Campus’ Choice

For CT216 - Software Engineering I

Eoin Griffin - 22331331 Oisín Mac Aindreasa - 22309711

Conor Moran - 22510003 Stephen Murphy – 22462694

Conor Salmon - 22402456

**Introduction**

Campus’ Choice is a website where users are presented with two randomly chosen images of buildings or locations in the University of Galway. The user is to click on which of the two they prefer, whereupon another set of two is provided.

A score will be tracked for every image, with a leaderboard on a separate page showing the highest-ranking images.

A third page invites users to submit photos of their own, which (following manual approval by us) will be added to the list of images that can be voted on.

The website was created as both a fun “timewaster” but also serves to gain an insight into students at UG and their favourite locations on campus. Right now, this website features exclusively buildings on the UG campus, however our site is very versatile, and a small adjustment is all that is needed for it to be used for other places, hobbies, TV shows, music, movies or any topic containing multiple subjects that can be ranked.

The website intends to utilize a new method of data collection which although is possibly less comprehensive than traditional data collection methods (surveys, questionnaires, etc) more than makes up for it with how much more engaging and entertaining it can be.

**The Team**

Eoin Griffin - Frontend developer

As one of the frontend developers, I was responsible for building the basic functionality of the website where the backend team would be able to implement their end's functionality. I also liaised with Conor when it came to implementing the CSS into the website.

Oisín Mac Aindreasa – Backend developer

I assisted Stephen in setting up the database and writing the functions to retrieve/update data in the database. I also implemented authentication for the website and created a login and a registration page.

Conor Moran - CSS and Frontend developer

My role in this project was working mainly on the appearance of the final website by implementing CSS through the pages. Some of the key features I implemented through CSS was the sidebar where the user can easily navigate from page to page, The uniform colour scheme through all the pages, and drop shadowed key areas of the website such as login areas and the modern buttons. I also helped Conor and Eoin out with the Frontend of the project.

Stephen Murphy – Backend developer

Me and Oisin set up the Firebase storage and database and made the functions that retrieved info from the database and uploaded data to the database for use on the website. I then implemented these functions onto the websites framework which was set up by the front-end developers.

Conor Salmon – Frontend developer & Project Co-ordinator

As a developer, I helped Conor with the UI of the website and deciding on how it should look. As project co-ordinator, I instructed the team on what to do and what goals we should work towards. I took the lead when it came to work not directly associated with coding, for example writing these reports.

**Engagement**

As the idea is so simple “Pick your favourite of 2” with no option for anything else or very much effort needed at all from the user, we seek to increase the engagement of anyone we would like to survey.

An example where we believe our idea could be well suited is in YouTube ad surveys.

If you have ever watched YouTube for a length of time you may have encountered a pop-up survey/questionnaire asking you to rate specific ads and give you satisfaction with your ad experience on YouTube. However, the data collected from these surveys is often useless as most people either select the worst option available because they dislike ads or click any button on the screen rather than reading them to essentially skip the survey to get to the video they would like to watch.

If instead the user was presented with two ads they had seen recently and asked to pick one of them this would eliminate both problems as it is simple enough read and understand to partake in seriously very quickly and there is no way for every answer to give little information through lack of engagement.

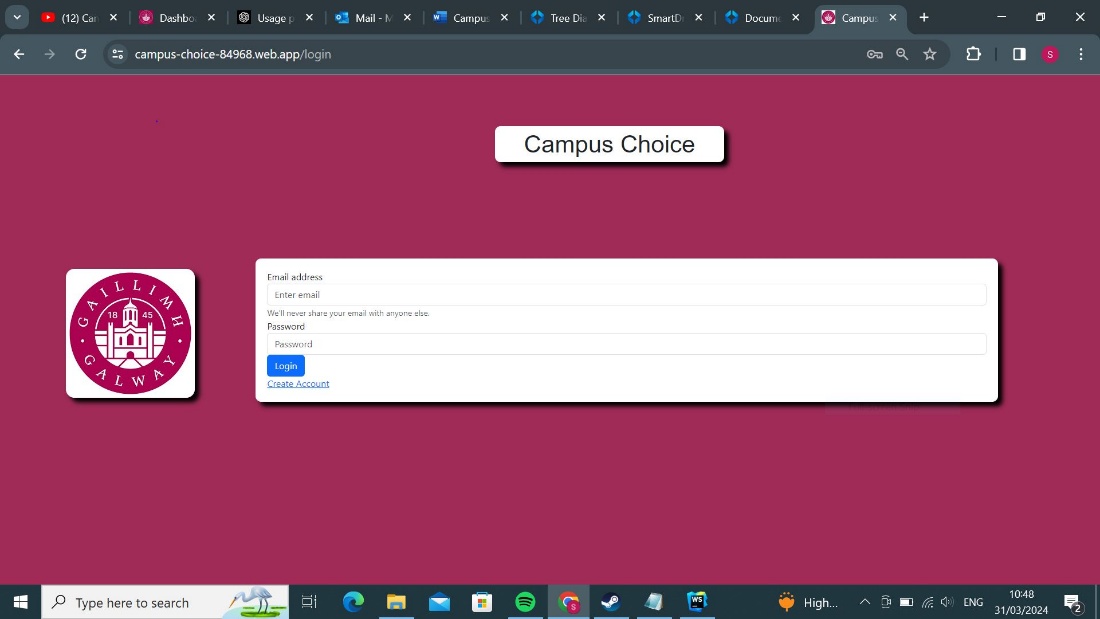
If implemented with YouTube’s large audience, the data that could be collected using comparisons of ads would be incredibly useful. This is just one example of how our idea could be applied.

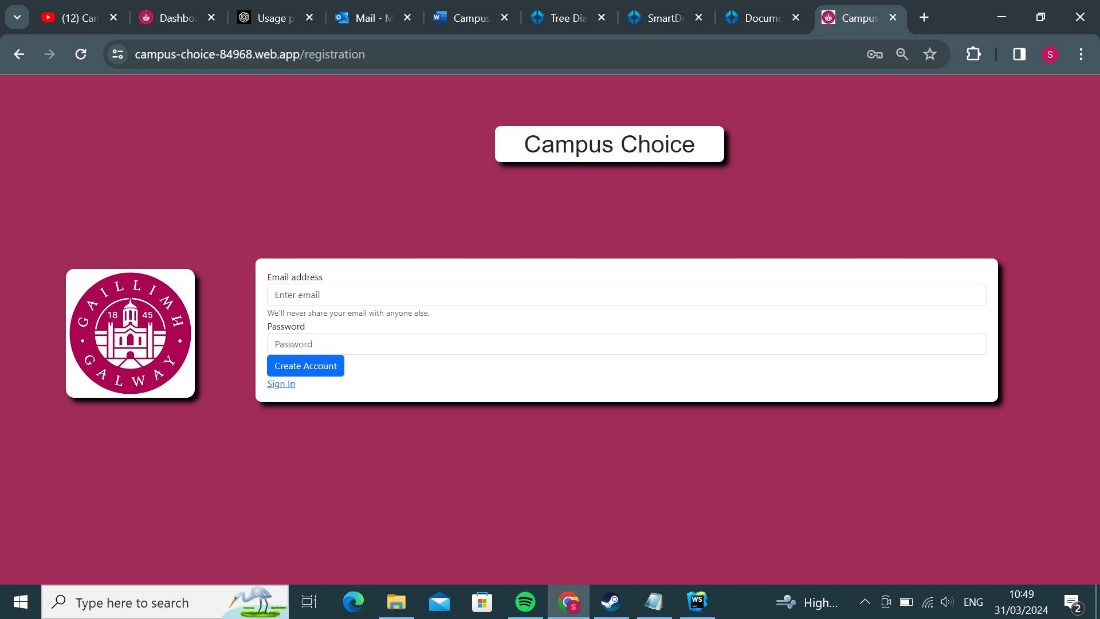
**Frontend Features**

Campus’ Choice is comprised of several web pages, included the main voting page, a leaderboard page, a login/register page and the submissions page. All pages will have links to the other sites and will work from the same CSS to maintain a uniform appearance. The Login/Register page is the first one users will see, as they will need to login with their email and a password before they can continue. As such, all other pages will redirect to the login page until the user is signed in.

Login Page/Registration Page

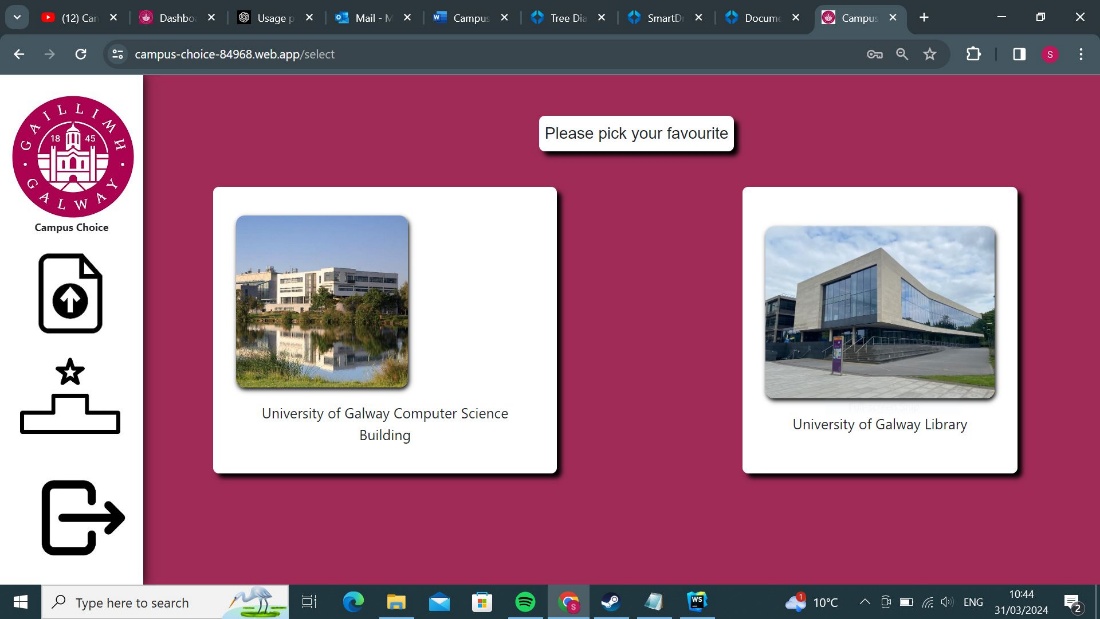
The Login Page is the first page users will be presented with when they open the website. It contains input fields for the user's email address and password, along with a button to submit the login information. Additionally, there is a link to the Registration Page for users who don't have an account yet. The Registration Page contains input fields for the user’s email address and password as well, and there is a button to create the user’s account. Also, there is a link to the Login Page for users who already have an account.





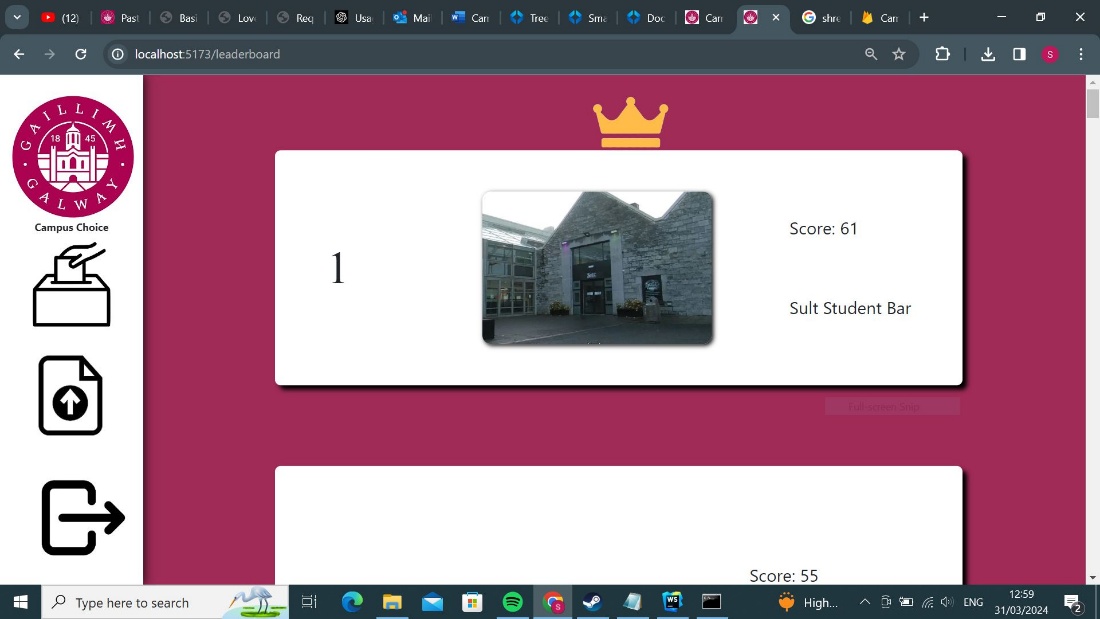
Selction Page

The main page presents the user with two images selected at random from the database. The pictures are chosen by taking two random numbers, with the maximum number determined by “totalentries”, a document in Firestore which contains the total number of images to choose from. If the generator returns the same number twice or if a number’s corresponding image is yet to be approved by us, then it will get a different number. Once two unique approved images are chosen, then these are the ones that will be displayed. From here, the user can click on one of two of these images. The image that is clicked will have its “score” increased by 1, a value which is stored on the database and will be used later in the leaderboard page.



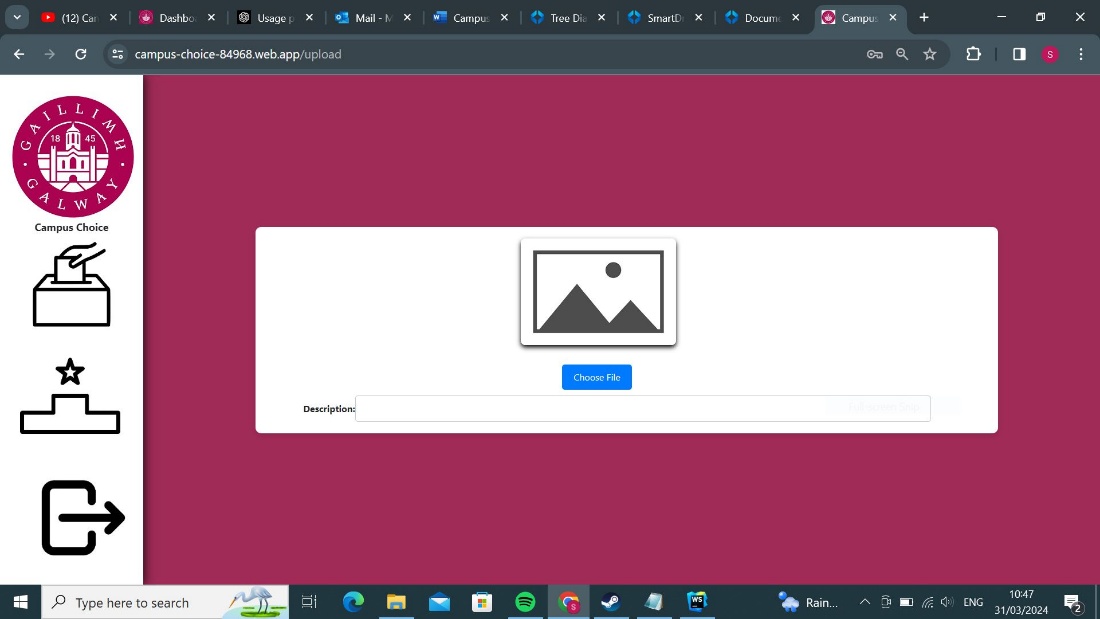
Leaderboard Page

The leaderboard page, like the main page, makes use of the images in the database. In addition to this, it also takes into account the scores gathered from the main page, which are stored in entries alongside the image URLs and their descriptions, detailed further in the Database section below. When this page is loaded, all entries that are approved to appear on the main page are sorted based on their score.



Submission Page

The submission page is where users can upload their own images to Campus’ Choice. It contains a file input and a text box for both the image itself and a description of it to be used on the leaderboard. A successful submission requires both an image and description. The image file can only be accepted if it is a PNG, JPG or JPEG, and must be less than 2MB in size. The description, like the password on the login screen, has a character requirement. However, unlike the password, which has a minimum of 6 characters, the description has a maximum of 50 characters in length and can’t be any longer. If the file is too large, the wrong type or missing, or if the description is too long or missing, then an error message specific to what the issue is is provided to the user and no submission is made. If the submission is successful, then the image and description will enter our database, but won’t appear until it has been manually approved by someone with access to the database. This was included in order to moderate submissions and ensure no inappropriate images made their way onto our site.



CSS

We worked carefully on the CSS to ensure it had a presentable, non-complicated look. To that end, we chose a colour palette of maroon and white inspired by Galway’s colours as well as those used in the current University of Galway logo, given that our website centres around the university and its locales. From the screenshot you’ll notice that we use a maroon background with white rectangles, which help the content on them stand out, as the black font colour wouldn’t show properly on maroon.

8

**Backend Features**

Backend features most commonly involve receiving information from the users through the frontend features and storing it in our database. This includes the email and password when the user logs in, which of the two photos the user clicks on in the main page and the photo and description the user submits. These are further detailed in Frontend Features above and Database and Authentication below.

**Database**

The database we made use of was Firestore, as recommended by our lectures. We utilised Firestore to store images submitted by the users of Campus’ Choice. These images are stored on the database, but in addition a corresponding entry is created, which are documents assigned both the URL of the image and the description provided by the user. As well as the user-provided variables, these entries are given an index number to make it unique from the other entries on the database and a Boolean value named “Approved”, which as its name implies denotes whether the photo has been approved to be shown on the main page. This value is set to false by default and can only be manually changed to true by one of us. Lastly, every entry has a “score”, which starts at 0 but increments by one every time it is clicked on by a user. There is a final document in the database, named “totalentries”, which contains a value that increments with every new entry to the database. This exists so that random number generator that provides the two random images on the main page knows how many images there are to choose from.

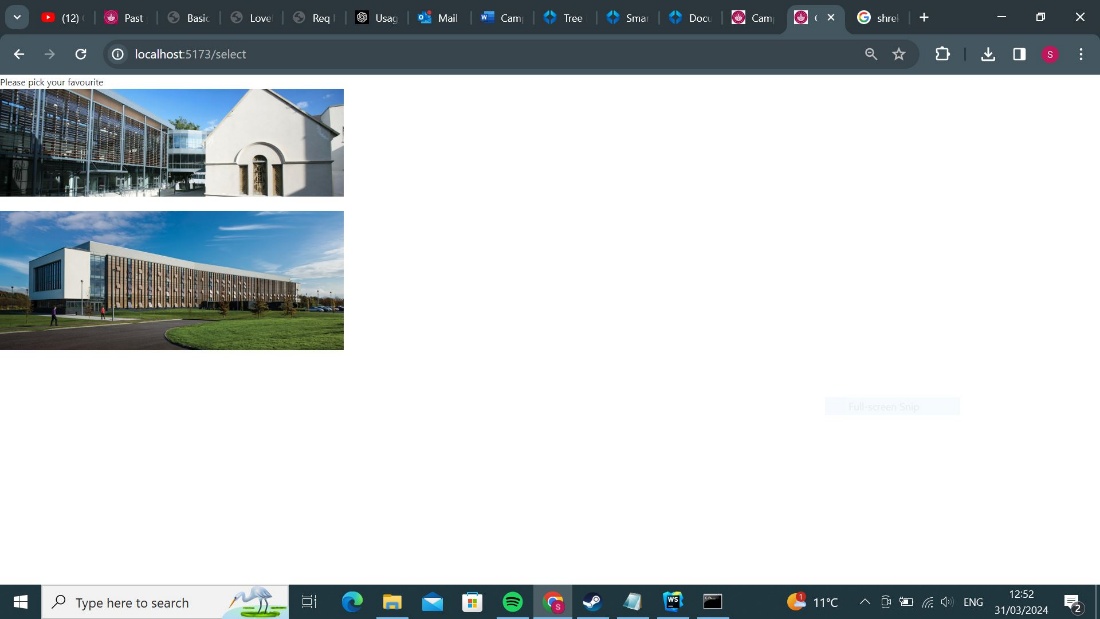
**Authentication**

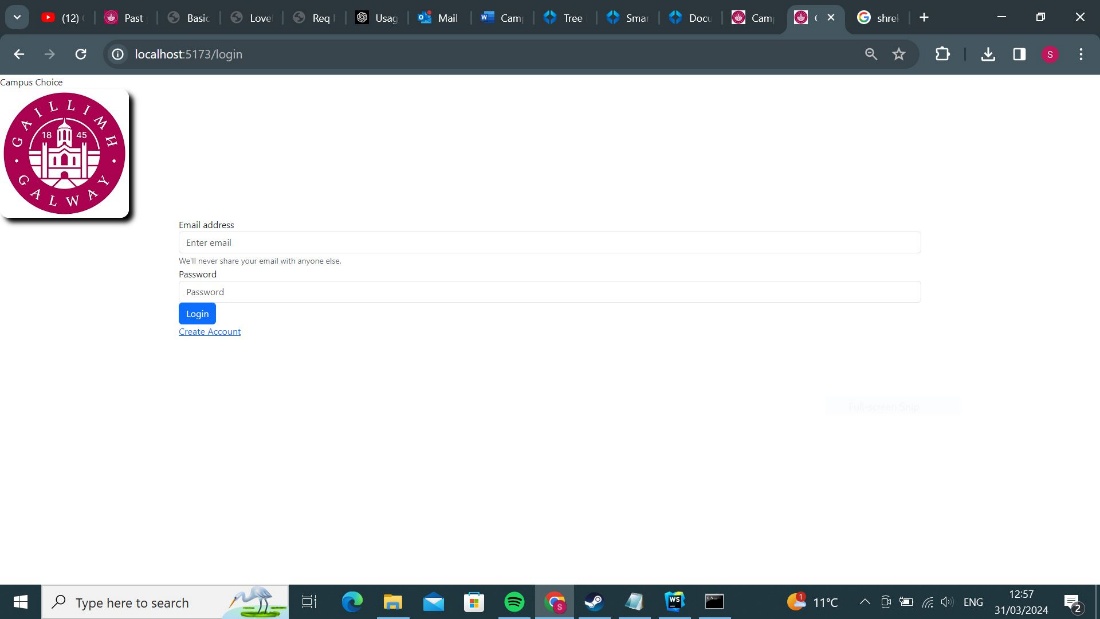
We implemented the authentication that Firebase provides. Users can sign up using their emails and passwords, and Firebase hashes the passwords and stores them securely. The Selection, Leaderboard, and Submission pages are only accessible to users who have been signed in and verified. This greatly improves the security of the web app.

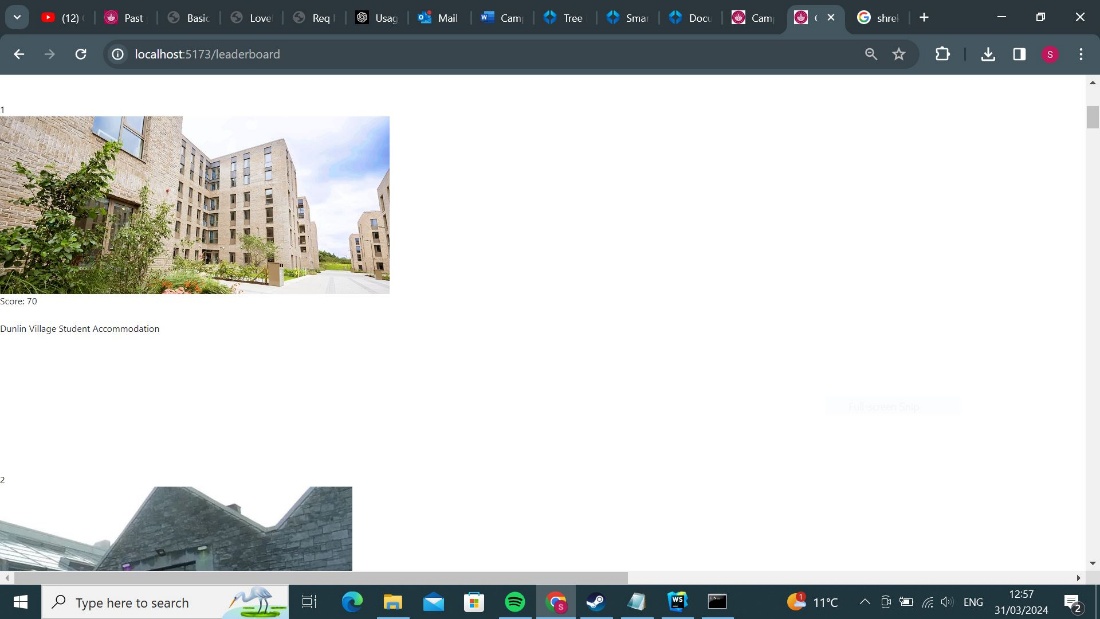
**Testing:**

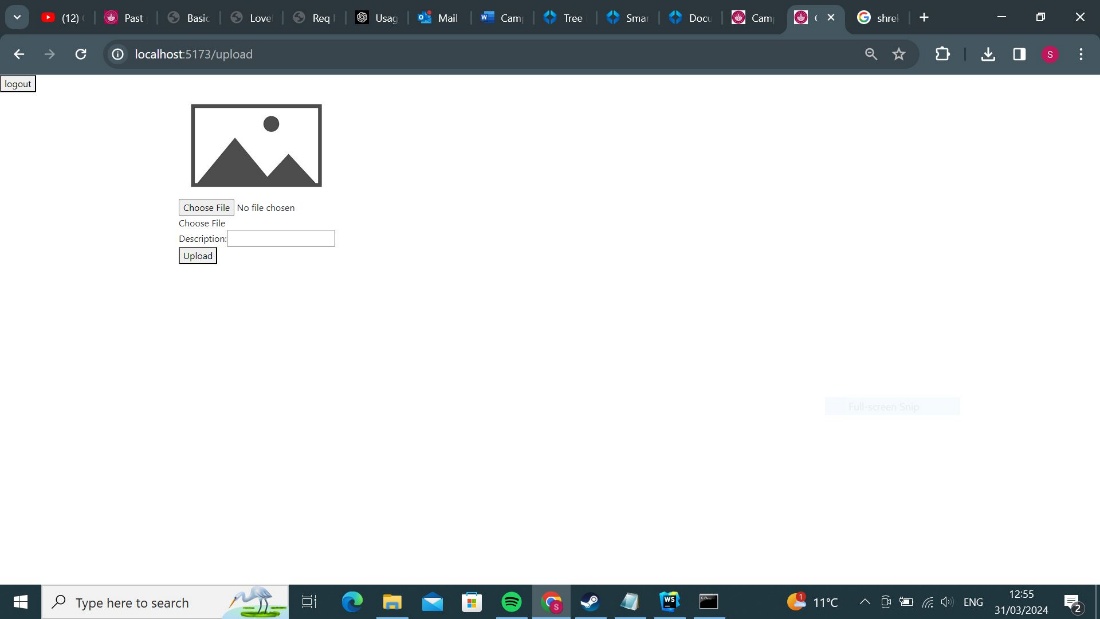
When it came to testing our project, we made a collective decision to not use Postman to test aspects of the project such as the file upload. The way we designed our website made it easier, faster and clearer to test features and functions by simply using the frontend of the project. Some of the main errors that we identified while testing was that we needed to add a file size limit and only allowed certain file formats (only images), which we weren’t able to test with Postman.

Below are some work-in-progress screenshots of Campus Choice before the application of CSS:









**Difficulties:**

Learning to use VueJS and implement Firebase functions into our product was very challenging. We had quite a lot of trouble retrieving and uploading info to and from our database and creating a system of functions that completed all the tasks we needed as simply and efficiently as possible.

We discovered that our CSS did not scale when the browser is zoomed in or out, which makes the website’s features overlap with each other on phones or when zoomed in.

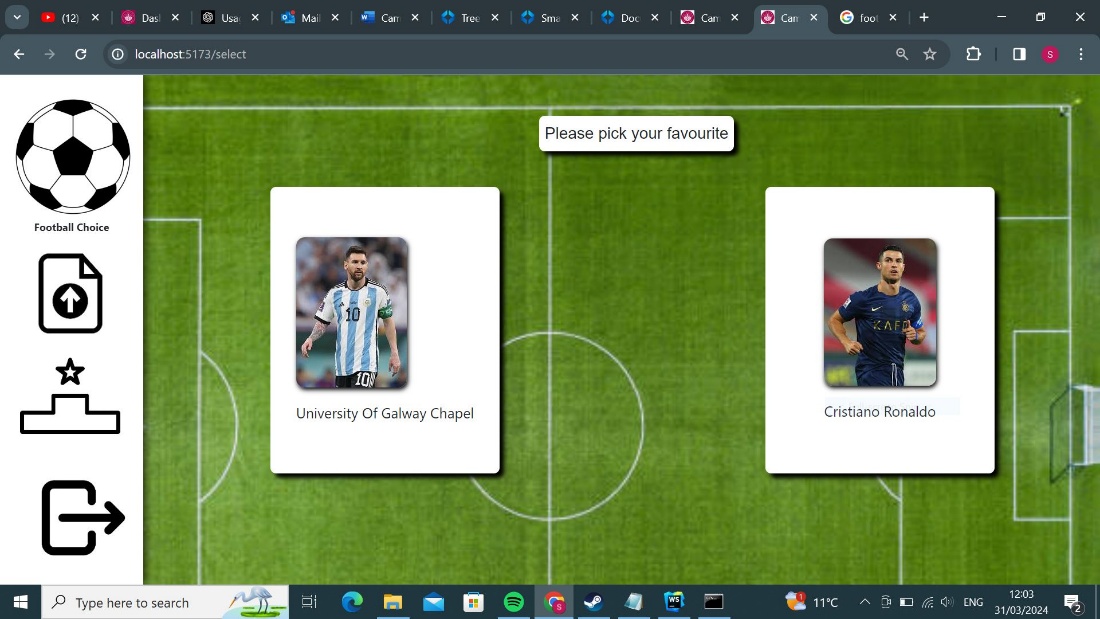
**Future Changes**

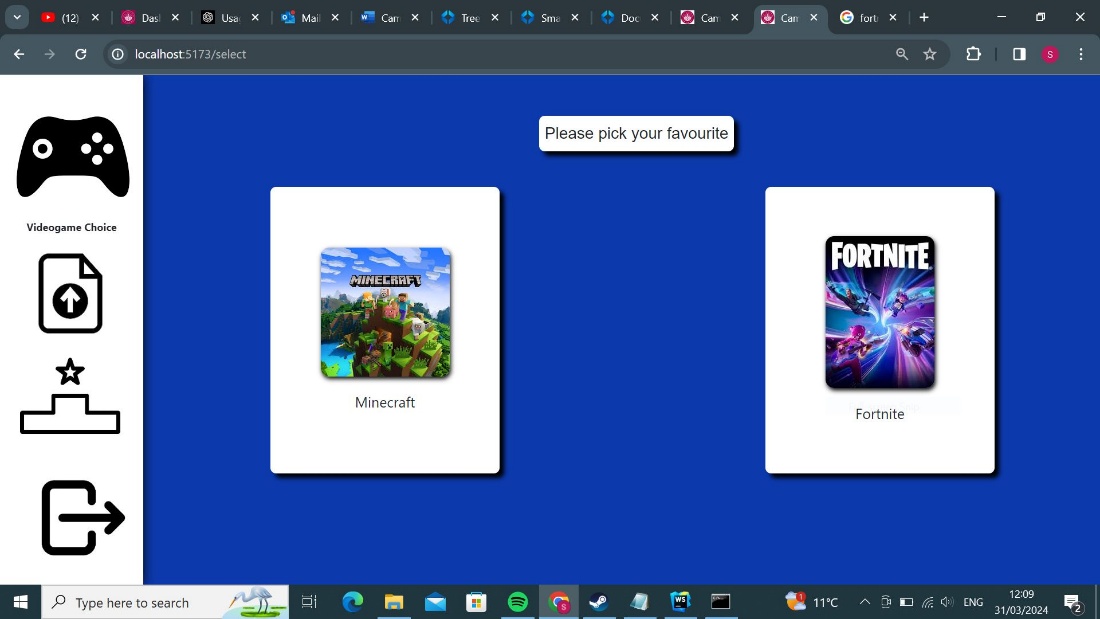
We would redesign our website to be much more mobile-friendly as our website idea is suited well to mobile devices and for added convenience. Additionally, we believe a mobile phone audience would be more likely to resonate with our low attention-span idea for the way our website collects data.

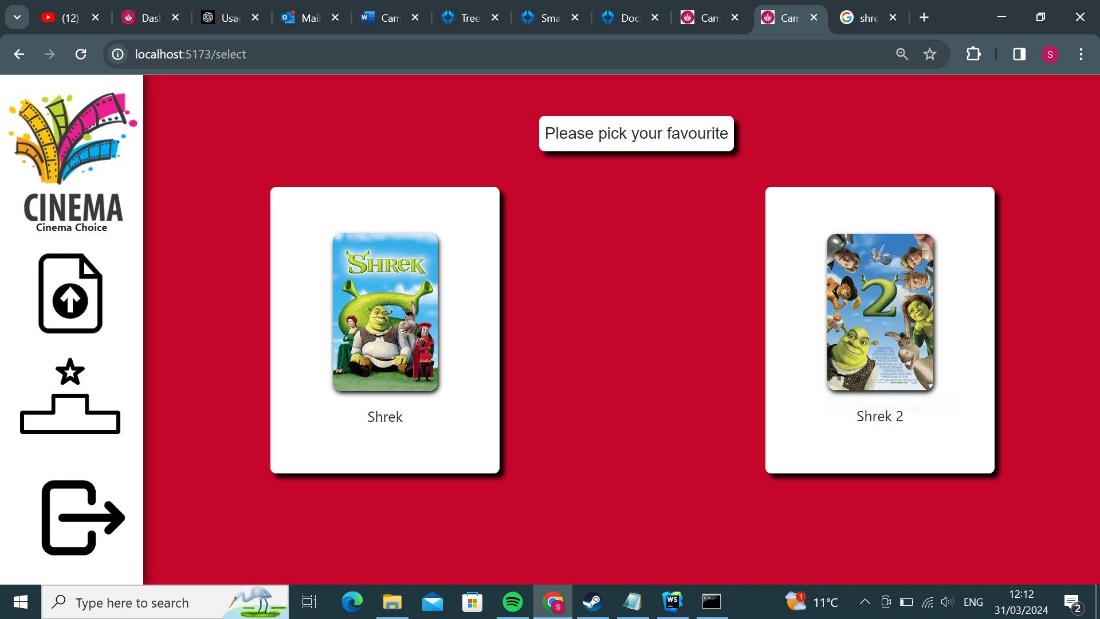
We would add pages that showed the leaderboard for each specific user instead of the leaderboard only displaying all the scores from every user as an option.

We could also tally the total amount of times each image is clicked and divide by total clicks altogether on the website to get percentages for out data and maybe find a more sophisticated way to tally the score than simply adding 1 to the image clicked.

We could add pages for more than just campus buildings and show off the versatility of our idea with other voting pages for movies videogames or footballers, for example:







We could also add a way for certain users to be admins so that they could access pages on the website that regular users can’t. This would enable us to approve image on the website itself through an admin only approval page instead of by approving images manually in the Firestore database.

This way, we could entirely separate the approved images and unapproved images in to two separate collections in the database, entirely removing the need for the checkapprove() function.

Unfortunately, the many times the website must check each entry for approval significantly slows down the website so having this feature would simplify the approval process and increase the websites overall performance.

**How it Works**

**Entries/Firestore database**

The Firestore database holds all the entries on the site. Each entry has these parameters:

Name/document name: entry1, entry2, entry3 etc.

Image URL: retrieved after upload from Firestore database

Score: updated/incremented when entry is selected on selection page initially 0.

Approved: Boolean true or false used to determine whether an entry can be displayed on the leaderboard/selection screen initially false.

Entryno: entry number for this entry.

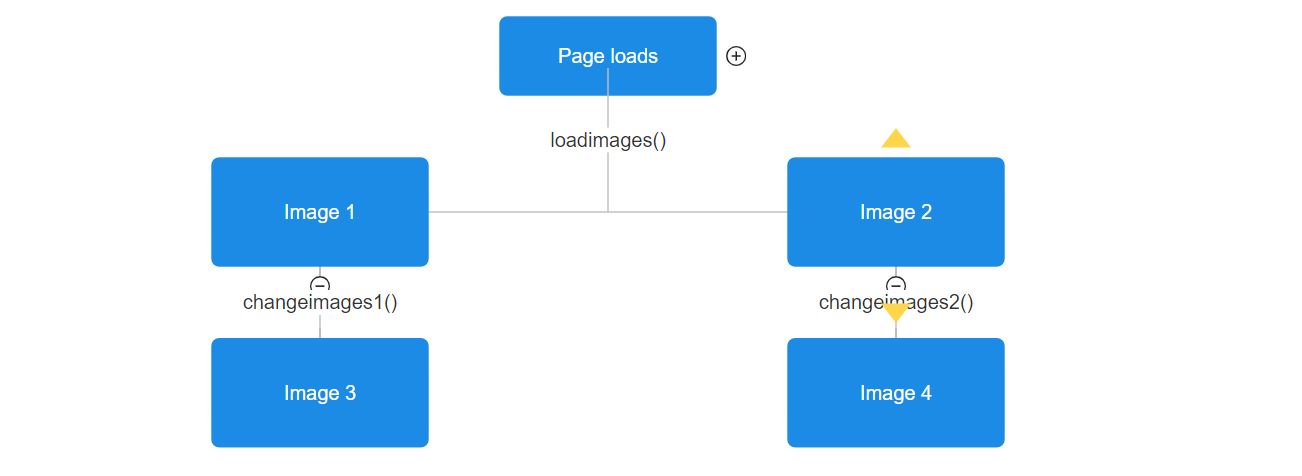
Description: string to store description of entry.

Also inside the images collection is the document totalentries. This document contains the total number of entries currently in the collection.

**Selection Page**

When the selection page loads two random and unique approved entries’ image URLs and descriptions are retrieved from the database and displayed on screen this is all completed by the function loadimages()

Once the user clicks on one of the two images displayed either changeimages1() or changeimages2() runs and two new images are displayed. **The key difference is that the changeimages functions also update the score of the image clicked on the Firestore database**. Changeimages1() increases the score of the image clicked on the left and changeimages2() increases the score of the image clicked on the right.

**load/changeimages1&2()**

If changeimages1() is called the score for the entry on the left is updated if changeimages2() is called the score for the entry on the right is updated.

When any of these functions are called two numbers are selected at random number 1 and number 2, These numbers will correspond to the entry number of the entry selected. number2 is selected using the generateUniqueRandomNumber() function using number 1 as its argument. GenerateUniqueRandomNumber() selects a number at random and returns it **so long as it is not the excluded number in this case number 1.** This function ensures both numbers are **unique** to prevent the same entry appearing twice on the screen. Then both numbers are checked for approval.

**Checkapproved()**

The checkapproved function checks if the approved: variable in the Firestore database is true or false. It returns true if true and false if false.

**Returnapproved()**

The return approved function checks using checkapproved if number 1 or number 2 is approved and cycles to the next number in if the number is not approved.

Two functions returnapproved1() &returnapproved2() exist for numbers1 and 2 respectively.

if number 1 is not approved returnapproved1 calculates a new number using:

number = (number % max) + 1;

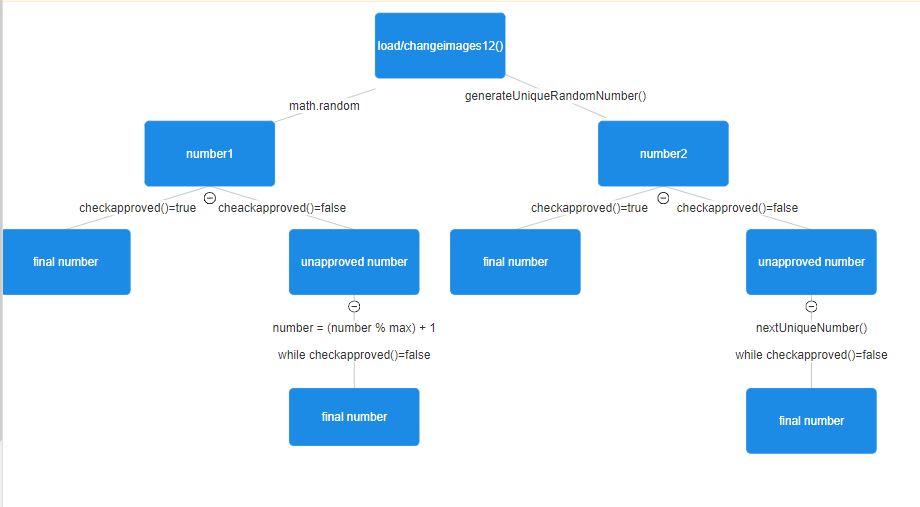
if number 2 is not approved returnapproved1 calculates a new number using:

number = this.nextUniqueNumber(number, approved1, max)

In both cases max is equal to the total number of current entries retrieved from totalentries document in the image collection on the Firestore database

Next unique number takes the final number1 returned by returnapproved1() and uses it as the number to exclude when selecting a new number 2. put simply number2 cycles to the next number if it is not approved **or if it is the same number as number1** once again to ensure that they are unique.

Finally, the new two entries are selected using the final versions of number1 and number2 returned from returnapproved() functions and the description and image URL for both are retrieved and displayed on screen.



**Submission page**

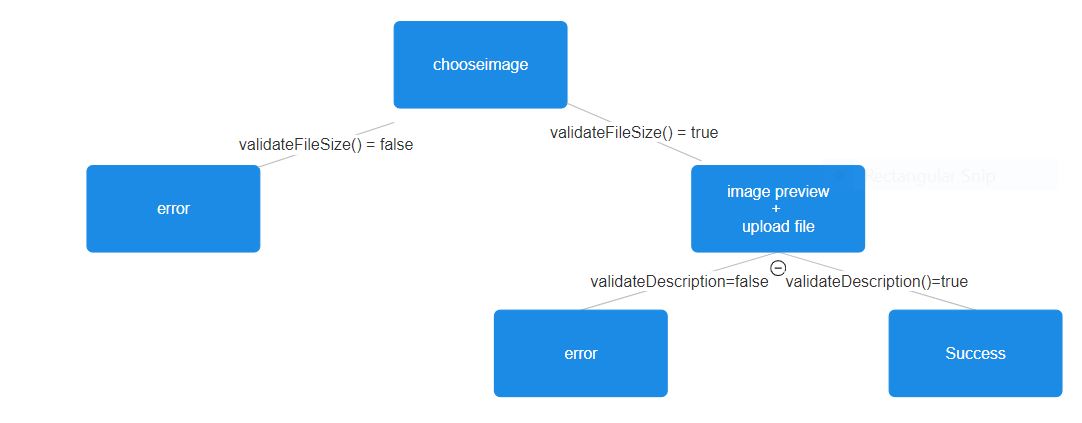
On the submission page the user is prompted to choose an image to upload and add a description to it. If the image is over 2mb in size an error appears.

Otherwise, the upload file button will appear, and an image preview of their selected image will appear on screen.

If there is no description or the description is longer than 50 characters an error appears when upload file is clicked. Otherwise, a message to show that file has been uploaded successfully appears.

Upon successful upload the image is uploaded to our Firebase storage and its URL and description, and entry number are uploaded to a new entry in our Firestore database. The new entries score is set to zero and its approved variable is set to false so that it does not appear elsewher on the website until it is manually approved by us on the database.

Also, the totalentries number is incremented by 1 in the totalentries document in the images collection in the Firestore database



**Leaderboard page**

The leaderboard page creates a leaderboard dynamically using only the approved entries in our database in order of their score highest to lowest.

It first creates an approved array of entries by cycling through all the entries in our database and using the same checkapproved() function above from the selection page. If an entry is approved its added to the array along with its score: and entryno: also taken from the database. The array could look like this

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 4 |
| Score: | 23 | 34 | 97 | 3 |
| Entryno. | 3 | 6 | 8 | 19 |

Then using the score at each index, the array is sorted using selection sort in the orderentries() function. Afterwards the same array above should look like this

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 4 |
| Score: | 3 | 23 | 34 | 97 |
| Entryno. | 19 | 3 | 6 | 8 |

The createleaderboard() function the calls the createleaderboardentry() function for each entryno in the array in reverse order so that the leaderboard displays highest to lowest.

**Createleaderboardentry()**

In the html there is the following.

<div class="leaderboard">  
 <p v-html="htmlMessage"></p>  
</div>

The htmlMessage variable is changed to the html to create the leaderboard using the createleaderboard entry function. The function

The function simply writes the following html inserting the score URL and description of the entry number passed as its argument.

<div class="entry-container">

<div class="entry-whitebox">

<div class="place-number">

Placenumber

</div>

<img src="ImageURL alt="Image" class = "entryImage">

<div>

<div class="score">

<span class="score-text">

Score: Score

</span>

</div><br>

<div class="description">

<span class="description-text">

Description

</span>

</div>

</div>

</div>

</div><br><br><br><br><br><br><br><br><br><br>

**Createleaderboard()**

The createleaderboard() function repeats the createleaderboardentryfunction() for every entry in the approved entries array passing the entryno as its argument.