Sammy Hass

Introduction to Multimedia

Report

Table of Contents

[Overview 2](#_Toc56261698)

[Design Process 3](#_Toc56261699)

[Home 4](#_Toc56261700)

[Video 5](#_Toc56261701)

[Hobbies 5](#_Toc56261702)

[Coursework 7](#_Toc56261703)

[Animation 8](#_Toc56261704)

[Bricks Animation 8](#_Toc56261705)

[Scene 1 8](#_Toc56261706)

[Scene 2 9](#_Toc56261707)

[Scene 3 10](#_Toc56261708)

[Scene 4 10](#_Toc56261709)

[Scene 5 10](#_Toc56261710)

[Scene 6 10](#_Toc56261711)

[Scene 7 10](#_Toc56261712)

[Scene 8 11](#_Toc56261713)

[Football Animation 11](#_Toc56261714)

[Scene 1 11](#_Toc56261715)

[Scene 2 12](#_Toc56261716)

[Scene 3 13](#_Toc56261717)

[Scene 4 14](#_Toc56261718)

[Scene 5 15](#_Toc56261719)

[Scene 6 15](#_Toc56261720)

[Summary 16](#_Toc56261721)

# Overview

I chose to develop my website using React because it gives me the ability to create reusable components for the web page, giving me flexibility when it comes to designing. In addition, it also speeds up my development process as it reduces code duplication. React also allows me to use features such as State and Props, both of which make creating dynamic web content more straightforward.

I used [Create React App](https://create-react-app.dev/) to create my working environment and to configure the build processes I required to get my app up and running.

I leveraged the power of React in a variety of ways, I used [React Router](https://reactrouter.com/web/guides/quick-start) so that I could create multiple pages that could be navigated around with ease. I then created a consistent navigation bar using my knowledge of CSS, JavaScript and HTML for all pages which allows the user to freely browse the site.

Figure Navigation Bar (Desktop)

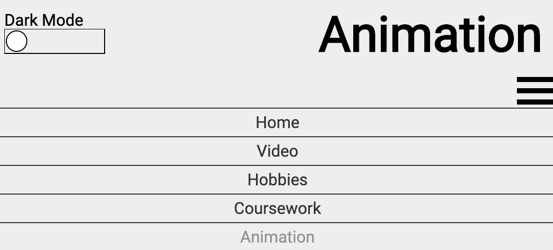


Figure Navigation Bar (Mobile)

I made efforts to create a responsive web page so that no matter the screen the user is viewing the page on, the experience will remain consistent. To do this, I leveraged the power of CSS media queries as well as React to create a [Hook](https://reactjs.org/docs/hooks-reference.html) which would allow me to access the device’s viewport dimensions and adjust content accordingly.

I created a site-wide dark mode, which, when enabled by the user will switch the styles of the site to a dark interface. The user can click on the toggle I created using Raphael.js to turn on or off dark mode. This animation uses a circle which will change its fill and its location on the toggle to indicate to the user whether dark mode is enabled. The animation has two active states, one for dark mode off and one for dark mode on.

Shape

Description automatically generatedA picture containing shape

Description automatically generated

Figure 3 Dark Mode Disabled State

Figure 4 Dark Mode Enabled State

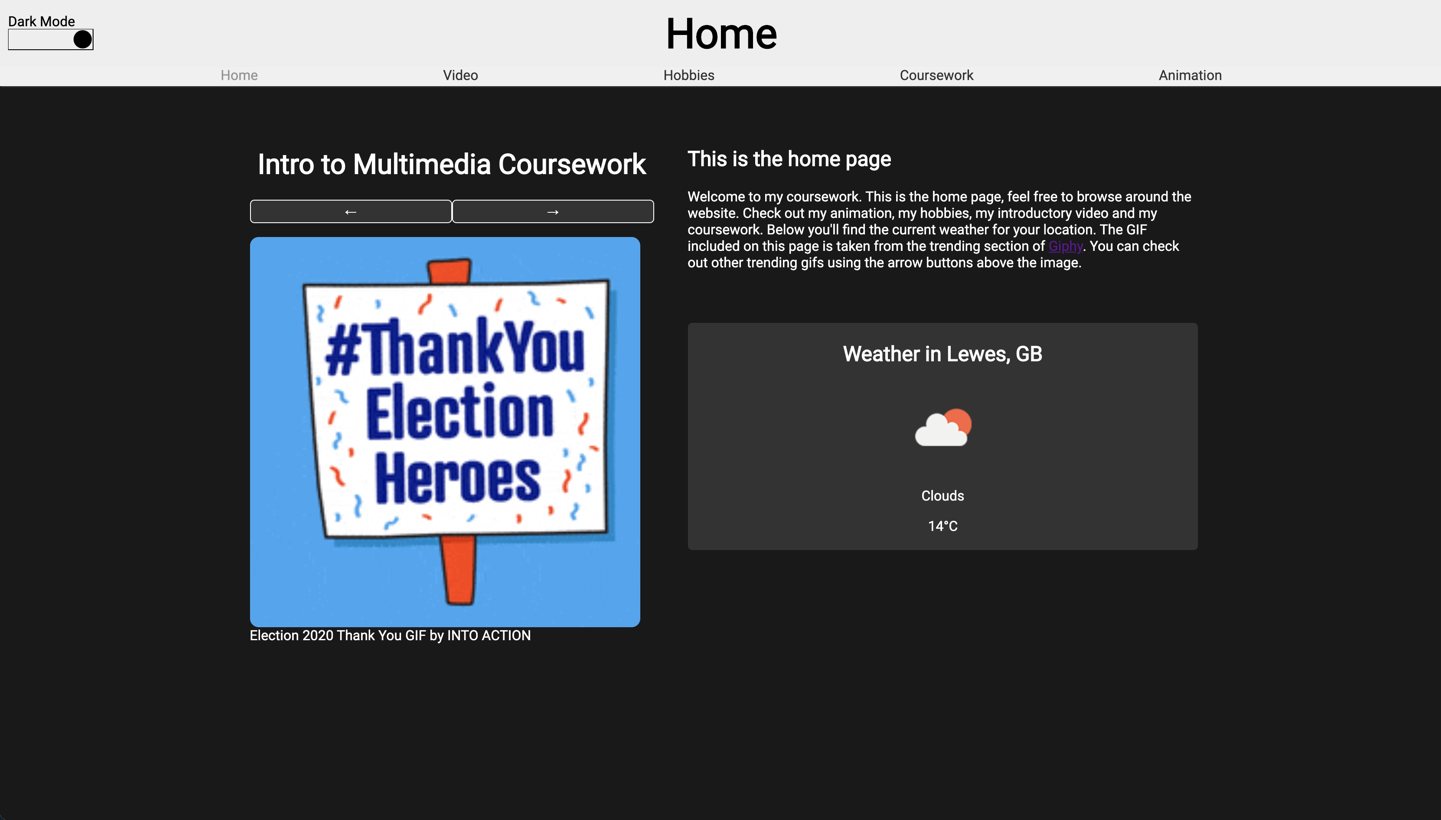


Figure Dark Mode Enabled on the Home Page

# Design Process

In developing my website, I went through multiple iterations of my design. My design process began by some creating mockups on [draw.io](https://app.diagrams.net/), I then used React and CSS to create the prototype the components I needed and positioned them to appear correctly on the screen. After this, I fully implemented all the features I initially wanted on the site.

Once these basic features were implemented, I was able to give myself more freedom with features I wanted to use in the website. As I came up with ideas, I would create components to implement them and then put them on the site where I thought was appropriate. Whilst iterating like this, I developed features such as the weather widget, the trending GIF widget. In addition I created the bricks animation after developing my first animation as I thought it would be nice to have two animation that the user could switch between.

# Home

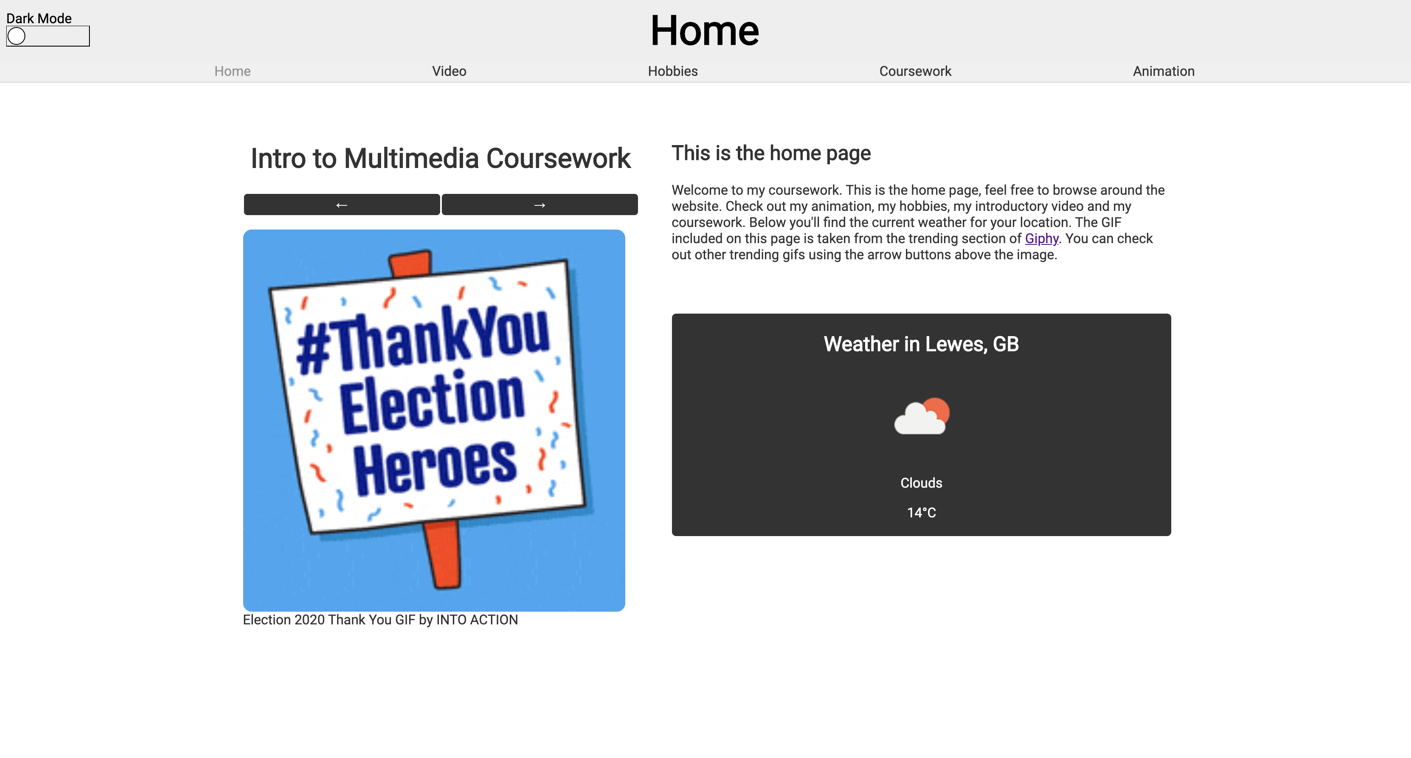


Figure Homepage (Light Mode)

On the home page, I used JavaScript to consume two different APIs. The first of which is the [GIPHY API](http://giphy.com/), which I used to retrieve the top 10 trending GIFS from the site and display them on the page, the user can then navigate between the GIFs using the arrow buttons above the image. To get the source of each GIF, the user may simply click on the image and be taken to the GIF on the GIPHY website.

The second API I used is the [OpenWeatherMap](https://openweathermap.org/) API, which I used to get the weather for the user’s current location. To achieve this, I first had to use the [JavaScript Geolocation API](https://developer.mozilla.org/en-US/docs/Web/API/Geolocation_API) to estimate the coordinates of the user, then I send I request to the OpenWeatherMap API with the user’s coordinates as input. The API also includes icons that reflect the current weather. When the API response is received, I use these icons to show the user visually what the weather is.

To style this page, I used [CSS Flexbox](https://css-tricks.com/snippets/css/a-guide-to-flexbox/) which allows me to display content in a flexible and attractive way. React allowed me to create a stateful home component, which will send requests using the [useEffect Hook](https://reactjs.org/docs/hooks-reference.html#useeffect) to make requests to APIs as well as [useState](https://reactjs.org/docs/hooks-reference.html#usestate) to keep track of data received with API requests.

The home page has a responsive design created using CSS Media Queries, this means that when viewed on a phone, the page is still easy to navigate for the user.Graphical user interface, text

Description automatically generated

Figure Home page (Mobile, navbar excluded)

# Video

The video page uses a video that I created on the app TikTok to transition between multiple different royalty free photos.

Graphical user interface, application

Description automatically generated

Figure Video Page (Still, Desktop)

# Hobbies

The hobbies page takes advantage of React reusable components to create a grid of ‘cards’ with my favorite pastimes on them.

Graphical user interface, website

Description automatically generated

Figure Hobbies Page (Desktop, Light Mode)

This page takes data from a JSON file which contains information about my hobbies and procedurally generates each card component at render time.

This page uses media queries to change the dimensions of the grid when the user is on a mobile device.

A picture containing graphical user interface

Description automatically generated

Figure Hobbies Page (Mobile, Light Mode)

# Coursework

The coursework page includes links to my bouncing animation from the lab in week 4 and a link to the work from the week 3 lab on HTML and CSS.

In addition, the page includes the image I created using photoshop in the Photoshop Lab.

I also included a button to download this report within the page, I did this using the HTML anchor tag with the ‘download’ attribute.

Graphical user interface, text

Description automatically generated

Figure Coursework Page

I also ensured the coursework page would be properly visible on smaller devices, to do this, I used media queries to change the styling of the content depending on the size of the display.

A picture containing person, person, holding, building

Description automatically generated

Figure Coursework Page, as it appears on smaller screen sizes

# Animation

On the animation page, I included two separate HTML5 SVG animations. Both of these are controlled by the audio player visible below the animation window. The user can toggle which animation is showing with the button that appears above the animation.

I created the audio player as a reusable component in React, when the play button is pressed, the animation and the music will start, when the pause button is pressed, the animation will reset. The player also includes a slider so that the user may go to a specific point in the music simply by dragging the slider’s ‘head’ or by clicking on a point in the slider. The player also shows the current time in the song in seconds.

Shape, rectangle

Description automatically generated

I created both animations by designing ‘scenes’ within the animation and then animating transitions between those scenes.

## Bricks Animation

The Bricks animation consists of twelve scenes and the transitions are handled by [Raphael](https://dmitrybaranovskiy.github.io/raphael/reference.html). Initially, the screen shows an empty 10 by 20 grid.

### Scene 1

When the animation starts. the first brick begins falling from the top of the screen, once it reaches the bottom of the screen, it will bounce for a short period and then turn white, meaning it has found its fixed position

A picture containing table

Description automatically generated

Figure End of Scene 1, Brick has fallen and turned white.

### Scene 2

The next brick will begin falling once the first has found its fixed position. This brick is a 2x2 square on the grid. It will have a green fill until it hits the bottom of the screen, at which point, it too, turns white.

A picture containing building, player, game, ball

Description automatically generated

Figure End of Scene 2, Square brick has finished falling

### Scene 3

Chart

Description automatically generatedThe third brick will fall once the square has finished its animation, this brick is special in that it is not just a simple rectangle, but rather a compound ‘L’ shape made by creating an array of two rectangles and then iterating over it to move the whole compound shape downwards.

Figure Third Brick falls

### Scene 4

The next brick will fall beside the last, in the same fashion as before, once it hits the ground it turns white.

### Scene 5

Finally, another square brick will fall, completing the bottom line.

### Scene 6

Now that all the bricks have fallen and the first line is completed, it will be highlighted to reflect that it is complete.

### Scene 7

The highlighted line will now be removed along with the bricks that it was previously covering. Each of the bricks that are now floating will fall to the ground.

### Scene 8

At this point, the animation now switches to looping over the letters in the word ‘Brick’ and displaying them on screen. These letters are drawn using the squares on the grid. Once the ‘k’ of brick has been shown, the animation will loop back to showing the ‘B’, it will then show the whole word again.

## Football Animation

To create the football animation, I took the route of creating GIFs in photoshop so that I could switch between the running player and the shooting player. I found a royalty free GIF of a footballer running and created a copy which I edited in photoshop to look like the player is shooting the football.

This animation takes advantage of dark mode on the website to change between a day and night view on the animation.

### Scene 1

A picture containing game, sport, table, ball

Description automatically generated

Figure Scene 1, Football Animation

The animation starts by displaying an empty football field with a goal visible.

### Scene 2

The player will then run on to the field, dribbling the ball. This is simply a GIF that uses Raphael to move itself to the right.

A picture containing game, sport, ball, green

Description automatically generated

Figure Player runs towards goal

### Scene 3

Once the player has run onto the field, the GIF showing the running player will change to instead show the GIF I created with the player shooting the ball.

A picture containing game, ball, green, player

Description automatically generated

Figure Scene 3, GIF has changed to show player kicking ball

### Scene 4

When the ball is kicked by the player, the GIF stops looping and is fixed at a point where the ball is invisible. The ball will reappear as its own GIF (not attached to the player), when the ball reappears, it will be animated along a path to goal.

A picture containing chart

Description automatically generated

Figure Ball moving towards goal

### Scene 5

Once the ball goes in the goal, the player will become invisible. The ball will now bounce out of the goal and stop at the center of the screen. The sun/moon (depending on whether dark mode is enabled) will now expand to take up the size of the screen. Now, the text appears on the screen telling the user a goal has been scored and that they can click to restart the animation if they wish.

### Scene 6

Graphical user interface, application

Description automatically generatedNow, if the animation detects a click event, the whole animation will restart, going back to Scene 1.

# Conclusion

I am happy with the site I developed. React was the correct choice when it came to develop this site as it made implementing new features extremely straight forward, as well as making rapidly prototyping my site far easier. React also allowed me to implement features such as dark mode with relative ease, dark mode simply uses a custom hook so that it can be accessed from any functional component.

I have taken steps to create a site that will perform well on both desktop and mobile browsers. For example, the navigation bar will transform into a dropdown of the viewport width is less than 750px and all but one page (animation page) uses CSS Media Queries to adapt the page styling based on the size of the device it is using.

In total I created three animations using Raphael for this website, the dark mode toggle, the footballer animation and the bricks animation. The animation page is the only page where it wasn’t possible to adapt to different screen sizes. This is due to the fact that Raphael requires that the animation be drawn on a page using a fixed pixel width and that the animation container itself not be redrawn after initial render.