

# COMSM0104: Web Technologies 2019

## Final Assignment Report

Tao Xu  
si19010@bristol.ac.uk

Yinan Yang  
ff19085@bristol.ac.uk

June 7, 2020

### Abstract

Our two-person team consists of Tao Xu (si19010) and Yinan Yang (ff19085). Due to the impact of Covid19, we collaborated remotely via GitHub to co-develop this project.

Our website is an online CV maker, featuring a simple and convenient interface for editing, online storage (at our server) and extensibility (the CV templates are easy to make).

The frontend of our website was based on the Vue framework, taking advantage of Vue's MVVM, the Model-View-View Model, which helped us in keeping code modular and implementing reactive user interfaces. The backend of our website was powered by Node.js with Express.js and SQLite.

**Keywords:** *Vue; SQLite.*

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Project setup	2
1.2	Compiles and hot-reloads for development	2
1.3	Compiles and minifies for production	2
1.4	Lints and fixes files	2
1.5	Start the server	2
1.6	Start the server	2
<b>2</b>	<b>Self Evaluation</b>	<b>2</b>
2.1	Estimation of marks	2
2.2	Client Side	3
2.2.1	HTML	3
2.2.2	CSS	4
2.2.3	JS	6
2.2.4	PNG	6
2.2.5	SVG	7
2.3	Server Side	10
2.3.1	Server	10
2.3.2	Database	10
2.3.3	Dynamic pages	10
<b>3</b>	<b>Working practices of the group</b>	<b>10</b>

# 1 Introduction

We have created a website that generates resumes called Simple Resume Maker. The website provides basic user registration and login functionality. Once logged in, the user can edit the resume template provided on the website on the web page and download a .pdf version of the resume.

We try to simulate the user experience of editing documents online so that what the user sees is what they get. What would otherwise be a cumbersome formatting process is made easier with different CSS. This addresses the initial point we made in designing the product, which was to make things easier.

In building this site, we used the VUE framework, which is a progressive framework for building user interfaces.

Unlike other monolithic frameworks, Vue is designed from the ground up to be incrementally adoptable. The core library is focused on the view layer only, and is easy to pick up and integrate with other libraries or existing projects. On the other hand, Vue is also perfectly capable of powering sophisticated Single-Page Applications when used in combination with modern tooling and supporting libraries.

— Official development documentation from Vue

## 1.1 Project setup

npm install

## 1.2 Compiles and hot-reloads for development

npm run serve

## 1.3 Compiles and minifies for production

npm run build

## 1.4 Lints and fixes files

npm run lint

## 1.5 Start the server

npm start

## 1.6 Start the server

See [Configuration Reference](#).

# 2 Self Evaluation

## 2.1 Estimation of marks

- A for HTML
- A for CSS

- A for JS
- A for PNG
- A for SVG
- A for Server
- A for Database
- A for Dynamic pages

## 2.2 Client Side

### 2.2.1 HTML

In terms of front-end architecture, we introduced the Vue architecture, using Vue-CLI as a pre-development generation tool. We chose Vue because we tend to develop a less web-heavy application. If we had used React, it would have been the right choice at some level, but the whole architecture would have been weighty. And that is a departure from the original intent.

For front-end routing, we use vue-route to perform page hops. We compromised on this point because vue is better at performing single-page operations. If you preview our project, you will see that while it looks like we are doing multiple pages, the user is in fact only on one page. We are using routing to control the display of components, which plays to the strengths of vue.

```
17 const routes = [{
18   path: '/',
19   component: App,
20
21   children: [
22     {
23       name: 'App',
24       path: '',
25       components: {default:Home, top:Menu},
26     },
27     {
28       path: '/selectTemplate',
29       components: {default:selectTemplate, top:Menu},
30     },
31     {
32       path: '/about',
33       components: {default:about, top:Menu},
34     },
35     {
36       path: '/login',
37       components: {default:login, top:Menu},
38     },
39   ]
40 }]
```

Figure 1: routes

We completed a total of more than 30 Vue components. Thanks to Vue's component-based development, our front-end applications are not the mess they were a decade ago. Moreover, this design is effortless to maintain later.

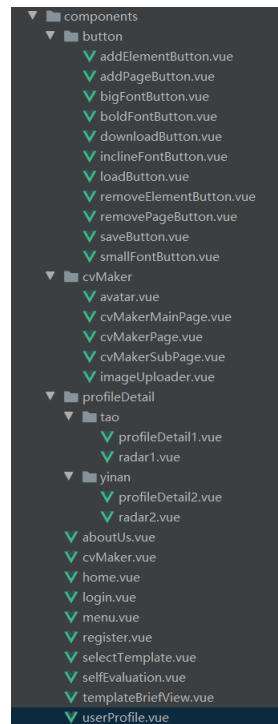


Figure 2: components

We make full use of the strengths of Vue component communication in the mutual communication of components. Below is an example of a parent component calling a child component method. In the CVMaker page, when we press the bold button in the tools bar and complete the click, the button in an active state.

```

449     boldifyText(){
450       this.recoverAllButtons();
451       if(this.mode !== MODE_BOLDIFY){
452         this.mode = MODE_BOLDIFY;
453         this.$refs.bold.active();
454       }else{
455         this.mode = MODE_EDIT;
456       }
457     },

```

Figure 3: component communication

## 2.2.2 CSS

We use a separate CSS file that is loaded inside the Vue component. Although the Vue template allows us to define styles internally using the `<style>` tag, we still load all the CSS placed separately under the folder `src/view/index/assets` for easy management. For CSS, we have three instructions.

- Use of basic CSS and changing style dynamically by changing style parameters. Each CSS file contains multiple class ids that are used on specific pages. We also set up a global background CSS, one with a furry glass effect to give the whole screen more colour.

The following is an example of how to adjust the progress bar according to the download progress in the downloadButton. We dynamically adjust the width of the bar to match the expected download. Of course, to avoid single-threaded js blocking, we set a timer to control how often the progress bar refreshes.

```
<div class="progress-bar" :style="{width: progressBarWidth+'%'}" ref="progressBar"></div>
```

Figure 4: change style

- SVG-based animation

We completed the svg-based animation in CSS, which is also explained in detail in the SVG section.

```
110 @keyframes dashLoop {
111   from {
112     stroke-dashoffset: 7;
113   }
114   to {
115     stroke-dashoffset: 0;
116   }
117 }
118
119 @keyframes blink {
120   from {
121     opacity: 1;
122   }
123   50% {
124     opacity: 0.5;
125   }
126   to {
127     opacity: 1;
128   }
129 }
```

Figure 5: css animation

- Store CSS in the back-end database for dynamic retrieval

The main feature of our website, making different templates for resumes is based on the replacement of different CSS. This is the main logic behind the primary function of our website. In CVMaker, we dynamically retrieve the CSS stored in the database according to the user's choice, to generate the user's Selected resume template. This feature set makes it easier to run this system, maintain it later, and increase the number of resume templates without spending vast resources.

```
fetchTemplate(){
  const templateElemId = 'cv-template'
  if(this.templateId === undefined) return;
  // removing existing template
  let existingTemplates = document.querySelectorAll( selectors: `#${templateElemId}` );
  for(let templateNode of existingTemplates){
    document.head.removeChild(templateNode);
  }
  // add template
  const styleElemHTML = `<link id="${templateElemId}" rel="stylesheet" href="${this.templatePath}">`
  document.head.insertAdjacentHTML( where: 'beforeend', styleElemHTML );
  // perhaps find a better way
  // this.$forceUpdate();
  // console.log('template applied.');
```

Figure 6: resume template

### 2.2.3 JS

Because of the vue template, it becomes straightforward to embed js in the page. Each page component exists separately as an element, and we write js methods internally or externally that can change the arithmetic. Of course, this is not technically a javascript file anymore, but primarily at the developmental level, they are a kind of Stuff.

We wrote a lot of front-end and back-end js logic to ensure that the complete project documentation worked adequately. Similar to the button or switch screen functions are used to implement the vue methods. The example below is the js method after the download button is pressed.

```
67 methods:{
68   downloadButtonClick() {
69     bus.$emit( event: 'downloadAsPdfClick', args: null);
70     this.start=null;
71     this.progress=null;
72     this.aniation();
73     this.$timer.start('grow');
74   },
75   aniation(){
76     if(this.$refs.button.classList.contains("downloaded")){
77       this.$refs.button.classList.remove( tokens: "downloaded");
78     }
79     this.$refs.button.classList.add("downloading");
80     setTimeout( handler: ()=>{
81       this.$refs.button.classList.replace( oldToken: "downloading", newToken: "downloaded");
82     },this.inputTime);
83   },
84   grow(){
85     if((this.progress < this.inputTime)||(!this.progress)){
86       var timestamp=new Date().getTime();
87       if(!this.start) {
88         this.start=timestamp;
89         this.width=0;
90       }
91       this.progress=(timestamp-this.start);
92       this.width= (this.progress / this.inputTime) *100;
93     }else {
94       this.$timer.stop('grow');
95     }
96   }
97 }
```

Figure 7: resume template

### 2.2.4 PNG

We used GMIP for mapping, including the default resume header and png diagrams for 404 pages. The source file of the 404 page is saved in *src/view/index/img*. We used masks, filters and transparent alaph channels, among other techniques.

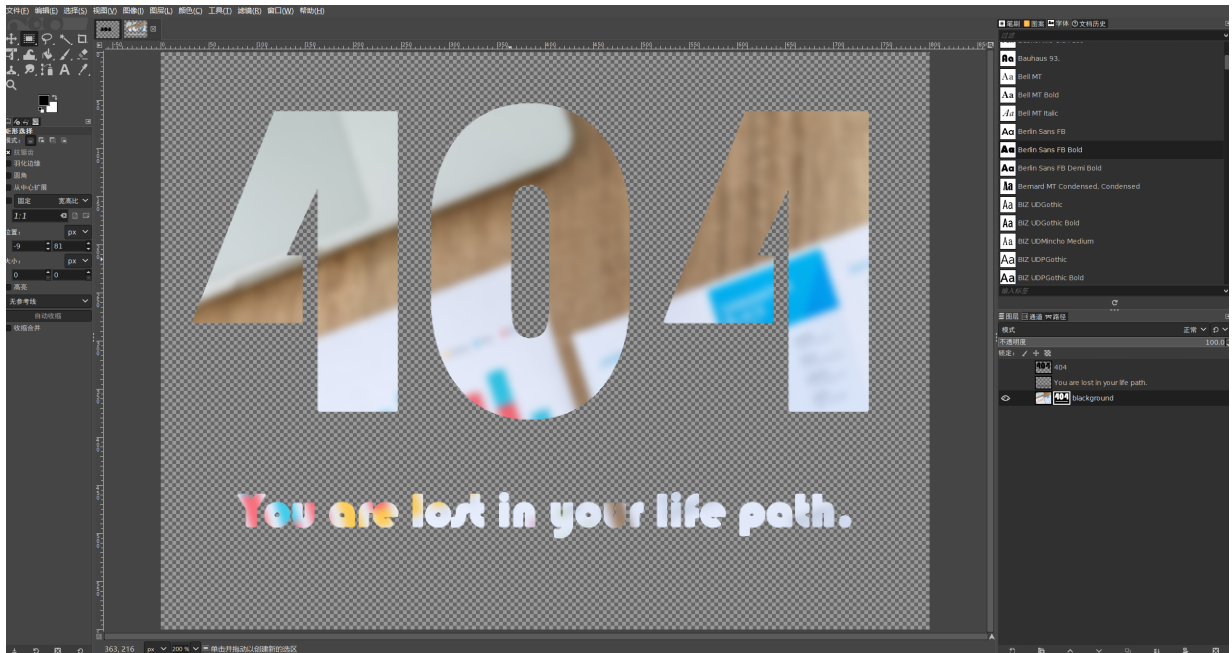


Figure 8: 404.png

### 2.2.5 SVG

In the use of SVG, we have used a variety of ways to construct SVG images to take full advantage of his benefits. We even created SVG animation on the home page. We will describe this in detail below.

- Basic SVG images

We used tools like Inkscape to draw simple SVG portraits, and since the team has the skills to operate adobe Kit experience, so we are light on SVG production.

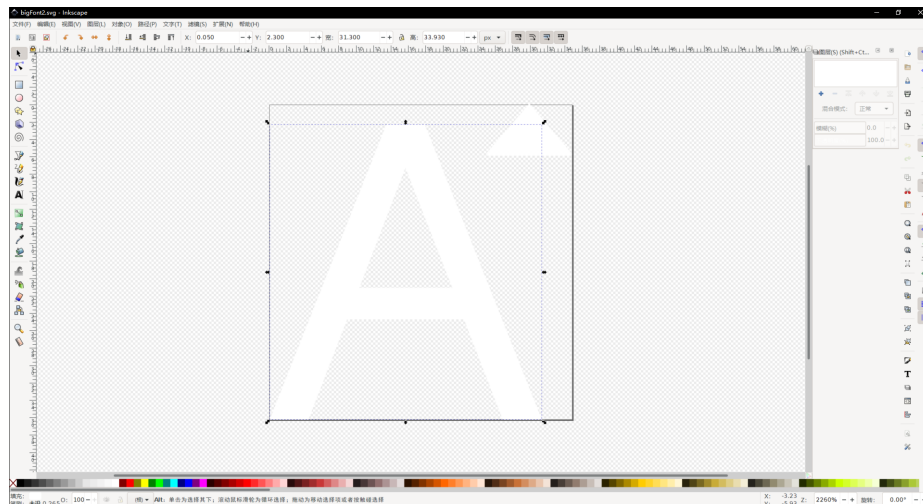


Figure 9: Inkscape

We used this basic graphical drawing to create 12 buttons, 3 of which are embedded in the page, while the remaining nine are used as Individual components are independent of the elements in the *src/components/button*. We take advantage of

the object-oriented component design of the Vue components so that each module is easy to maintain and update later.



Figure 10: buttonBar

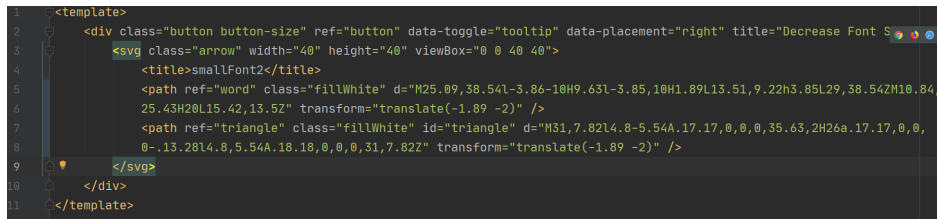


Figure 11: smallFontButton

- SVG-based css animation

We were not satisfied with making basic SVG graphics. We created four SVG animations with CSS animation effects. They are the start button on the home page, the continue and new buttons on the user-profile page, and the download button inside CVMaker. The most complex one is the download button, which activates the animation by changing the button's class when clicked.

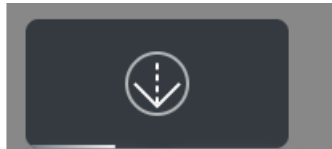


Figure 12: downloadButton animation

The download animation is divided into four parts, the first is the flashing of the outer ring, the second is the downward movement of the vertical line in the middle, and the third is the download of The middle arrow pattern becomes a checkmark when finished, and the fourth is the download progress bar at the bottom.



```

90 .button.downloading circle{
91   animation: 1.5s linear blink infinite;
92 }
93
94 .button.downloaded .arrow-top{
95   animation: 1s linear arrowTransform forwards;
96 }
97
98 .button.downloaded .checkmark{
99   opacity: 1;
100  stroke-dasharray: 100 100;
101  stroke-dashoffset: 100;
102  animation: 1s linear checkmarkTransform forwards 0.5s;
103 }
104
105 .button.downloaded .middle-line{
106   transition: 0.3s linear;
107   opacity: 0;
108 }

```

Figure 13: downloadButton animation

- svg animation based on vue-lottie

Of course, doing this will not satisfy our ambition to try the coolest animations. So we introduced the vue-lottie open-source package, which is based on the [lottie](#). Vue-lottie project vue architecture lottie can be interpreted as an SVG animation interpreter, and he supports the use of SVG in adobe After Effects exports complex animations to a JSON file and then self-rendering through the front-end of the web page to get cool effects.

We've made a dynamic animation on the home page to highlight our theme, which we're sure you've seen. We save the exported animation JSON file that we send to AE in the *src/view/index/assets/animation* folder.

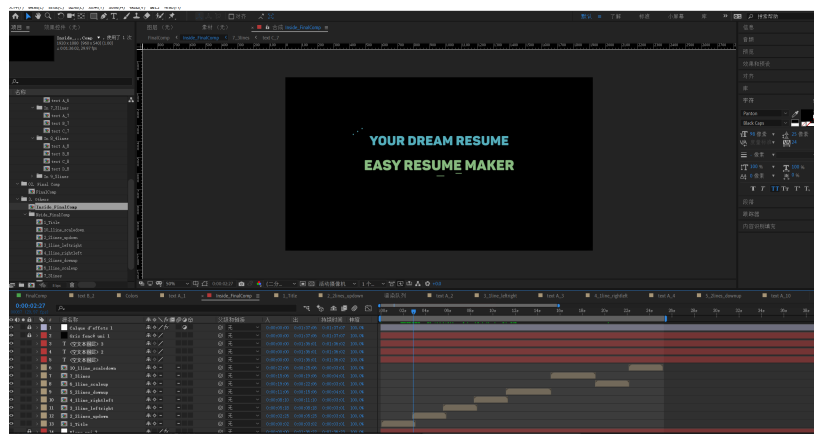


Figure 14: After Effect

Since this lottie tool is so new, we think we have made it pretty far ahead of the curve in terms of SVG usage.

## 2.3 Server Side

### 2.3.1 Server

### 2.3.2 Database

### 2.3.3 Dynamic pages

## 3 Working practices of the group

We used GitHub technology for remote collaboration, with Tao Xu handling the back-end technology and Yinan Yang is in charge of front-end technology. Our project address is <https://github.com/Nonac/webtech>. The screenshot below reflects the progress of our project.

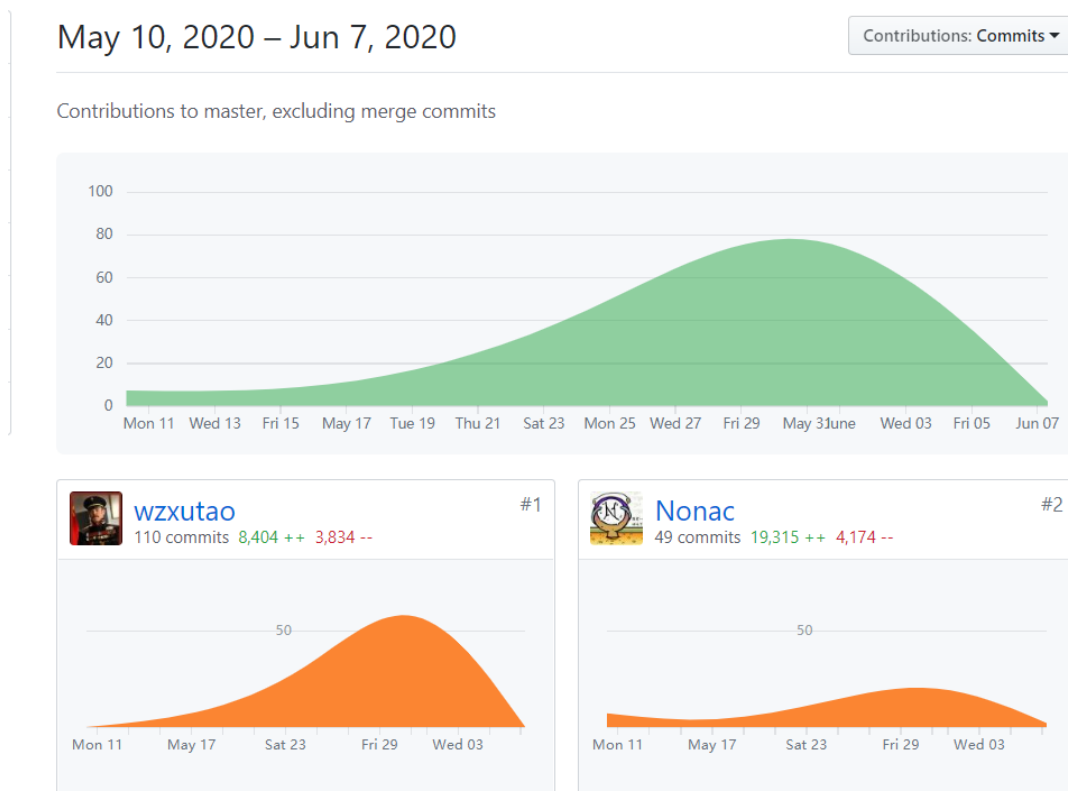


Figure 15: After Effect