button.3. The user is shown a post creation form with Title, Content, and Post Type.4. Once filled in, the form can be submitted, and the new post is published to the correct location.		
Extension: <ol style="list-style-type: none">1. N/A		
Post-Conditions: User is redirected back to the show page.		
Author: Matthew Larby	Date: 24.02.2022	

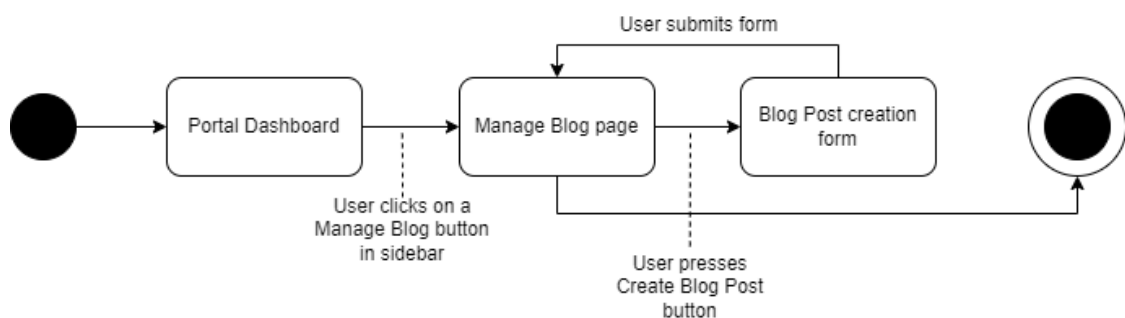
3.2.6b Use Case 6 STN



3.2.7a Use Case 7

Use Case No.7	Use Case Name: Post Blog Post	Rating: Must
Purpose: A blog writer must be able to write and publish blog posts.		
Main actor: Blog Writer		Secondary Actor: N/A
Pre-conditions: A blog writer is on their dashboard.		
Trigger: The user clicks the Manage Blog button in the dashboard.		
Description: <ol style="list-style-type: none"> 1. The user is shown the Manage Blog page. 2. The user clicks on a Create Blog Post button. 3. The user is shown a blog creation form with Title, Content, and Publish Date fields. 4. Once filled in, the form can be submitted, and the new blog post is published to the correct location. 		
Extension: <ol style="list-style-type: none"> 1. Upload a Word Document which will be converted to a new blog post. 		
Post-Conditions: User is redirected to the Manage Blog page.		
Author: Matthew Larby		Date: 24.02.2022

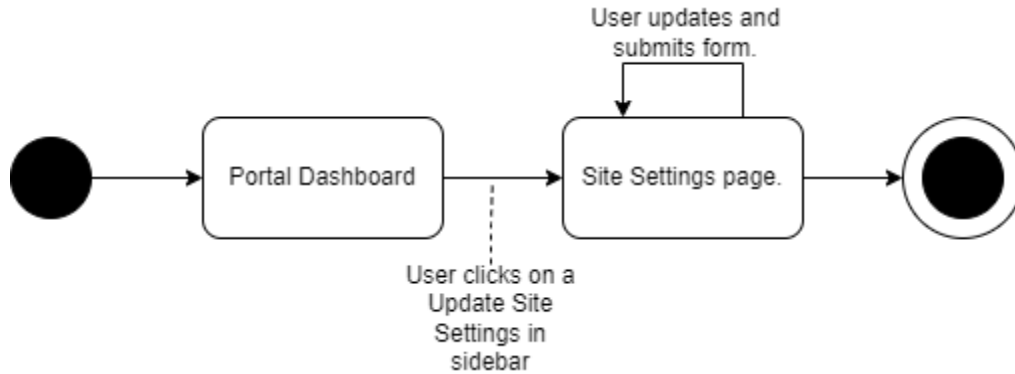
3.2.7b Use Case 7 STN



3.2.8a Use Case 8

Use Case No.8	Use Case Name: Update Site Settings	Rating: Should
Purpose: A site admin should be able to update the site settings from the web portal.		
Main actor: Webmaster/Admin		Secondary Actor: N/A
Pre-conditions: A Webmaster/Admin is on their dashboard.		
Trigger: The user clicks the Update Site Settings button in dashboard		
Description: <ol style="list-style-type: none"> 1. The user is shown the Site Settings page. 2. The user edits the form field of the setting they wish to update and submits it. 		
Extension: <ol style="list-style-type: none"> 1. N/A 		
Post-Conditions: User is redirected to the Site Settings page with the changes saved.		
Author: Matthew Larby		Date: 24.02.2022

3.2.8b Use Case 8 STN



3.2.9a Use Case 9

Use Case No.9	Use Case Name: Update Site Settings	Rating: Should
Purpose: A site admin should be able to invite members to create an account.		
Main actor: Webmaster/Admin		Secondary Actor: N/A
Pre-conditions: A Webmaster/Admin is on their dashboard.		
Trigger: The user clicks the Invite User button in dashboard		
Description: <ol style="list-style-type: none"> 1. The user is shown the Invite User page. 2. The user fills out the form with the details of the member that is being invited and submits it. 		
Extension: <ol style="list-style-type: none"> 1. N/A 		
Post-Conditions: User is redirected to a page that shows the invite link for being sent to the invitee.		
Author: Matthew Larby		Date: 24.02.2022

3.2.9b Use Case 9 STN



3.3 Technology Choice

3.3.1 Backend

For the webapps backend, the Python-based Flask micro-framework will be used along with various associated libraries for connecting to the database, handling web-sockets, and verifying forms. Flask was chosen due to factors like previous experience; how lightweight and modular it is compared to alternatives like Django; its great performance; and that has been used by many high-traffic websites such as Reddit, Netflix, Trivago, Patreon, AirBnB and Uber. [6]

To serve the webapp, research was done into two different WSGI (Web Server Gateway Interface) implementations to determine the best one to move forward with for deployment. The two chosen for further research were Gunicorn and uWSGI because of the number of deployment guides that are available.

During research, Gunicorn was found to have fewer configuration settings than uWSGI which makes it simple to work. This does not reduce functionality as the options it is missing tend to focus on deployments aimed at much larger amounts of traffic than the client's site is going to experience. Some performance testing of various WSGI implementations has previously been done by a company called AppDynamics [7]. In their testing, they noted that uWSGI had "poor results" in most of their tests which they attributed to a broken release. On the other hand, Gunicorn was found to be "A good, consistent performer for medium loads" which fits the needs of the client perfectly, so it was chosen for this project.

The webserver decision was based on the recommendation made in the documentation of Gunicorn. As the Deploying Gunicorn page states, "we strongly advise that you use Nginx" [8] and as that it is the only webserver they provide an example configuration file for, Nginx was chosen. Just as with the chosen WSGI, other alternatives are available to be used however it is recommended to use Nginx.

3.3.2 Database

When planning the project, a couple of different database types were considered and explored. The main decision to make was choosing between a SQL or a NoSQL type database.

SQL-type databases are heavily relationship oriented which aligns quite well with the data that the project will be storing, in particular the cast and crew archives for each show. Having the relationships formally defined within the database allows the front-end to easily query and display data from different perspectives such as the cast that appeared in a show versus the shows that a cast member has appeared in.

Another benefit of using an SQL-type database is how well it integrates with the chosen backend framework, Flask. Flask is built with a dependency of SQLAlchemy, a module designed to simplify the construction of SQL queries in Python, which means that a lot of its premade helper classes will only work with an SQL database. This module also has a lot of helpful built-in features, for example its integrated User model which simplifies authentication and confirming which user account is logged in.

On the other hand, NoSQL-type databases are very high-performance and lack the formally defined schemas that can slow down the development of SQL based projects which allows for on-the-fly data structure modifications when implementing a new feature. The downside of NoSQL is that it lacks the relationship links that make restructuring data in SQL queries so easy and makes the NoSQL queries complicated.

The final decision for the database technology was to use an SQL type database due to the relational nature of the data, the complex queries, prior experience, and compatibility with other parts of the project. As for the particular SQL database service, PostgreSQL was chosen for its regular updates, open-source development, and simplicity for deployment. In addition to this, the client already has a PostgreSQL database set up for some of the temporary features that were added to the existing website prior to the start of this project.

3.4 Site Accessibility

One of the major design considerations of the User Interface is how accessible it is. The target demographic for the website has quite a wide range which makes it important that no group gets excluded from being able to use the website as intended.

3.4.1 Text Contrast

When choosing text colour for a site, it is important to consider the contrast ratio between the text and its background, and to follow the recommendations set out in the Web Content Accessibility Guidelines [9]. For WCAG 2.0 level AA conformance, the minimum contrast ratio is 4.5:1 for normal text and 3:1 for large text and for WCAG 2.0 level AAA (the highest level), contrast ratio of at least 7:1 for normal text and 4.5:1 for large text. In designing the site, all text is should to conform to the minimum level AA guidelines with the majority of the site meeting the full AAA level requirements.

3.4.2 Font Size

For those who struggle with reading small text, no matter the contrast ratio, it is important to include a way to adjust the site wide font size to help with readability. To facilitate this, a 1x, 1.5x, and 2x font size adjustment buttons will be placed in the footer of the website for ease of access and provide font adjustments that track across every page in a user's session.

3.4.3 Site Theme (Positive vs Negative Contrast)

In separate discussions with both the client and project supervisor, the issue of text contrast was brought up particularly the issues relating to age when reading text with positive contrast (white text on dark backgrounds) which causes blurring of the text in an effect called "Halation". This effect decreases reading comprehension speeds for older age groups (those aged 45+) which accounts for a considerable proportion of the theatre group patrons. [10]

To improve the readability whilst maintaining a modern style, the site will automatically detect the user's device preferences and adjust the theme accordingly, while also providing an option to change the theme manually in the accessibility section of the footer. Just like the font size adjustment, this manual change will also track across the entire session.

3.4.4 Device Compatibility

Since 2017, mobile devices have accounted for over half of all the web traffic worldwide [11] which means that designing a site to have mobile compatibility is no longer optional. It is now a requirement to build a working and tested layout specifically tailored to mobile devices and desktops. Using responsive design techniques, the site will detect the type of device and adjusts the

layout and design automatically including a hamburger-style navigation menu to reduce screen clutter, vertically arranged elements to allow more words per line for faster reading comprehension, and larger fonts to adjust for the much higher Pixel Density of mobile screens in contrast with desktop displays.

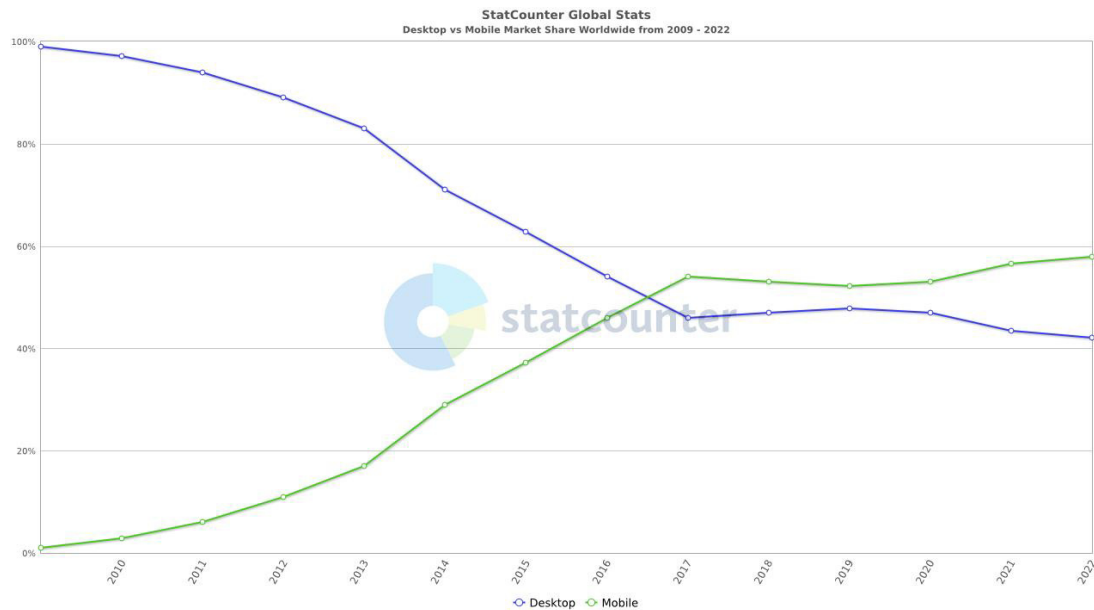


Figure 4: Website Traffic by Device Type (Mobile, Desktop) 2009-2022 [11]

4. Implementation

4.1 Web Portal Authentication

For granting access to the restricted functions of the web portal, a user needs to be able log in. To this end, a simple username and password login form was implemented using the SHA256 hash algorithm to verify the password against the hashed version stored in the database.

In the modern era of the internet, simply securing a login with a password is often considered insufficient as they can be brute-forced or collected with social engineering methods (such as phishing scams). The best way to provide an extra layer of security for a login form is to enable Two Factor Authentication (2FA). 2FA relies on having a second form of authentication of a different type to a password, the three types being "something you know" like a password or security question, "something you are" like biometrics including fingerprints, or "something you have" like a physical device that can generate a specific code. The form of 2FA that is most user friendly with the widest device compatibility is the third option, "something you have". This method allows a 2FA code generator app, like Google Authenticator, to be used to create a one-time code that is entered into the login form with the password.

To achieve this, the project will use a time-based code as it is less susceptible to phishing attacks compared to a counter-based codes due to the time limit on the validity of each code. This keeps the users more secure and prevents a desynchronisation issue between code generator and the website.

```

@app.route("/members/otp", methods=["GET", "POST"])
def otp():
    if request.method == "GET":
        if 'email' not in session:
            return redirect(url_for('members'))
        return render_template("otp.html", css="members.css")
    else:
        user = User.query.filter_by(email=session['email']).first()
        totp = pyotp.TOTP(user.otp_secret)
        session.pop('email', None)
        if totp.verify(request.form['otp']):
            login_user(user)
            return redirect(url_for("frontpage"))
        else:
            return redirect(url_for("members"))

```

Figure 5. The route that handles the 2FA code verification.

Collects the user input from a form that a user with 2FA enabled is redirected to upon correct password entry. The entered code is then verified by the TOTP class that was initialised with the user's two factor secret that is stored with their account.

4.2 Database

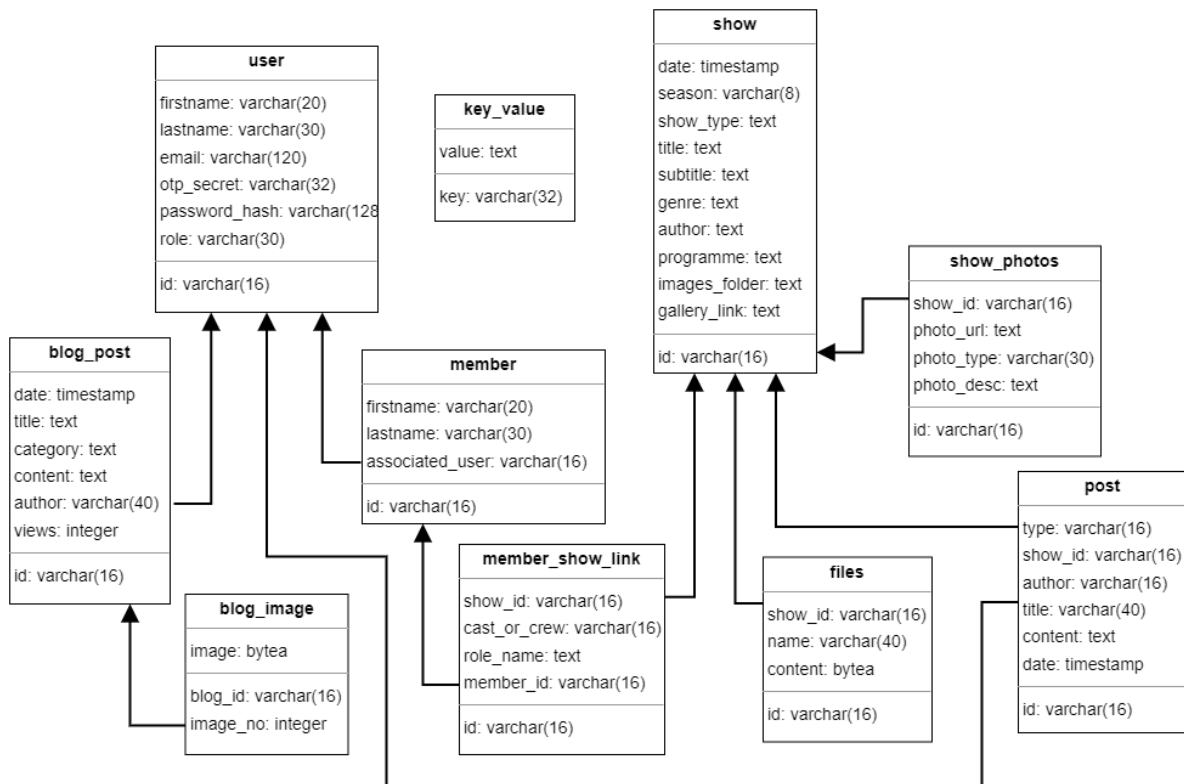


Figure 6: Database Relationship Diagram

To store the details about a particular show, the Show table was created with the expected attributes like "title", "genre", and some date attributes. The ID field of the Show table is then used to link together any other table that contains show related information such as any files being shared, announcements/update posts for the website, or the photos to be displayed in the 'Past Shows' section of the site.

The ShowPhotos table is used to help reduce the number of API calls made to Google Photos by storing the image identifier string for making a direct request when it gets displayed on the website. This was the best approach to storing the photos as is outlined in section 4.3 Google Photos API.

The User table is handled a differently to most webapps as it is used to group together the Member entries for those who have performed under different names (changing their last name when they got married etc.) and not every performer may need to be able to log into the website (those who have moved away and left the group or passed away) but consolidating all these possible use cases into specific states of the User table simplifies the database schema and helps reduce the complexity of any queries involving members.

The Posts table handles all of the public announcements, auditions notices, and the private posts that only get shared with authenticated users. It stores the basic data needed to distinguish and display the posts in the correct locations on the site.

Similar to the Posts table, the BlogPost table contains the data for the blog that a member of theatre group runs. This table has been brought over from the temporary blog that was set up on their previous site while this project was in development. Considering the similarity between the BlogPost and Post tables, this area has been identified for improvement in the Future Work section.

The Members table stores a list of all the names as they appear in any programmes and links them to a User account if there are multiple names for one member. Using this table also allows for members who do not need a full User profile to still be credited in the Show Archives.

Connecting Members with the Shows they involved in is done in the MemberShowLink table. This table connects the member_id and show_id and also contains more information about what role the member had in that particular show. Being able to link a role back to the User account will allow a member to update their personal details such that their correct name will be reflected on the site. In the case where the name on a programme differs from that stored in the associated User account, the account name is shown with the older record being shown alongside in a style inspired by IMDb.

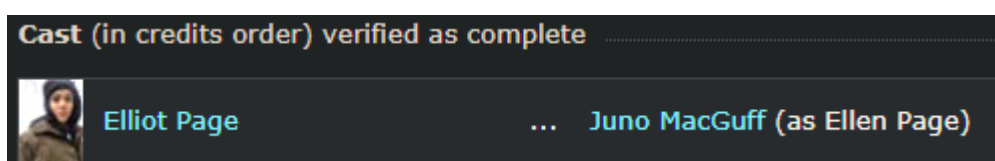


Figure 7. Cast Member Name Change in IMDb

Julia Simmons	Tristan Quittenton (as Scarlett Quittenton)
---------------	---

Figure 8. Cast Member Name Change in this project

There are also a handful of small tables that only store specific data. The KeyValue table is a simple variable storage section for handling admin options that need to be kept after a server restart. The BlogImage table stores the images uploaded in the blogs which are served alongside the text content. The Files table stores files such as scripts that are being shared for each show.

4.3 Google Photos API

When implementing the code that links the show archives to the photos that the client has stored on their Google Photos account, a variety of issues were faced.

4.3.1 API Authentication with OAuth

The first issue that came up when interfacing with the Photos API was the lack of support for any “server to server” authentication. Most of Google’s APIs have three modes of authentication, an API key, a “Service Account”, and OAuth, the first two are aimed at server-side applications whereas the last is typically for client-side ones. For this project, a lot of the logic is handled server-side to reduce load times, so either the API key or a service account would be ideal, however, the specific Google Photos API only supports OAuth. This limitation meant that updating the photos on the site has to be performed by someone with access to the clients Google account and requires user input in order to perform the authentication which makes it harder to perform API queries in the background.

The reason OAuth is the only supported API authentication method is the data in Google Photos that gets exposed when authenticated could be of a sensitive nature as it is entirely user generated content in contrast to published publicly accessible data like that of the Google Books API which can be used with the other two authentication methods. OAuth ensures that the owner of the account being accessed has given permission and provides transparency surrounding what permissions have been requested from the user.

4.3.2 BaseURL Expiry Documentation

The first attempt to store photos in the database without keeping the raw image data, required storing the BaseURL, an endpoint on Googles servers that returns the image associated with the link. In initial testing, successfully displayed the images on the site, however, when demonstrating the site to the project supervisor, the images later stopped loading with just 403 Forbidden errors.

After nearly a day of investigating the aforementioned errors, it became apparent that issue was the image BaseURL only being valid for 1 hour from generation which mandated a change in what data was stored.

4.3.3 Re-Authenticating without an OAuth Pop-up

One issue with not being able to store BaseURLs for images is that getting any images will always require an API call which necessitates finding a method for getting it authenticated without going through the full OAuth access request.

When using OAuth, a successful login will return an "Access Token" which will authenticate any API call for the just next hour. As this token has an expiry time, it means that it is not suitable for getting the images as that would require an OAuth login once an hour.

After plenty of documentation reading, a parameter was found that can be added to the initial OAuth login that will generate a "Refresh Token" alongside the "Access Token". This new "Refresh Token" allows an application to make another request to Google's authentication servers to generate a new "Access Token" without requiring user input. These "Refresh Tokens" do eventually expire, and can be manually revoked by the account holder, but the expiration only happens after 6 months of in-activity which usually will not happen on a public website.

```
@app.route("/members/set_show_photos/oauth")
def oauth():
    if "localhost" in request.url_root:
        redirect_url = request.url_root + "members/set_show_photos/form"
    else:
        redirect_url = str(request.url_root + "members/set_show_photos/form").replace("http://", "https://")
    refresh_time = KeyValue.query.filter_by(key="refresh_time").first()

    if (refresh_time is not None) and (int(refresh_time.value) > int(time.time())):
        client_id = app.config['g_client_id']
        client_secret = app.config['g_client_secret']

        auth_endpoint = "https://accounts.google.com/o/oauth2/v2/auth"
        scope = "https://www.googleapis.com/auth/photoslibrary"

        oauth2_url = f"{auth_endpoint}?" \
            f"scope={scope}&" \
            f"access_type=offline&" \
            f"include_granted_scopes=true&" \
            f"response_type=code&" \
            f"state={redirect_url}&" \
            f"redirect_uri={redirect_url}&" \
            f"client_id={client_id}"

        return redirect(oauth2_url)
    else:
        return redirect(redirect_url)
```

Figure 9. The route redirecting the user to the OAuth login screen.


```

@app.route("/members/set_show_photos/form", methods=["GET", "POST"])
def choose_album():
    client_id = app.config['g_client_id']
    client_secret = app.config['g_client_secret']

    if request.method == "GET":
        if request.args.get("code") is not None:
            url = "https://oauth2.googleapis.com/token"
            data = {
                "code": request.args.get("code"),
                "client_id": client_id,
                "client_secret": client_secret,
                "redirect_uri": request.args.get("state"),
                "grant_type": "authorization_code"
            }

            x = requests.post(url=url, data=data)
            access_token = x.json().get("access_token")
            refresh_token = x.json().get("refresh_token")
            if (db_refresh := KeyValue.query.filter_by(key="refresh_token").first()) is not None:
                db_refresh.value = refresh_token
                KeyValue.query.filter_by(key="refresh_time").first().value = int(time.time())
            else:
                new_token = KeyValue(
                    key="refresh_token",
                    value=refresh_token
                )
                new_time = KeyValue(
                    key="refresh_time",
                    value=int(time.time())
                )
                db.session.add(new_token)
                db.session.add(new_time)
            db.session.commit()

```

Figure 10. Authorisation code exchange.

Figure 10 shows the section of the API interaction that exchanges the authorisation code for Access and Refresh tokens which are collected and stored for future use.

4.3.4 The Final Implementation

In the end, the optimal solution was to store the permanent ID attribute in the database for each photo that needs to be displayed. The ID can then be used to generate a new BaseURL on demand through the Photos API. This does require an API call per photo being loaded on a webpage but the additional loading times from API calls can be mitigated with other load time reductions, for example Section 4.4 Fast Photo Gallery.

```

data = {
    "albumId": request.form.get("album"),
    "pageSize": 100,
    "access_token": access_token
}

url = f"https://photoslibrary.googleapis.com/v1/mediaItems:search?access_token={access_token}"
z = requests.post(url, data).json()

next_token = "not none"
photos = []
while next_token is not None:
    for photo in z.get("mediaItems"):
        # print(photo.get("id"))
        photo_tuple = (
            list({"photo", "video"}.intersection(set(photo.get("mediaMetadata").keys()))[0],
            photo.get("id"),
            f"{photo.get('mediaMetadata').get('width')},{photo.get('mediaMetadata').get('height')}",
            )
        # print(photo_tuple)
        photos.append(
            photo_tuple
        )
    # print(len(photos))
    data["pageToken"] = (next_token := z.get("nextPageToken"))
    z = requests.post(url, data).json()

```

Figure 11. Image Collection

Figure 11 shows the API code that collects the ID, media type, and dimensions for storing in the database. The results from the API are paginated so this code loops until all pages of results have been processed.


```

existing_sql = ShowPhotos.query \
    .filter_by(show_id=request.form.get("show")) \
    .all()

used_ids = [value[0] for value in ShowPhotos.query.with_entities(ShowPhotos.id).all()]
existing = set([(i.photo_type, i.photo_url, i.photo_desc,) for i in existing_sql])

to_remove = existing.difference(set(photos))
to_add = set(photos).difference(existing)

for photo in to_add:
    new_photo = ShowPhotos(
        id=(new_ID := corha.rand_string(photo[1], 16, used_ids)),
        show_id=request.form.get("show"),
        photo_url=photo[1],
        photo_type=photo[0],
        photo_desc=photo[2]
    )

    used_ids.append(new_ID)
    db.session.add(new_photo)

for photo in to_remove:
    ShowPhotos.query \
        .filter_by(
            show_id=request.form.get("show"),
            photo_url=photo[1],
            photo_type=photo[0]
        ) \
        .delete()

db.session.commit()

```

Figure 12. Logic for efficiently adding photos to the database.

Figure 12 shows how Sets are used to calculate the difference between what is currently stored and what the API returns in order to reduce the number of database operations when updating the media items for a specific show.

4.4 Fast Photo Gallery

As the client has over two hundred photos attached to many of their shows, loading all of those images takes over 10. Obviously, this is not the best user experience, so some optimisation must be done to reduce the time each page takes to load in.

In order to facilitate an apparent instant loading of the images in the gallery without preloading each of them, the gallery element must be capable of fetching the next image in the queue as it gets requested by the user without showing a blank space while the image data is being downloaded.

By preloading just five of the images, the gallery can use two images either side of the currently displayed one as a buffer with loaded images that can be displayed instantly to the user while the next unloaded image in the gallery gets loaded in the background. The images are loaded on either side of the

Open AmDram Portal

starting one as the gallery supports wraparound view (i.e., going from image 1/200 straight to 200/200 by clicking the left arrow).

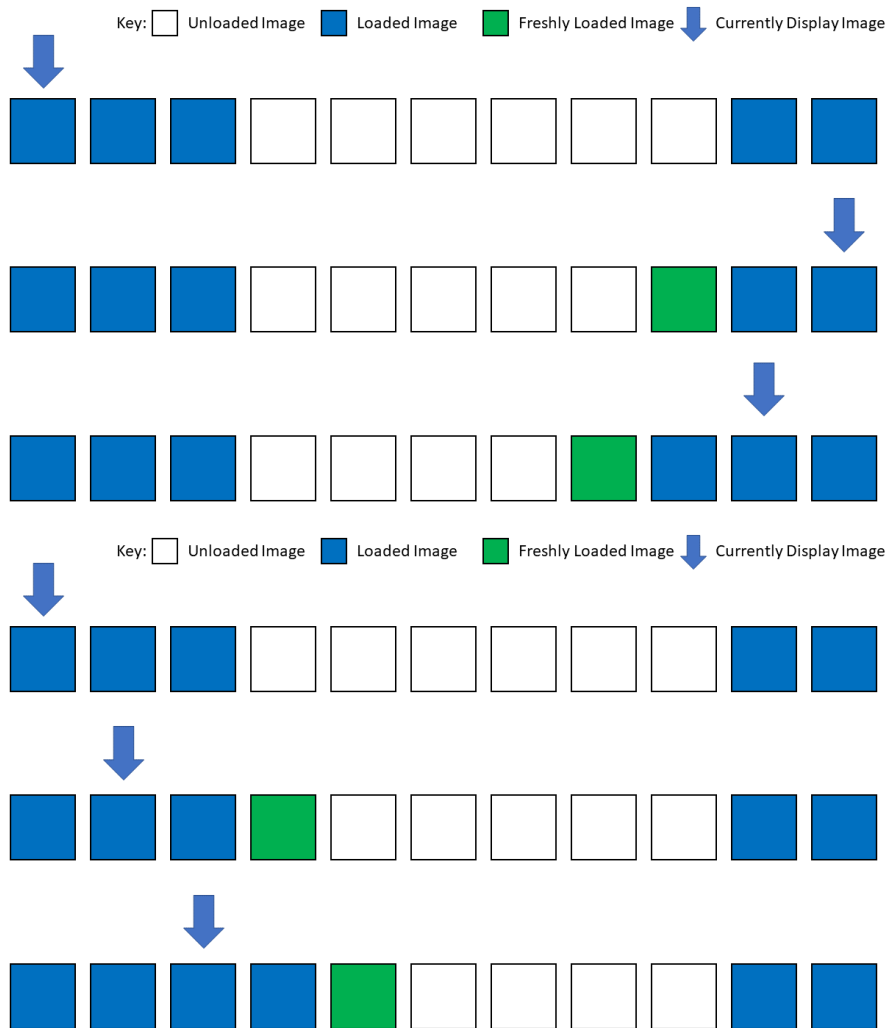


Figure 13: Visual representation of the gallery.

Figure 13 shows an example ten-image gallery and how images get loaded when being navigated in either direction.

By using more than one image as the buffer, the user is less likely to see a gap if one image is particularly slow to load while still keeping the initial load time for the page low.

```

let currentPhoto = 1

function galleryIncr(n) {
    document.querySelector(selectors: "#photo"+currentPhoto).classList.add("img-hidden")
    currentPhoto = currentPhoto + n
    if (currentPhoto > photoMax) {currentPhoto = 1}
    if (currentPhoto < 1) {currentPhoto = photoMax}
    loadImg(currentPhoto, n)
    document.querySelector(selectors: "#photo"+currentPhoto).classList.remove(tokens: "img-hidden")
    document.querySelector(selectors: "#photocount").innerHTML = currentPhoto
}

function loadImg(id, n) {
    id += 2*n
    if (id > photoMax) {id = id - photoMax}
    if (id < 1) {id = photoMax + id}
    let toLoad = document.querySelector(selectors: '#photo'+id)
    if (toLoad.src === ""){
        toLoad.src = toLoad.dataset.dataSrc
    }
}

```

Figure 14. The JavaScript gallery element.

The galleryIncr() function brings the correct image into view while hiding all others in the gallery and supports increasing and decreasing the current image number by one. It also handles “wrapping around” from the final image back to the first and vice versa.

The loadImg() function handles the actual loading of an image from its URL at a later point than on page load using custom HTML attributes that interface with the image element dataset property. It also houses the logic for loading two images in advance (See Figure 13 for a visual explanation of this feature) and only in the direction that the user is navigating within the gallery including the support for wrapping around.

5. Results and Evaluation

5.1 Results

5.1.1 The Prototype

As part of the testing phase, the last version of the solution that was made during the time scale of this project has been hosted as an internet accessible site. This version was given its own URL (See Appendix) which will not be updated with any future improvements and should stay online for at least 6 months for anyone reading this report.

As that version of the project will have a test login (See Appendix) within this report, the database it is connected to will also be independent of any future development or data entry.

5.1.2 Measurable Results

The measurable aim of the project was to reduce the webpage loading times compared to the client’s previous solution. The statistics that follow were generated from mobile-based tests however similar differences between the

Open AmDram Portal

two solutions do occur on desktop as well. The full set of results can be found below.

Figure 15 shows a comparison of the load times of the two websites. It shows how the new website has a speed index (“average time at which visible parts of the page are displayed” [12] half that of the clients existing site. In fact, the new site is able to load before the first byte of data from the existing one has even been received.



Figure 15. Sites Speed Comparison (Generated with WebPageTest.org)

Another test that was generated was a side-by-side video of the two sites loading simultaneously (Figure 16) which is a visual comparison that highlights the difference between the two solutions. Part of the comparison video is the stopwatch measuring the seconds that each site takes to load which puts into perspective the real-world differences between the two solutions that would be harder to perceive if testing by loading one after the other and shows how the new solution is able to be fully loaded before the old site has even begun to draw content on the screen.

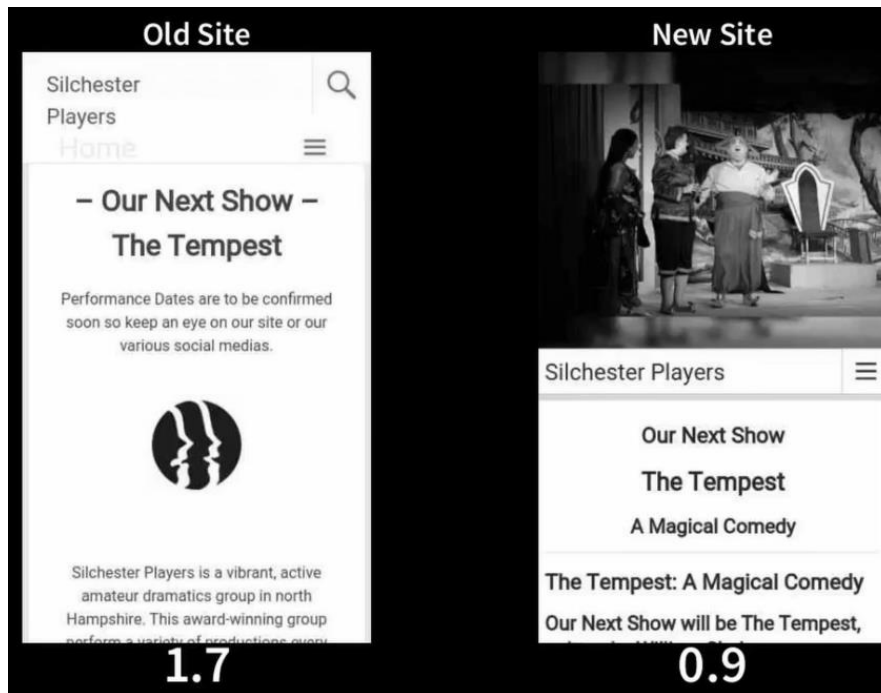


Figure 16. Still from Load Speed Test (in seconds)

<https://youtu.be/ft76G32-akc>

5.1.3 Testing

All tests were conducted on desktop and mobile and focused on verifying the functionality of each purpose. Screenshots were gathered along with video of the desktop testing as evidence for the test cases.

Video Evidence of the Test Cases <https://youtu.be/q-hf8c8pCwA>

5.1.3.1 Test 1


Test Case ID: 01	Test Purpose: Verify a Public Site Visitor can read the blog posts.		
Pre-requisites: <ul style="list-style-type: none">User is viewing the website from any page.			
Test Steps:	Expected Results	Actual Results	Pass / Fail
<ul style="list-style-type: none">The user clicks the Blog link in the navigation bar	User is redirected to a list of the blog posts.	User is redirected to a list of the blog posts.	Pass
<ul style="list-style-type: none">User clicks on a blog post to read.	User is redirected to the page that shows that specific blog post.	User is redirected to the page that shows that specific blog post.	Pass
<ul style="list-style-type: none">User presses on page Back Button to return to the list of blog posts	User is redirected to a list of the blog posts.	User is redirected to a list of the blog posts.	Pass
Author: Matthew Larby		Date: 23-05-2022	

Silchester Players
Home
Blog
Pas

Back to all posts

Some Time Off

May 2022
by Stephen Bibby



Silchester Players

May has been an unusually quite month for Silchester Players. Normally it is a very busy time when we put on a play with rehearsals commencing after the end of our January pantomime. Unfortunately, this year, because Covid delayed the panto, we have not had the time and space to mount the usual May production.

This has meant that the whole company has been able to take rest before we start work on Shakespeare's *The Tempest* for which rehearsals begin in the first week of July. Prior to that we will be holding our AGM on 28 June. Once again this will be via Zoom, both because of Covid and also because this medium saves us the cost of room hire.

Although there has been a lapse in production preparation, our committee has still been meeting and actively discussing a number of challenges facing us. Perhaps it is not widely known, but when we have a surplus over production costs we make charitable donations. Thus, by supporting Silchester Players audience members are also supporting good causes. However, following the significant loss of income during the Covid pandemic, when we were not on stage for over 2 years, and in the face of steeply rising prices we need to review our approach to philanthropic activity. We very much hope it can continue, but like other organisations, we are likely to be facing some tough financial decisions over the coming year and need to adjust accordingly.

We remain grateful for the loyal following we have in Silchester and nearby communities and sincerely hope that this continues. We are eagerly looking forward to our autumn production of *The Tempest* when we aim to bring Shakespeare's last play to life in a colourful, amusing and easily understandable way. Make a forward note of performance dates 23/24 September and 30 September /1 October and watch for further information in this magazine, on our website <https://silchesterplayers.org> or follow us on Facebook.

Figure 17. Test Case 1 Step 3 Screenshot

5.1.3.2 Test 2

Test Case ID: 02	Test Purpose: Verify a Public Site Visitor can browse the Past Shows Archive		
Pre-requisites: <ul style="list-style-type: none">User is viewing the website from any page.			
Test Steps:	Expected Results	Actual Results	Pass / Fail
<ul style="list-style-type: none">The user clicks the Past Shows link in the navigation bar.	User is redirected to a list of the past shows.	User is redirected to a list of the past shows.	Pass
<ul style="list-style-type: none">The user clicks on the “Aladdin 2022” show to see more details.	User is redirected to the page that shows details about that show.	User is redirected to the page that shows details about that show.	Pass
<ul style="list-style-type: none">User clicks an arrow on the image gallery	User is shown a new image in the gallery.	User is shown a new image in the gallery.	Pass
<ul style="list-style-type: none">User presses play on video content.	Video content begins to play.	Video content begins to play.	Pass
<ul style="list-style-type: none">User clicks on a cast or crew member name.	User is redirected to a list of shows that member was involved in.	User is redirected to a list of shows that member was involved in.	Pass
Author: Matthew Larby		Date: 23-05-2022	

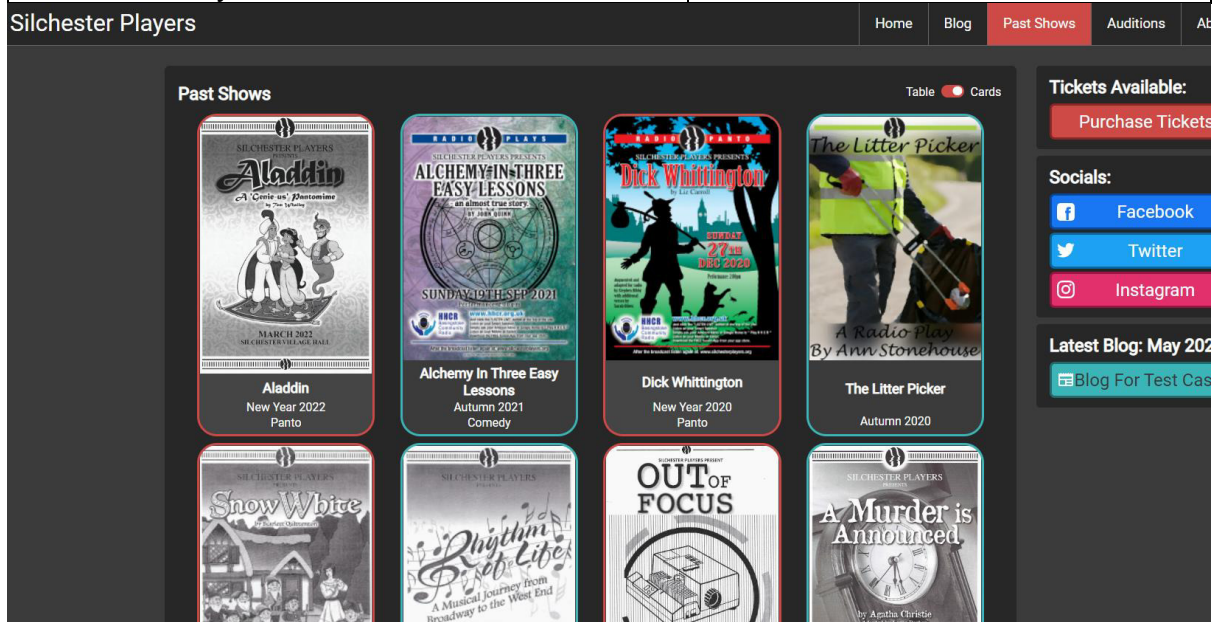


Figure 18. Test Case 2 Step 1 Screenshot

5.1.3.3a Test 3a

Test Case ID: 03a	Test Purpose: Log into Web Portal without 2FA		
Pre-requisites: <ul style="list-style-type: none">User is viewing the website from any page.			
Test Steps:	Expected Results	Actual Results	Pass / Fail
<ul style="list-style-type: none">The user clicks the Members link in the navigation bar	User is redirected to the members login form.	User is redirected to the members login form.	Pass
<ul style="list-style-type: none">User submits “test@example.com” and “test” as email and password respectively	User is redirected to their dashboard	User is redirected to their dashboard	Pass
Author: Matthew Larby		Date: 23-05-2022	

5.1.3.3b Test 3b

Test Case ID: 03b	Test Purpose: Log into Web Portal with 2FA		
Pre-requisites: <ul style="list-style-type: none">User is viewing the website from any page.			
Test Steps:	Expected Results	Actual Results	Pass / Fail
<ul style="list-style-type: none">The user clicks the Members link in the navigation bar	User is redirected to the members login form.	User is redirected to the members login form.	Pass
<ul style="list-style-type: none">User submits "test2fa@example.com" and "test" as email and password respectively	User is redirected to the 2FA code form.	User is redirected to the 2FA code form.	Pass
<ul style="list-style-type: none">User enters the 2FA code generated with this secret key "5PBOZDRUNHQKX243J SR44R56MQTGV2R7"	User is redirected to their dashboard	User is redirected to their dashboard	Pass
Author: Matthew Larby		Date: 23-05-2022	

5.1.3.4 Test 4

Test Case ID: 04		Test Purpose: Access Shared Files	
Pre-requisites:			
• An authenticated user is on any page of the site			
Test Steps:	Expected Results	Actual Results	Pass / Fail
• The user clicks the Members link in the navigation bar	User is redirected to their dashboard	User is redirected to their dashboard	Pass
• User clicks on a show in the side bar (Aladdin 2022)	User is redirected to the selected show page.	User is redirected to the selected show page.	Pass
• User clicks a file to download	File is downloaded	File is downloaded	Pass
Author: Matthew Larby		Date: 23-05-2022	

File name:

ALADDIN_2021_REHEARSAL-COPY-1 (4).pdf

Save as type:

PDF File (*.pdf)

Hide Folders

Save

Cancel

Files

ALADDIN_2021_REHEARSAL-COPY-1.pdf Delete

Figure 19. Test Case 4 Step 3 Screenshot

5.1.3.5 Test 5

Test Case ID: 05		Test Purpose: See Private Announcements		
Pre-requisites: <ul style="list-style-type: none">An authenticated user is on any page of the site				
Test Steps:		Expected Results	Actual Results	Pass / Fail
<ul style="list-style-type: none">The user clicks the Members link in the navigation bar		User is redirected to their dashboard where the latest announcements are shown	User is redirected to their dashboard where the latest announcements are shown	Pass
<ul style="list-style-type: none">User clicks on an announcement		User is redirected to the detailed announcement page	User is redirected to the detailed announcement page	Pass
Author: Matthew Larby			Date: 23-05-2022	

Dashboard

Shows

The Tempest
Aladdin
Alchemy In Three Easy Lessons
Dick Whittington

Blog Tools

Manage Blog
Create New Blog

Dashboard

Aladdin

Test Case 6

27 May 2022

Test for video

Figure 20 Test Case 5 Step 1 Screenshot

5.1.3.6 Test 6

Test Case ID: 06		Test Purpose: Post Updates	
Pre-requisites: <ul style="list-style-type: none">An authenticated user with at least director/producer privileges is on their dashboard			
Test Steps:	Expected Results	Actual Results	Pass / Fail
<ul style="list-style-type: none">User clicks on a show (Aladdin 2022)	User is redirected to the selected show page.	User is redirected to the selected show page.	Pass
<ul style="list-style-type: none">User clicks Create Post button	User is redirected to a post creation form	User is redirected to a post creation form	Pass
<ul style="list-style-type: none">User fills out Create Post form and submits it.	User is redirected back to the show page where new post is shown	User is redirected back to the show page where new post is shown	Pass
Author: Matthew Larby		Date: 23-05-2022	

Figure 21 Test Case 6 Step 3 Screenshot

5.1.3.7 Test 7

Test Case ID: 07		Test Purpose: Post Blog Post	
Pre-requisites: <ul style="list-style-type: none">An authenticated user with at least Blog Writer privileges is on their dashboard			
Test Steps:	Expected Results	Actual Results	Pass / Fail
<ul style="list-style-type: none">The user clicks the Manage Blog button in the sidebar	User is redirected to the Manage Blog page	User is redirected to the Manage Blog page	Pass
<ul style="list-style-type: none">User clicks Create Blog Post	User is redirected to the blog creation form.	User is redirected to the blog creation form.	Pass
<ul style="list-style-type: none">User fills out and submits the blog creation form.	User is redirected to the Manage Blog page showing with new blog post added	User is redirected to the Manage Blog page showing with new blog post added	Pass
Author: Matthew Larby		Date: 23-05-2022	

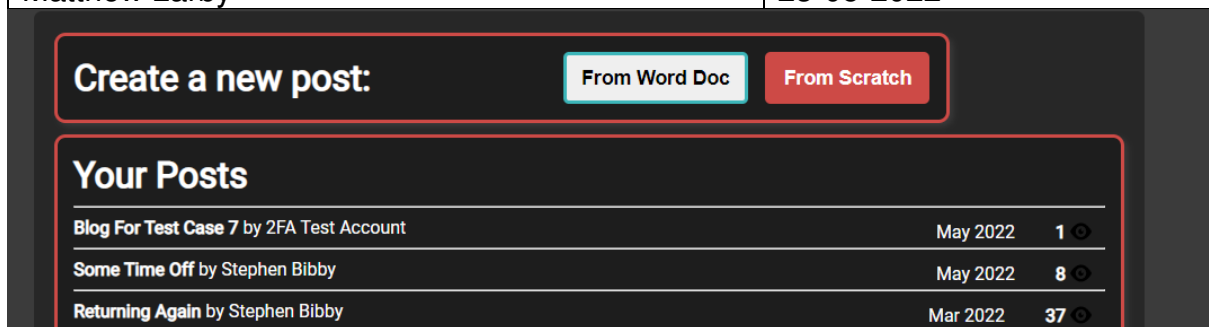


Figure 22. Test Case 7 Step 3 Screenshot

5.1.3.8 Test 8

Test Case ID: 08		Test Purpose: Update Site Settings	
Pre-requisites: <ul style="list-style-type: none">An authenticated user with Webmaster/Admin privileges is on their dashboard.			
Test Steps:	Expected Results	Actual Results	Pass / Fail
<ul style="list-style-type: none">The user clicks the Update Site Settings button in the sidebar	User is redirected to the site settings page.	User is redirected to the site settings page.	Pass
<ul style="list-style-type: none">User updates some settings and submits the form.	Popup confirming changes have been saved appears.	Popup confirming changes have been saved appears.	Pass
Author: Matthew Larby		Date: 23-05-2022	

Admin Settings

SITE NAME
Silchester Players 2

TICKETS URL
https://silchester-players.square.site/shop/all-performances/8

SOCIAL LINKS (COMMA SEPERATED)
https://www.facebook.com/silchplayers/,twitter.com/silchplayers, www.instagram.com/silchesterplayers

Figure 23. Test Case 8 Step 1 Screenshot

5.1.3.9 Test 9

Test Case ID: 09		Test Purpose: Create User Invite	
Pre-requisites: <ul style="list-style-type: none">• An authenticated user with Webmaster/Admin privileges is on their dashboard.			
Test Steps:	Expected Results	Actual Results	Pass / Fail
• The user clicks the Invite User button in the sidebar	User is redirected to the Invite User form.	No such button or resulting page exists.	Fail
• User fills out and submits Invite User form.	User is redirected to a page that shows the invite link for being sent to the invitee.	N/A – Unable to test due to previous test step failing.	Fail
Author: Matthew Larby		Date: 23-05-2022	

5.1.4 Project Accomplishments

One of the most ambitious parts of the project was creating a 'Past Shows' archive feature similar to IMDb. This feature has been successfully implemented and the site is capable of showing all the shows from the database, the cast and crew members involved in each show and the photos taken for each show. The cast and crew lists for each show also provide links to pages for each member that will list all of the shows they have been credited in.

Another ambitious goal of the project was the replacement of the discontinued Picasa galleries that the client used to display their images up to 2016. Interfacing with the Google Photos API for collecting and loading the images was quite a challenge but displaying images is one of the most important features for a theatre group's website as it helps show what their performances look like and can contribute to selling more tickets. The final implementation of the API interaction has shown to be stable during testing across multiple network types (Residential broadband, Mobile data, and Business internet connections) and no longer stops working after any fixed period of time.

Another advantage of interfacing with Google Photos is that video content can also be added to the Google Photos galleries and will be properly displayed on the website. This is all handled automatically when syncing a Google Photos album to a specific show on the website. As video content is not mixed in with the photos on the page, it is clear when a video is available to watch.

The gallery element itself was also successfully implemented with cross-device compatibility, working on both desktop and mobile, and loads the images rapidly across those devices. One aspect of the gallery that should be improved in the future is a full screen view (see 6. Future Work).

Due to the limited time scale of this project, some areas of the site are less polished than would be desired in a fully finished version, however, as those areas are functional, writing the style rules is more of a nice-to-have and can be assigned as future work.

5.2 Evaluation

On reflection, the initial goals of the project, whilst correct, were ambitious in relation to the time scale and the level of polish with which the project was aiming for.

Have functioning implementations of the main requirements that were set out in the beginning means that a major milestone has been achieved by this project in that a working minimum viable product has been produced and exceeded.

Any features that were dropped in the interest of having a working prototype finished were those that only affected the user interface or facilitated the rollout or deployment of using the projects solution in the real world such as the admin-managed user registration process.

For the members-only section of the site, the majority of functions within the portal itself are working. Access to the portal for those with user accounts is

working and even supports Two Factor Authentication but the registration and management of said user accounts is still to be fully completed. This is a minor missing feature as it would take less than a week to implement.

In relation to the publicly available section of the website, all the functions have been completed including the constantly updating announcements pages that always show the newest and most relevant posts as well as the past shows archive which successfully displays all the entered past shows (36 have been entered into the demo site) and links together all the shows by the members of the group who have appeared in them. In addition to that, the show pages fully support the displaying of visual media from albums stored in Google Photos.

6. Future Work

6.1 Consolidation of Similar Database Tables

During the report writing, it was noticed that the BlogPost and the announcement Post tables had remarkably similar fields which makes them an ideal candidate for consolidating into a single table. This is not quite as simple as copying all the data from one into another, however, as every database query in the backend of the website that refers to either of these table would need to be updated to make sure that no blog posts start appearing in places where only announcement posts are supposed to and vice versa. This issue has shown that more planning of the database tables should have taken place in the lead up to the actual development of features as the "Incremental" approach taken to this project has led to the segmentation of code that makes each feature work.

6.2 Full Screen Photo Gallery

One issue identified during testing was the small size of the photos within the gallery, particularly when viewed on smaller mobile devices. The best way to address this issue would be to implement a full screen gallery mode that can be activated on click so that the images fill the screen.

6.3 General Styling of the Web Portal

One of the areas that got overlooked during the limited timescale of the project was the styling of specific pages, particularly the pages restricted behind the web portal as the functionality was prioritised. The future work for this section is to clean up the formatting of the interface as well as make sure it is all optimised for mobile devices as the theatre group will want to be able to access these pages on their phones.

6.4 User Account Registration and Management

One of the minor features that was pushed aside in favour of finishing core functionality was the invite only registration form and a page within the web portal for managing the current users account details. As this is only a small feature, development of it will not take too long in the future and will have to be finished before rolling out the web portal.

6.5 Any New Requested Features

In the past, the theatre group have had custom built features, including a drawing contest with a submission form, embedded into their existing WordPress site, however, those features were often prone to failure and lacked the consistency of the site. With a fully custom solution handling their entire website, they can request new custom features to be added.

7. Conclusions

The main aims of the project have been successfully accomplished when it comes to the functionality of the solution however there are some small areas of improvement that have been identified regarding the styling of a few of the webpages. In addition to the styling, there are some areas of code that can be cleaned up and consolidated to make the back end of the site even more lightweight. In spite of the aspects mentioned in future work, the project is fully functional and could be deployed as the clients' full time public facing website within a short time frame, most of which would be spent reconfiguring their domain name to point to the new site. This project does admittedly need some improvement before its fully usable by the client for the web portal section, but those changes are minor and do not require many functionality changes.

In conclusion, although the solution is not fully polished, it does include the majority of the high priority features that were discussed in the initial plan. I believe that the project has the fundamentals of a unique website and web portal that will become increasingly more important in the client's future productions as more of their members start to use it and thus, I deem the project a success.

8. Reflection on Learning

8.1 Learning OAuth

One of the biggest transferable technical skills that I learnt during this project was how connecting an application to OAuth for authenticating a user with their login credentials for a third party service.

OAuth is becoming one of the most prevalent technologies on the web as it allows for one application to be granted access to aspects of a user's account on another service or as an alternative sign in method so a user can login with their Google or Facebook account.

The actual process of authenticating with an OAuth server is not the most intuitive and takes quite a bit of reading, testing and thinking in order to make sense of the process and to fully understand it. However, once that understanding has been found, it is comparatively easy to apply the same methods to another services OAuth implementation, as the actual steps involved are usually the same.

8.2 Working with a Real-World Client

One of the most beneficial skills I gained from working on this project was the experience of developing for a real client. Actually developing for a client involves working through what their expectations of the project are, which of those initial aims are actually technically feasible and then whittling these

down to the final list of requirements for the site.

This skill is directly applicable to working with any other client in the future and has given me valuable experience which will help in future projects for other clients.

8.3 Timescales

One of the biggest changes in skill level across the duration of this project was my ability to judge the time that implementing a specific feature or change will take. Prior to working on this project, I found estimating the required time to complete a task to be a bit of a challenge and often resulted in under-quoting the length of time. As the project progressed, the accuracy of my estimates improved but the overall project scope was still larger than the allotted module duration which meant that some aspects had to be cut back or dropped altogether.

Glossary

- **Google Photos** - A photo sharing and storage service developed by Google.
- **Markdown** - A lightweight markup language for creating formatted text using a plain-text editor.
- **OAuth** - An open standard for access delegation, used for granting websites or applications access to users' information on other websites but without giving them the passwords.
- **Pixel Density** – The number of pixels in a given area of a screen.
- **WordPress** - WordPress is a free and open-source content management system.

Table of Abbreviations

- **2FA**: Two Factor Authentication
- **AmDram**: Amateur Dramatics
- **API**: Application Programming Interface
- **CMS**: Content Management System
- **IMDb**: Internet Movie Database

Appendix

- **Demonstration URL**: <https://CM3203.0xA455.dev>
 - **Admin User Login**: test@example.com
 - **Password**: test
- **Clients Old Site**: <https://silchesterplayers.org>
- **Video of Test Cases** - <https://youtu.be/q-hf8c8pCwA>
- **Video of Speed Comparison** - <https://youtu.be/ft76G32-akc>

References

- [1] University of Waterloo, "Project and Development Approaches," [Online]. Available: <https://uwaterloo.ca/ist-project-management-office/methodologies/project-and-development-approaches>. [Accessed 17 May 2022].
- [2] L. Goode, "Google will shut down Picasa this spring," The Verge, 12 Feb 2016. [Online]. Available: <https://www.theverge.com/2016/2/12/10980882/google-picasa-photo-app-shut-down-may-2016>. [Accessed 15 May 2022].
- [3] Wikipedia, "Blackboard Learn," [Online]. Available: https://en.wikipedia.org/wiki/Blackboard_Learn. [Accessed 14 May 2022].
- [4] Software Advice, "Blackboard Learn Software - 2022 Reviews, Pricing & Demo," [Online]. Available: <https://www.softwareadvice.co.uk/software/4829/blackboard-learn>. [Accessed 14 May 2022].
- [5] IMDb, "IMDb Developer," [Online]. Available: <https://developer.imdb.com/>. [Accessed 14 May 2022].
- [6] Stackshare, "Flask - Companies Using Flask," [Online]. Available: <https://stackshare.io/flask>. [Accessed 14 May 2022].
- [7] O. Habib, "A Performance Analysis of Python WSGI Servers: Part 2," AppDynamics, 11 May 2016. [Online]. Available: <https://www.appdynamics.com/blog/engineering/a-performance-analysis-of-python-wsgi-servers-part-2/>. [Accessed 15 May 2022].
- [8] B. Chesneau, "Deploying Gunicorn - Gunicorn Documentation," [Online]. Available: <https://docs.gunicorn.org/en/stable/deploy.html>. [Accessed 15 May 2022].
- [9] W3C, "Web Content Accessibility Guidelines (WCAG) 2.0," 11 December 2008. [Online]. Available: <https://www.w3.org/TR/WCAG20/>. [Accessed 14 May 2022].
- [10] B. A. Parker and D. L. F. V. Scharff, "Influences of Contrast Sensitivity on Text Readability," 20 December 1998. [Online]. Available: http://j.pelet.free.fr/publications/accessibilite/Influences_of_Contrast_Sensitivity_on_Text_Readability_in_the_Context_of_a_Graphical_User_Interface.pdf. [Accessed 16 May 2022].
- [11] StatCounter, "Desktop vs Mobile Market Share Worldwide," 2022. [Online]. Available: <https://gs.statcounter.com/platform-market-share/desktop-mobile/worldwide/#yearly-2009-2022>. [Accessed 13 May 2022].
- [12] WebPageTest, "Speed Index | WebPageTest Documentation," 2021. [Online]. Available: <https://docs.webpagetest.org/metrics/speedindex/>. [Accessed 20 May 2022].