Edinburgh Napier UNIVERSITY

Advanced Web Technologies

Gordon Swan
40202556@live.napier.ac.uk
Edinburgh Napier University - Advanced Web Technologies (SET09103)

Abstract

Advancements in web technology have made it easier and efficient to present high quality media such as images, audio and video. One such obscure form of media to arise in recent years has been the cinemagraph, an otherwise still image featuring one or two moving elements, such as flowing water or rolling clouds. Cinemagraphs rely on short MP4 or WEBM videos, embedded into websites and relying on the new HTML5 video player to dynamically and seamlessly present the cinemagraph in modern browsers. Cinemagraphs are non intrusive as they do not feature audio, and can be loaded fairly quickly as they don't normally feature more than a second or two of video.

Keywords – web, tech, technologies, internet, flask, python jinja2, app, html

REC - Cinemagraph Showcase

1 Introduction

REC is a web-based application that showcases high quality cinemagraph videos. REC makes use of Python Flask and Twitter Bootstrap to deliver powerful and responsive pages.

2 Design

REC has been designed to be sleek, modern and responsive.

User Interface Design The user interface has been constructed using the Twitter Bootstrap Framework[1]. Bootstrap is a web-framework that makes it simple to create responsive and modern websites. REC makes use of Bootstrap to provide a consistent and modern user interface across the entire application. In addition to Bootstrap, REC makes use of the latest HTML5 and CSS3 elements such as Video tags for playing the cinemagraph content, and SVG animations to create a responsive site logo.
One particular hurdle which was encountered during development of the user interface was overall performance of the application on low power hardware, such as a netbook - which was used predominantly to develop the site itself - as this particular computer struggles to load and play all cinemagraph videos at once.
This hurdle was overcome by implementing a small JavaScript element, which toggles the play/pause status of the cinemagraph video based on whether or not the cursor is hovering over it or not. This means that a user can watch the cinemagraphs as normally, whilst maintaining application performance, even on low power hardware. Below is the javascript which which only plays the content when the mouse hovers over the cinemagraph elements. Adapted from a CodePen by 'CaptainAnonymous' [2]

Listing 1: HoverPlay.js

```
var figure = $(".video").hover( hoverVideo, hideVideo );

function hoverVideo(e) {
    $('video', this).get(0).play();
}

function hideVideo(e) {
    $('video', this).get(0).pause();
}

```

Software Implementation REC is implemented using various languages and frameworks. The server-side of REC is implemented in Python, using the Flask Web Microframework, which supports complex and robust web applications.
One particular issue faced during the design of the server-side application was storage and retrieval of data, in a manner that would normally be expected from a database service. This was resolved by implementing a custom library known as 'data access.'
Data Access is a collection of functions, which provide the functionality of a simple database system, such as data searching, storage and retrieval. This collection of functions has made development of data-heavy pages much easier.
Included is an excerpt from 'data access' which searches the stored data for entered search keys.

Listing 2: searchData(string) in Python

```
# Method to search all of the stored data, returning records ↩
    where the search terms are found in the tags
def searchData(sTerm):
    sKeys = sTerm.lower().split(" ")

    allData = getData()
    resData = {}

    # Detect wildcard - If a single star is submitted as a ↩
    search key, return everything
    if "*" in sKeys:
        return allData

    for i in list(allData.keys()):
        tKeys = allData[i]["img_tags"].lower().split(" ")
        matchKeys = []

        for sKey in sKeys:
            if sKey in tKeys:
                if sKey not in matchKeys:
                    matchKeys.append(sKey)
```

```
20          if len(matchKeys) > 0:
21              rec = allData[i]
22              rec["res_rel"] = len(matchKeys) # Apply a ↩
        relavence value to this item in relation to the search
23              resData[i] = rec
24
25      return resData
26
```

3 Enhancements

User Uploads A particular feature that would greatly benefit the functionality of REC would be an upload feature, allowing users to upload new content to the site, adding metadata such as a description and tags.
One inherent obstacle posed by this feature would be ensuring that the content uploaded to the site was valid and not malicious, ensuring that code injection could not take place.

Up-vote or Like functionality In addition to allowing users to upload new content to the site, another key feature would be the ability for users to like or up-vote content which they like or would like to find easier at a later date.
The challenge with implementing such a feature would be preventing the system from potentially being exploited and abused by bots or malicious users.

Database Service As mentioned previously, REC does not make use of any database services, instead relying on a custom library for data operations. This application could benefit greatly from an external database service such as SQLite or CouchDB for data and credential storage, not only would this improve performance but also data security and integrity as the current method only makes use of JSON files.

4 Critical Evaluation

Search Functionality In order to find relevant content with the search bar, REC makes use of content tagging. This allows the search function to quickly locate relevant information, assuming that the tags are correct. This particular feature works well with the site as it allows users to quickly find content relevant to them. There are currently two search algorithms which are implemented in REC, the first is the simple tag searching algorithm which looks for exact matches between tags and search terms.
The second algorithm is a more advanced search which can look for a search term that may be located anywhere in the tag section, for example if a cinemagraph had the tag 'caravan', and a user searched for 'car' and 'van', the first search algorithm would not find any matches, however the second search would find two matches as both terms are found in the 'caravan' tag.

Hover Play As mentioned previously, one major hurdle faced during development of the user interface was performance on low power systems. For this reason, Hover Play is an important feature as it not only improves performance but also adds a professional feel to the overall presentation of the content in general as it adds to the interactive experience. Hover Play has a particular flaw, however, as it does not work on mobile devices, and for this reason, an alternative method of controlling and presenting media will be required if the site is to be truly mobile-friendly.

Gallery Viewer The gallery viewer is the primary component of the user interface, as it presents all of the cinemagraph content in a grid view. The gallery viewer works well with similar sized content, for example, 16:9 ratio landscape videos or 1:1 ratio square videos. However, when the content size varies, the gallery viewer begins to show it's flaws. The primary flaw with the gallery viewer is the column floating, as such that each item in a row will be rendered at the same height, this leads to large gaps being left between rows, if the above row has an unusually sized element as can be seen in Figure 1.
There exists a simple implementation of a gallery page on the W3 website [3], which resolves this particular issue, however the solution seems to only be compatible with image content and not HTML5 Videos.

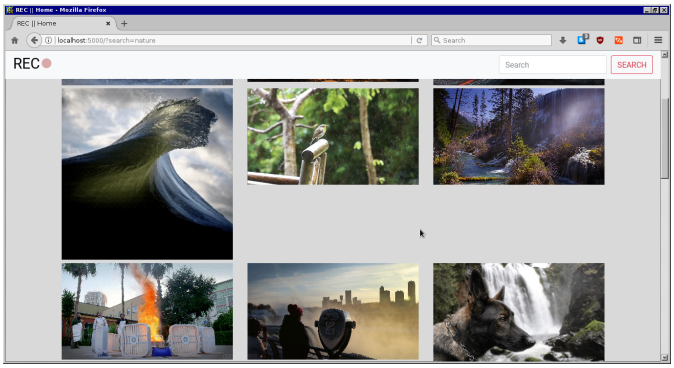

Figure 1: - **Column Gaps** - Large column gaps caused by oversized element in the above row

5 Personal Evaluation

User Experience Design Much more thought and planning than expected has gone into constructing a simple and easy-to-understand user interface. Many factors such as simplicity and page presentation have been specifically altered to allow an unfamiliar/new user to quickly navigate through the site without getting lost or stuck and having to ask how how to get to a specific page - with the exception of the account and sign-in page which are for behind the scenes access only.

JavaScript Interactivity As previously mentioned, there have been several hurdles which have stood in the way of this project in numerous different ways including performance and cosmetic challenges. A particular performance hurdle was overcome with the use of the Hover Play JavaScript, which drastically improved performance on low end hardware. Implementing this solution has been a particular challenge as experience with JavaScript has been limited until now.

Python and Flask Micro-framework The Flask Microframework has been a challenge to understand, however

learning how the framework can be used produce web-based applications has really shown it's potential. The Flask framework is an excellent solution for building fast and light web applications without much headache as the simplicity of Python allows for speedy and functional implementation.
A particular challenge has been data storage and retrieval, as mentioned previously, REC makes use of a custom collection of methods for all data operations. This collection of methods, called 'data access' reads and stores all the variable-based data in JSON files, which are subsequently read and rewritten as data is required or changed. As a result, an external database service has not been required, keeping the application in general very simple in nature.

6 Conclusion

In conclusion REC effectively achieves it's goal of neatly presenting Cinemagraphs in a sleek and modern form. REC is fast and efficient, making use of the latest web technologies available in modern browsers. There is most certainly room for improvement in REC as there are numerous changes and corrections that could be made in time, however in it's current state, the web application provides the ground work for a robust and user friendly website.

References

[1] J. T. Mark Otto, "Bootstrap - the most popular html, css, and js library in the world.."

[2] C. Anonymous, "Hoverplay.js."

[3] www.w3schools.com, "Responsive image grid."