DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

TECHNISCHE UNIVERSITÄT MÜNCHEN

Development of an Online 3D-Microfluidics Designing Platform

Student:	Eden Seet Jian Wei
Professor:	Prof. Dr.-Ing. Ulf Schlichtmann
Supervisor:	M.Sc. Yushen Zhang
Submission Date:	9 July 2021

Table of Contents

1 Introduction

Microfluidic systems are known for their usage of less volume of samples, chemicals and reagents while having adequate elements necessary for deep analysis. Majority of microfluidic chips are fabricated by the process of photolithography, which was invented in the 1950s. Photolithography is used in micro-fabrication to pattern either the bulk of a substrate or part of a thin film. Using light to transfer a geometric pattern from an optical mask to a light-sensitive chemical photoresist on the substrate. The well-developed technology derived from semiconductor fabrication which initially uses silicon [1]. Due to its rising demands with specific requirements e.g., specific optical characteristics, bio or chemical compatibility, reduced manufacturing costs and effortless prototyping, soft lithography was born. It is termed "soft" because it uses elastomeric materials, most notably Polydimethylsiloxane (PDMS) [2]. Soft lithography overcomes the limitations of photolithography when working with non-semiconductors materials, such as glass and polymers.

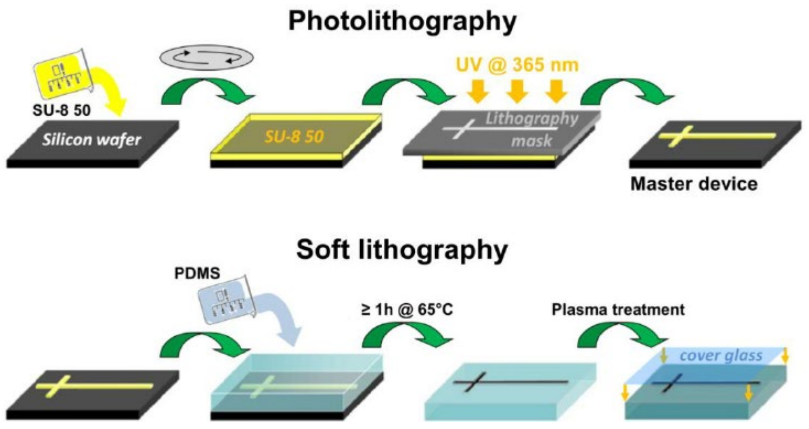


Figure 1: Photolithography vs Soft Lithography [3]

Commonly used soft lithography techniques are replica molding (REM), microtransfer molding (μT), microcontact printing (μCT) and micromolding in capillaries (MIMIC) which receive great results [4]. Thus, microfluidic chips are thriving in a few industries, notably the bio industry.

While microfluidic chips have many advantages, there is still a barrier with regards to mass production for prototyping and iterative design. Engineers experience the demanding process of designing microfluidic devices without much dedicated tools to ease it. Commonly, design iterations require the engineer to research architecture and manually draft out device outline, all while keeping the optimization of manufacturing in mind. While the microfluidic must work, it is stacked with the requirement to produce robust performance, causing the problem to be complex.

Hence, 3D-printing is introduced to ease the whole process. To utilize this technology, we can develop an online 3D-microfluidics designing platform.

In chapter 2 of this thesis, we will discuss more in depth about the problems that arise from fabricating microfluidic devices using PDMS by the conventional method of soft lithography. To counteract, we will be discussing the benefits of adopting 3D-printed microfluidic device over the former. Thus, leading us to the purpose and benefits of this thesis – development of an online 3D-microfluidics designing platform, to end off chapter 2. In chapter 3, we will discuss about the methodology and the elements needed to achieve it. In chapter 4, we will discuss about the implementation and realization, which includes explanation of the codes and their functionalities to create a functional design platform. In chapter 5, discussion on the results from the implementation. And in chapter 6, the conclusion.

2 Problem Statement

2.1 Overview

Most microfluidic devices use polydimethylsiloxane (PDMS). And even though PDMS-based microfluidic devices have been promising, some alarming disadvantages do exist. Due to its uneven pressure and/or leakage, PDMS devices often leads to flow issues. Though integration of PDMS devices have been reported, it usually consists of only two discrete devices. Hardly are there any reports which show the integration of several PDMS devices into a functional complex system. Moreover, the low efficiency of soft lithography, makes the fabrication process highly laborious. Tweaking the features of the device is almost impossible without fabricating a new piece when using PDMS [5]. All these disadvantages are a barrier which may constitute to the reason why large body of publications in microfluidics has not led to the realization of building practical systems to be used in other industries apart from biomedical engineering and chemistry field.

Alternatively, 3D-printing can be an efficient method to fabricate robust microfluidic devices while integrating beneficial and distinctive features into them. Although integrated components have been developed with PDMS devices, integrating several functional units on one device is tedious and costly. Since 3D-printing can produce any shape, there are reports of complex and functional 3D-printed microfluidic features.

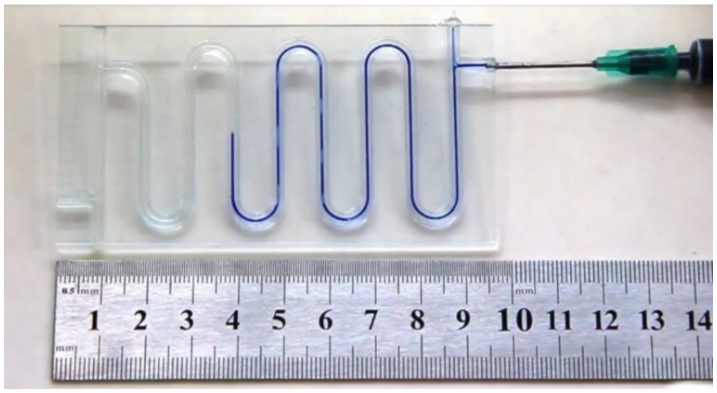


Figure 2: 3D-printed Microfluidic Model [6]

Therefore, for engineers to utilize this 3D-printing technology, it would require a platform to design the devices. The agenda of this thesis is to develop an online 3D-microfluidics designing platform. This platform serves to simplify and ease the burden of the engineers. It does that by transforming an extremely hands-on and tedious process into a simple, web-based, drag and drop one. Removing the need to draw the outlines of the device and doing architecture research [7]. It is also hassle-free and allows to be saved as a file for future expansion. Additionally, the file can be shared to other engineers to improvise. Lastly, it is a design platform which does not require any prior knowledge to it. Tools and functions on the online 3D-microfluidics designing platform are designed to primitive understanding.

2.2 Tasks

There are a few tasks for this thesis. They are:

1) Design and implement an interactive frontend user interface (UI)

2) Setup a demo environment

3) Deploy implemented user interface (UI)

3 Methodology

To produce an online design platform for 3D-printed microfluidics, there are three core programming languages that will be required. These are core languages because the design platform will not be possible without it. It is garnished with the help of one framework, one library and one runtime development platform.

3.1 Programming Language

3.1.1 HTML

HTML stands for hypertext markup language. It is the standard markup language for displaying content in a web browser which was invented in 1993. HTML elements make up HTML pages. With HTML construct, links and images can be embedded in the rendered page. HTML allows you to create organized documents by indicating structural semantics for text content such as lists, headings, paragraphs, links, and quotes. HTML elements are denoted by tags, which are written in angle brackets [8]. It is used to separate different tag elements. HTML tags are not directly displayed on the web browser, but instead, they are used to interpret contents of the webpage. Some examples of HTML are paragraph texts and tables.

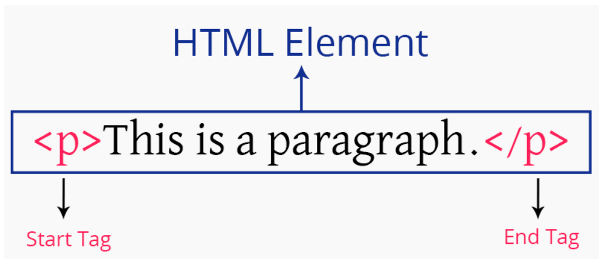
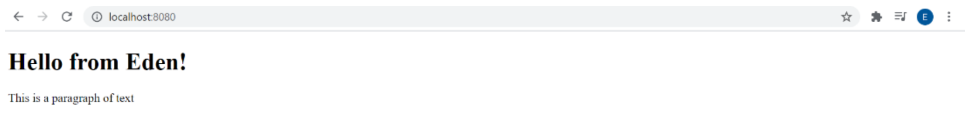


Figure 3: Example of a HTML element [9]

We will be utilizing HTML as the backbone of the design platform. The contents will be displayed on the design platform and grouped into their respective containers. We will utilize contents such as buttons to execute functions, icons to identify components, drop-down option list to select parts, as well as empty input fields to capture user's input specification values.

3.1.2 CSS

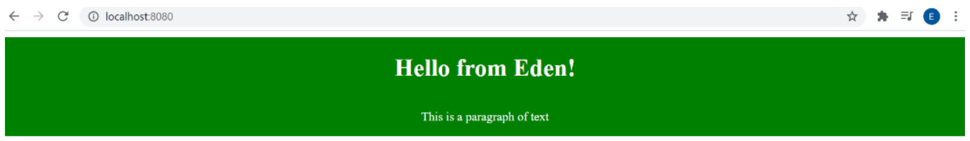
CSS stands for cascading style sheets. It is a style sheet language created in 1996, responsible for the presentation of the webpage which includes changing the appearance and layout. Presentation such as layouts, colors, and fonts can be isolated from content in CSS. Since it can be isolated, which means multiple web pages can share the same desired design after user specifies the relevant CSS file. Some advantages are more control and versatility in the specifications of presentation characteristics, increased content accessibility and reduced amount of repeated code to design every HTML element [10]. Some examples of CSS are setting the font, changing background colour, as well as aligning the contents.



Hello from Eden!

This is a paragraph of text

Figure 4: Before applying basic CSS characteristics to elements.



Hello from Eden!

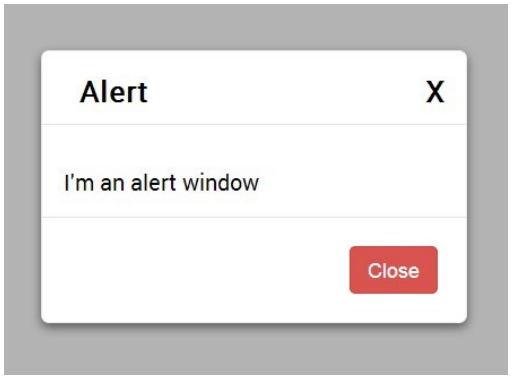
This is a paragraph of text

Figure 5: After applying basic CSS characteristics to elements.

We will be utilizing CSS to design the interface of the design platform. The interface will look neater and easier to use. For example, the contents will have appropriate spacing between each other, and the key buttons will be placed at optimal position such as the left and top of the screen, to capture user's attention. This will improve the user experience and ease of navigating around the design platform.

3.1.3 JavaScript

JavaScript is a text-based, high-level programming language created in 1995 and it is a cornerstone of web technologies alongside with HTML and CSS [11]. JavaScript is used to produce dynamic behaviours and special effects to the webpage that cannot be done with only HTML and CSS [12]. JavaScript is the most used programming language for good reasons [13]. Firstly, it is easy to learn with almost no requirement of prior coding knowledge. Secondly, most web browsers use JavaScript, and it is extremely versatile due to its involvement in areas like web development, mobile apps, as well as web servers. Some examples of JavaScript are animated 2D/3D graphics, popup windows and dynamic content updates on the webpage.



I'm an alert window

Figure 6: Popup window by utilizing JavaScript [14]

We will use JavaScript to get position of elements and obtain click events when user interacts with the design platform. We will also use JavaScript to edit the transition and animation of the sidebar menu.

3.2 Framework (frontend)

Framework is a pre-written application structure which allows user to build on top of. It is a collection of directories and files that you can add your own code and files to. This helps to ease the effort of the developer due to its additional tools and supported structures which the main language itself lacks [15].

3.2.1 Vue.js

Vue (pronounced as view) is a progressive framework for creating user interfaces formed in 2014. Derived from JavaScript, it is a framework which contains the best concepts from 2 other frameworks, namely AngularJS and ReactJS, and combined into one. Widely known to be user friendly, it is also simple to implement while having the flexibility of integrating with other components. Scalable at hand, it can be applied on huge single page apps, or it can be broken down to construct small, interactive parts for other technologies [16]. Compared to plain JavaScript, Vue.js saves time due to its progressive nature, since it removes the need to revise the code structure every time a programmer intends to add in new functionality. It also has an extreme advantage of superior performance while having more tools to support a page which changes content frequently and drastically.

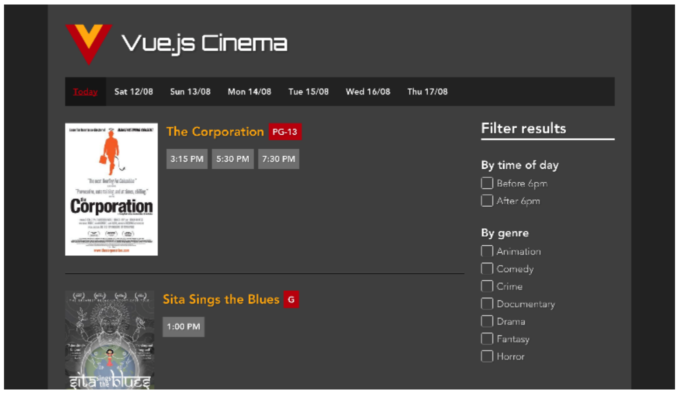


Figure 7: Webpage using Vue.js framework [17]

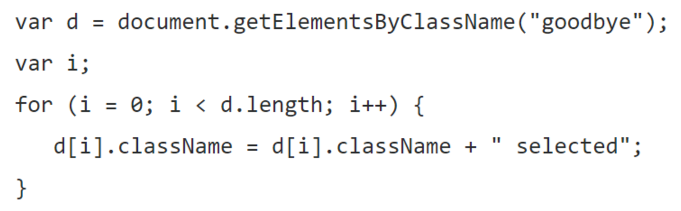
We will be using Vue.js to work on individual components such as **TopbarMenu**, **ShortcutMenu**, **DesignBoard**, and **SidebarMenu**, which will be discussed in chapter 4. Individual components will have its own function tested. We will also utilize the two-way data binding property of Vue.js which allows user to update the specification value of a part while concurrently display it on the design platform. Overall, we will integrate the components together for a functional design platform.

3.3 Library

Libraries are collections of prewritten code snippets which can be repeatedly utilized to achieve common functions [18]. They reduce the need for repetition which helps to keep the code concise and neat while saving time.

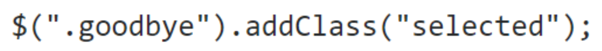
3.3.1 jQuery

jQuery is a swift, lightweight, and feature-rich JavaScript library created in 2006, which is designed to simplify the control of HTML elements [19]. With jQuery's motto of "write less, do more", it binds the multiple lines of JavaScript code required to accomplish common tasks into methods that can be called with a single line of code. Also, jQuery has the advantage of cross-platform compatibility [20]. This advantage saves developer's time, and the codes are easier to maintain since plain JavaScript would require additional lines necessary for cross-browser compatibility. Figure 8 and 9 below illustrate the lines of code needed to perform the same function by using plain JavaScript and jQuery respectively.



```
var d = document.getElementsByClassName("goodbye");
var i;
for (i = 0; i < d.length; i++) {
   d[i].className = d[i].className + " selected";
}
```

Figure 8: Code needed for a basic function in JavaScript [21]



```
$(".goodbye").addClass("selected");
```

Figure 9: Code needed for the same basic function in jQuery [21]

We will be using jQuery to access certain components of the design platform, such as **SidebarMenu**. It allows us to reach few levels deep in **SidebarMenu** without consuming much code and time.

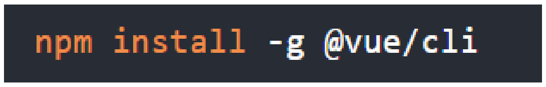
3.4 Runtime Environment

Runtime environment is an environment which an application or program is executed on. It consists of software components that facilitate the execution of code in real-time [22].

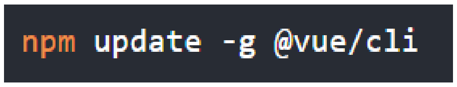
3.4.1 Node.js

Node.js is an open-source, cross-platform, back-end runtime environment which runs JavaScript code outside of a web browser. It allows the developers to write command line tools and run server-side scripts before the webpage is sent to the user's web browser to produce a dynamic webpage [23].

Specifically, we will be using node package manager(NPM) from Node.js. NPM oversee the packages which are local dependencies of a particular project, as well as globally installed JavaScript tools. With a single line of command, NPM can install the dependencies to be used in a project. Each dependency will display its installed version in the **package.json** file and allows developers to update the dependency to avoid undesired major changes [24]. These dependencies are either models and/or templates that save time from coding every single line from scratch. With its vast support from many programmers, it is the world's largest software library/registry. Therefore, this is a go-to when looking for suitable packages with code to cut down complexity.



```
npm install -g @vue/cli
```



```
npm update -g @vue/cli
```

Figure 10 and 11: Simple commands in NPM to install/update dependencies [25]

We will be using Node.js's NPM to install third-party packages such as **vue-konva**, **vuex**, **jQuery**, and **vue-sidebar-menu**. These are templates that we will build on top of to better suit the design platform, instead of coding from scratch.

4 Implementation

In this chapter, we will go in depth about the functions of each component and how they realize the objective of developing an online 3D-microfluidics designing platform.

4.1 Design Board

4.1.1 Background Grid

The background grid allows user to take reference to the grids and align the created parts to it. First, we design the background to look like a graph paper.

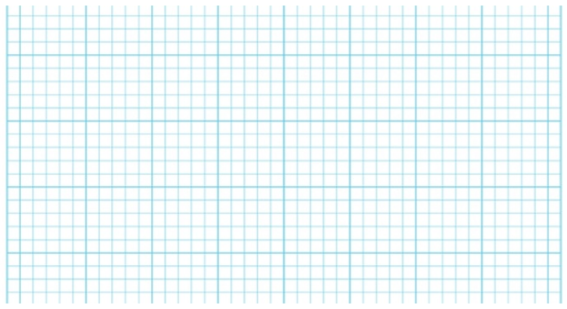


Figure 12: Graph paper structure [26]

To design, two types of lines with varied thickness and colour will be used. The internal lines (lines inside a single large square) need to be visible without being obtrusive. Since the grids are covering the whole page of the design platform, the obtrusive internal lines will cause the design platform to look extremely messy. This results to engineers having a hard time dragging and dropping. Therefore, the internal lines are set to be thin, 0.8 pixels, and light grey in colour. The external lines (perimeter of a large square) need to be thicker and more prominent. Its thickness is set to 1.5 pixels, and grey in colour. Next, we set the colour of the background to be slight orange. This is to enhance visibility since the board (part) will be white in colour.

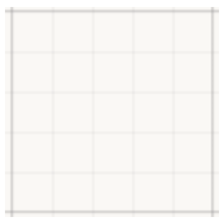


Figure 13: Designed single square grid

Lastly, we will reproduce the single square grid uniformly across the background to achieve a graph paper look.

4.1.2 Canvas

The design board has a canvas where user can design parts and allowed to drag and drop. This canvas is not visible to the user but is set on the design platform. The parts can only be dragged and dropped in the canvas. Therefore, we set the canvas's width and height to occupy 100% of the user's screen to enable the user to drag and drop anywhere on the design platform without obstruction.

Fig 14: Canvas holding shapes [27]

4.2 Sidebar Menu

The sidebar menu will be utilized for user navigation. It will consist of navigation items such as components, history log, as well as position of current component. This sidebar menu is a third-party package/dependency which is installed in this project. It is installed using Node.js's NPM, discussed in the chapter above. We will be changing some of its default settings while adding on necessary characteristics to better suit the design platform.

The sidebar menu is a sliding navigation bar, which allows user to expand and minimize. This is optimal because the user can open the sidebar menu to acquire the desired information of their objective and minimize the sidebar menu once the user is done, clearing more space for drag and drop on the design platform.

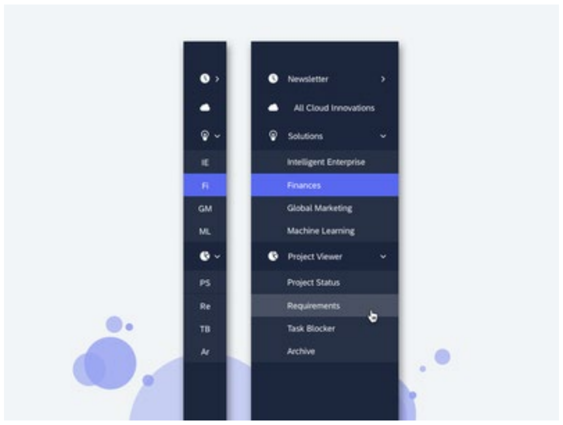


Figure 15: Example of expandable sidebar menu filled with drop-down main items [28]

To facilitate the expansion and minimization of the sidebar menu, there will be a button placed at the bottom of it. Each main item in the sidebar menu has a drop-down, showing its child items. The concept is similar to the sidebar menu in the figure above.

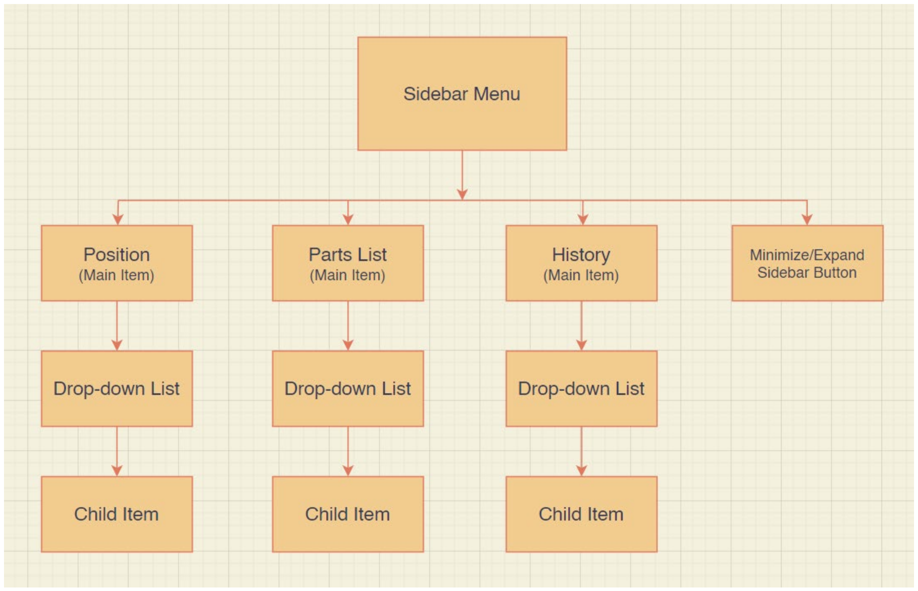
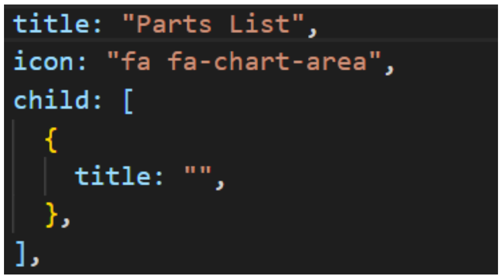


Figure 16: Hierarchy in sidebar menu

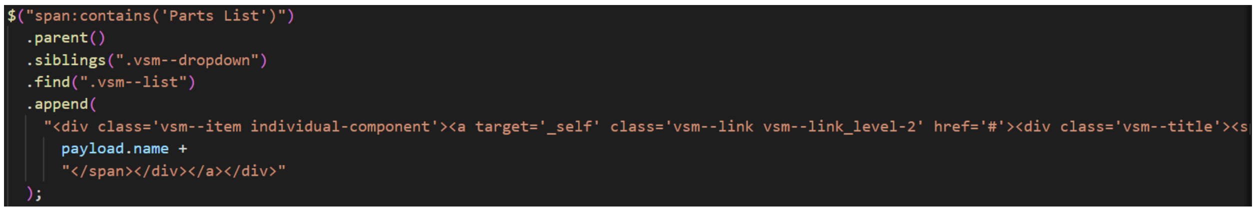
We can change the main item's name by changing the 'title' shown below in figure 17. Since user will be creating parts, we will need a main item that has a drop-down list to display them. Therefore, we set the main item's name to be 'Parts List'. Adding a part (child item) in the main item of 'Parts List' will require us to use the library, jQuery, mentioned above in chapter 3.3.1.



```
title: "Parts List",
icon: "fa fa-chart-area",
child: [
  {
    title: "",
  },
],
```

Figure 17: Segment of code for main items in sidebar menu

We will be using jQuery to access the drop-down of ‘Parts List’ and add the part (child item) in it. Whenever a user creates a part, this jQuery code will be executed, and that part will be added to the dropdown of ‘Parts List’. The whole process is similar for ‘History’ and ‘Position’, which are the two other main items in the sidebar menu. Figure 18 below shows the code that is used to add the part into the drop-down.



```
$("span:contains('Parts List')")
  .parent()
  .siblings(".vsm--dropdown")
  .find(".vsm--list")
  .append(
    "<div class='vsm--item individual-component'><a target='_self' class='vsm--link vsm--link_level-2' href='#'><div class='vsm--title'><s
      payload.name +
      "</span></div></a></div>"
  );
```

Figure 18: jQuery code to add child items in ‘Parts List’ of the sidebar menu

4.3 Shapes

Shapes are essential to the design platform. It replicates the shape of the part which user chooses to create. It also displays what will be printed on the 3D chip.

To create a shape, we will be using Konva. It is a package/dependency which is specifically built for drawing shapes and animations.

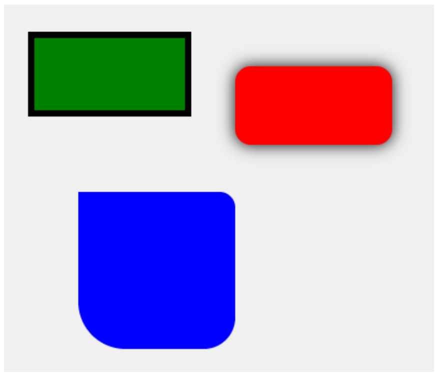
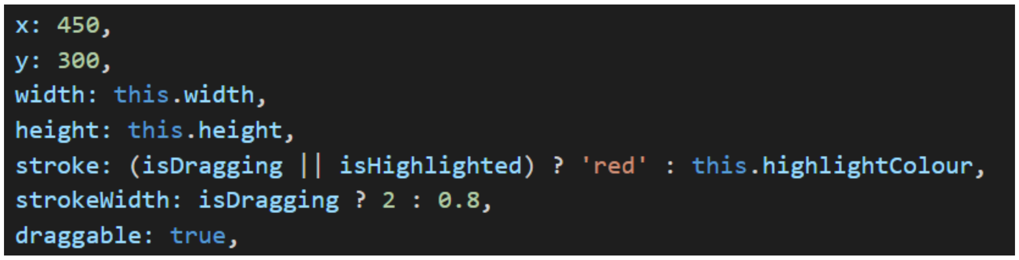


Figure 19: Drawing shapes using Konva [29]

First, we select the part to create. Different parts possess different shapes, which requires varying tag names because Vue interprets them to draw the shapes. For example, the tag **<v-rect>** represents the shape of a rectangle, **<v-circle>** represents the shape of a circle and **<v-line>** represents a line or a custom shape. Once we select the part, we can edit the shape properties of it. Properties such as thickness and colour of the stroke, as well as width and height of the shape can be adjusted.

Additionally, the position, X and Y coordinates, of the shape can also be adjusted. The keyed in value in X and Y coordinates will move the shape horizontally and/or vertically away with respect to the screen size. It is also the start-off after it is initially created. And to allow the shape to be dragged, we set its property, **draggable**, to true. To identify the current dragged part, we change the thickness to 2 pixels and colour of the stroke to red whenever drag happens. The thickness and colour of the stroke return to 0.8 pixels and black respectively after the part (shape) is dropped. Another function that utilizes shape property is highlight and unhighlight, which toggles the stroke colour between black and red when user clicks on the part.



```
x: 450,
y: 300,
width: this.width,
height: this.height,
stroke: (isDragging || isHighlighted) ? 'red' : this.highlightColour,
strokeWidth: isDragging ? 2 : 0.8,
draggable: true,
```

Figure 20: Editing the properties of the shape

4.4 Shortcut Menu

The shortcut menu consists of few buttons which will be often used. The shortcut menu will be placed close to the left border of the design platform and overlap the design board. This allows the user to reach without hassle. To improve visibility, the colour of the buttons is set to white to differ from the orange graph paper background. Each button is set to 40 pixels, it is a suitable size to get noticed by user without being too obtrusive while the user is using the design platform. Also, the buttons are vertically aligned with no spacing between them, reducing the amount of space it overlaps on the design board. Icon is fitted into each button and picked for primitive understanding; it serves to illustrate the function of the button.

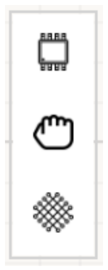


Figure 21: Shortcut Menu

4.5 Top Bar Menu

The top bar menu consists of several buttons which are seldomly used, and it is placed at the top of the design platform. Some of the buttons in the top bar menu are open file, undo, forward, delete, and save.

Similar to shortcut menu, the colour of the buttons in top bar menu is set to white, differing from the orange grid paper background to improve visibility. However, size of each button is set to 30 pixels. Since the several buttons on the top bar menu are horizontally aligned, it is appropriate to set 35 pixels of spacing between each button to prevent clustering. Icon is fitted into each button and picked for primitive understanding; it serves to illustrate the function of the button.

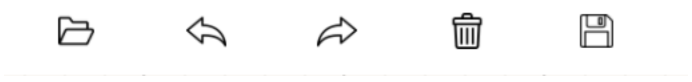
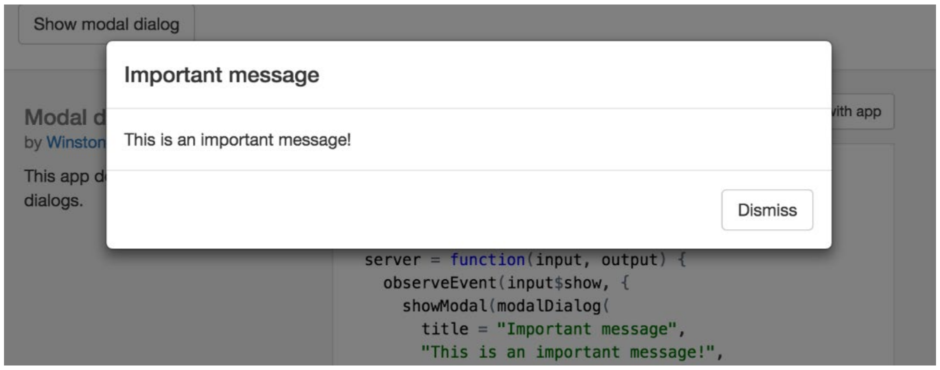


Figure 22: Top bar menu

4.6 Modal

A modal is used to interact with the user. In our case, it operates as a popup window to display parameter specifications and gather user inputs. Scenarios where we will use modal are, creating and editing components, as well as displaying netlist connection. The modals in all scenarios will have the same general design, although each will have its own size while formatting the contents differently. It is hidden from the design platform and will only be visible if the user triggers it.



This is an important message!

This app d
dialogs.

```
server = function(input, output) {
  observeEvent(input$show, {
    showModal(modalDialog(
      title = "Important message",
      "This is an important message!",
```

Figure 23: Example of modal with displayed content [30]

The trigger for modal to be fired is linked to a click and/or double-click event. For example, double-click of the part will trigger a modal for editing the part specification values and a single-click on ‘create part’ button will trigger a modal for creating part.

When the trigger for modal is fired, the modal will be displayed, and it will cast an overlay on top of the design platform. The background of the overlay will be darkened. This will direct user’s focus to the modal and user will not be able to interact with any contents outside of it.

4.7 State Management

Vuex is a state management library and pattern for Vue applications. It operates as a centralized store for all components in Vue. Vuex is not visible to the user, but it plays a vital role in the design platform. Without Vuex, we need to code different levels to pass and receive the data. It would be troublesome and ends up making the code messier. Moreover, when passing data in two levels (child to grandparent and vice versa) or more, it might get confusing.

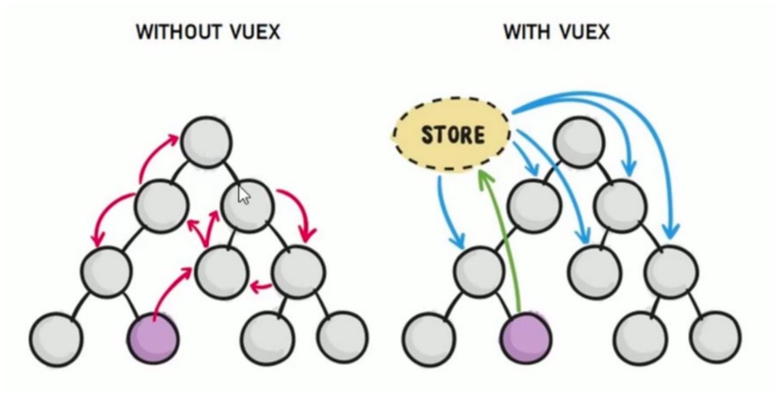
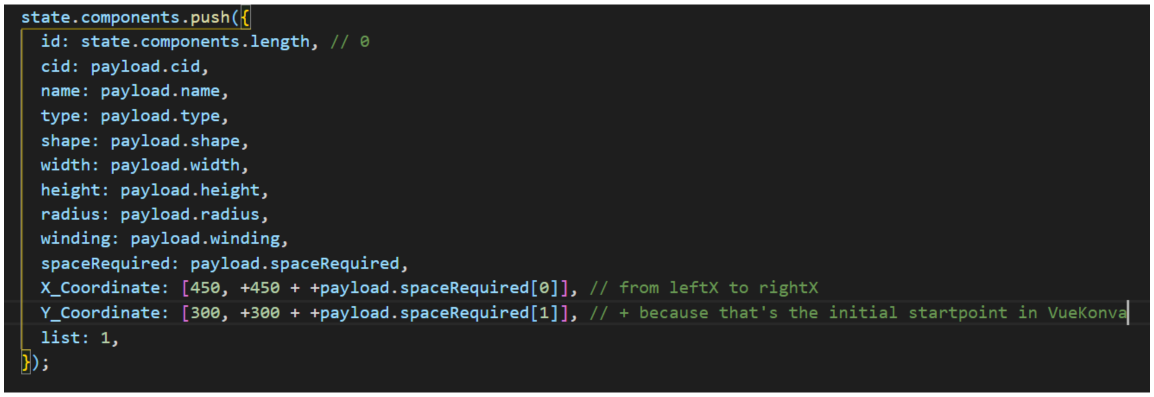


Figure 24: Comparison between using Vuex and without Vuex [31]

We will be executing codes that involves Vuex in a file named **index.js**, which is created to perform its function. Created parts' information are stored in Vuex store. This allows us to easily access them in different levels.



```
state.components.push({
  id: state.components.length, // 0
  cid: payload.cid,
  name: payload.name,
  type: payload.type,
  shape: payload.shape,
  width: payload.width,
  height: payload.height,
  radius: payload.radius,
  winding: payload.winding,
  spaceRequired: payload.spaceRequired,
  X_Coordinate: [450, +450 + +payload.spaceRequired[0]], // from leftX to rightX
  Y_Coordinate: [300, +300 + +payload.spaceRequired[1]], // + because that's the initial startpoint in VueKonva
  list: 1,
});
```

Figure 25: Adding a part into Vuex

After user inputs the specification values in part creation, we will gather them as an object array (payload) and dispatch it. They will be matched to Vuex's object array's key name, such as shape and width. Then, with the command 'push', that specific part with its information will be 'pushed', that will successfully add the part to the state, and we will be able to use it globally. The figure above shows the code for this process.

5 Results

5.1 Create Parts

To create a part, user clicks on the create part button which is located on the left side of the design platform. Once clicked, the 'Create New Part' modal will pop up as shown in figure 26 below.

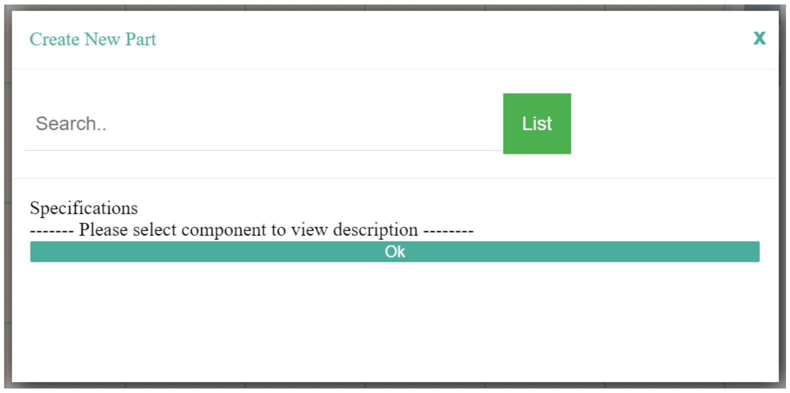

Create New Part
x
Search..
List
Specifications
------- Please select component to view description --------
Ok

Figure 26: 'Create New Part' modal

User can either click on the 'List' button to show a drop-down of the parts or type in and search. Then, user clicks on the part he/she intends to create. Upon clicking, the description and specification input fields will be shown as shown in figure 27 below.

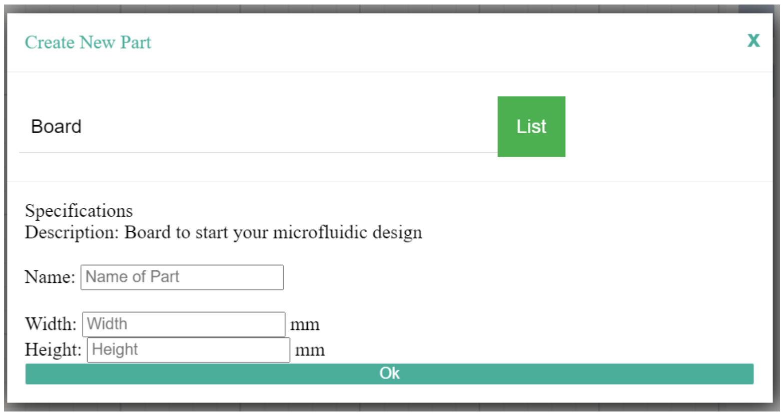

Create New Part
x
Board
List
Specifications
Description: Board to start your microfluidic design
Name: Name of Part
Width: Width mm
Height: Height mm
Ok

Figure 27: Part with its description and specifications

Once user is done filling in the input fields and clicks ‘Ok’ at the bottom of the modal, the part will be successfully created and be displayed on the design board itself. Subsequent parts follow the same process as the first.

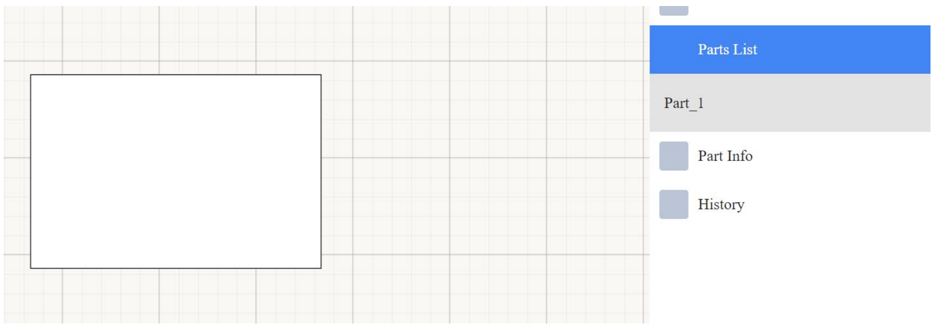


Figure 28: ‘Part_1’ is created. Appears on design board and sidebar menu as a child item of ‘Parts List’.

5.2 Drag and Drop

To perform drag and drop, user can click and hold on the part. The stroke colour will change from black to red, indicating that the part is currently dragged. And once user let go of the mouse click, the part will be dropped at the mouse position while the stroke colour returns to black.

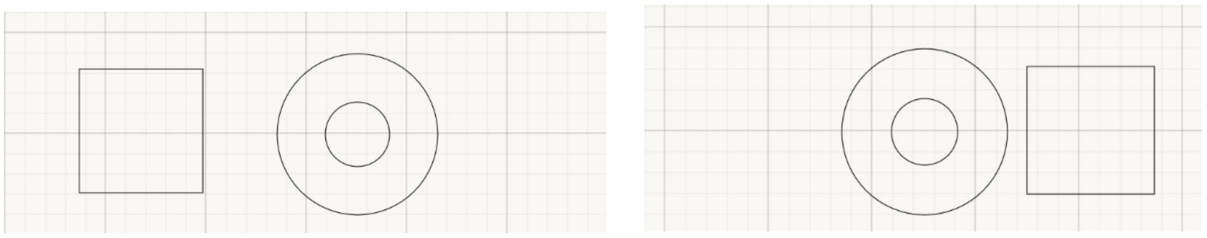
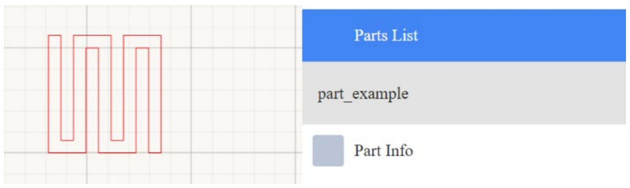


Figure 29 and 30: Before & After Drag

5.3 Delete Parts

To delete a component, user clicks on the desired part to delete. The stroke of the part will change from black to red, indicating that it is highlighted. Then, user clicks on the delete button at the top of the design platform. Part will be deleted and removed from the design board. The part will also be removed from ‘Parts List’ as shown in figure 31 and 32 below.



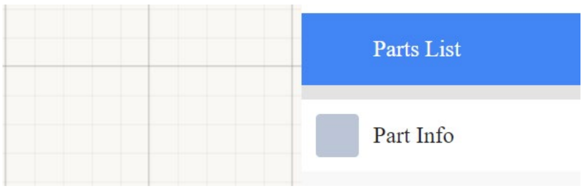
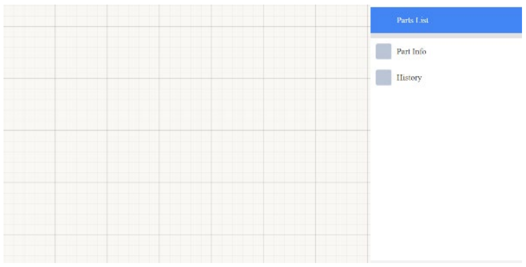


Figure 31 and 32: Before and after deleting part

5.4 Undo

To undo, user clicks on the undo button at the top of the design platform. It will undo the latest action. With undo, deleted parts can be retrieved and placed at original position.



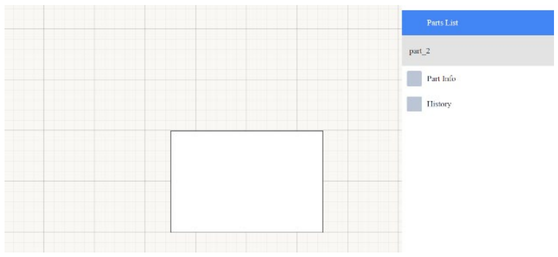
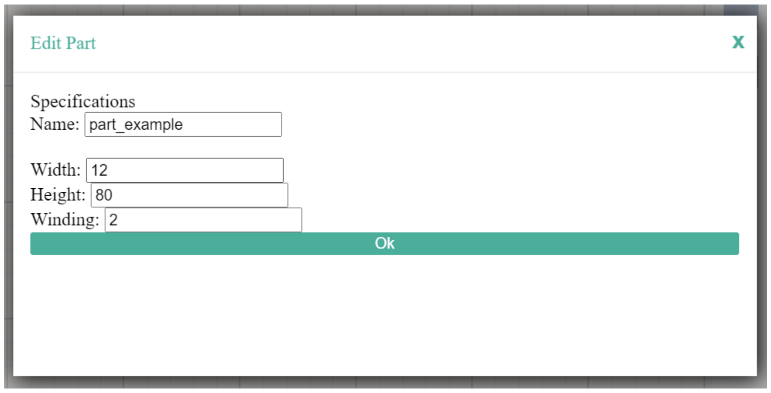


Figure 33 and 34: Before and after ‘undo’ to retrieve part

5.5 Edit Parts

To edit the part, user can double-click on the part and the ‘Edit Part’ modal will pop up. It will display the current values of the specifications.


Edit Part

Specifications
Name: part_example

Width: 12
Height: 80
Winding: 2

Figure 35: ‘Edit Part’ Modal

User can fill in the new values and press ok. The new values will be updated and the part changes.

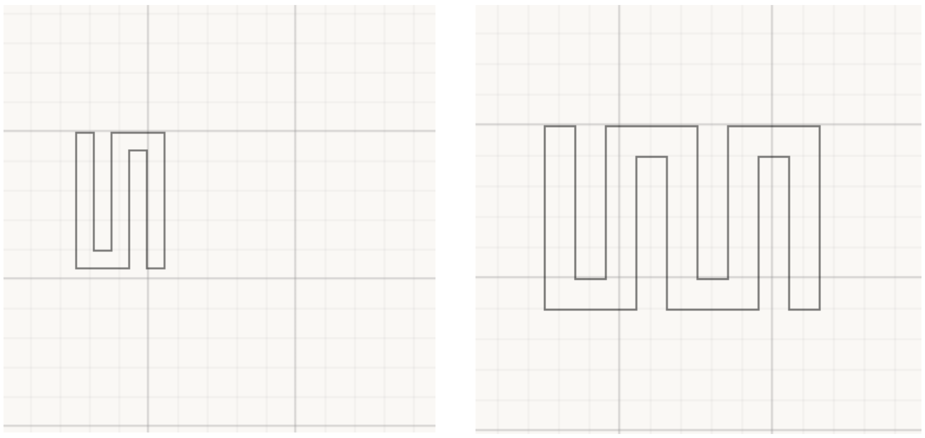


Figure 36 and 37: Before and after editing the part.

6 Conclusion

In this thesis, we have introduced the online 3D-microfluidics designing platform. It allows the engineers to design the microfluidics easily by simple drag and drop, removing the arduous and manual process of drafting the microfluidics while saving time and cost of materials. Thus, encouraging engineers to utilize the online 3D-microfluidics designing platform.

Currently, the design platform displays the designed parts in bird’s-eye view, which is in 2D. A suggestion would be to improve this design platform and turn it into 3D, where the engineer can set specifications and be able to view the designed parts in 3D, enabling a fuller grasp of the microfluidics design.

7 References

[1] A. Dellaquila, "The History of Microfluidics," Unknown. [Online]. Available: https://www.elveflow.com/microfluidic-reviews/general-microfluidics/history-of-microfluidics/. [Accessed 1 July 2021].

[2] Wikipedia, "Soft Lithography," 12 December 2020. [Online]. Available: https://en.wikipedia.org/wiki/Soft_lithography. [Accessed 1 July 2021].

[3] J. T. L. A. W. T. Ma Yujie, "Fabrication of Microfluidic Devices Combining Photolithography and Soft Lithography," November 2013. [Online]. Available: https://www.researchgate.net/figure/Fig-S1-Fabrication-of-microfluidic-devices-combining-photolithography-and-soft_fig1_258444837. [Accessed 1 July 2021].

[4] Elveflow, "Introduction to PDMS soft-lithography and polymer molding for microfluidics," Unknown. [Online]. Available: https://www.elveflow.com/microfluidic-reviews/soft-lithography-microfabrication/introduction-about-soft-lithography-and-polymer-molding-for-microfluidic/. [Accessed 1 July 2021].

[5] B. T. M. A. S. M. A. D. T. D. M. S. R. S. M. Chengpeng Chen, "3D-printed Microfluidic Devices: Fabrications, Advantages and Limitations - a Mini Review," 21 August 2016. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5012532/. [Accessed 2 July 2021].

[6] B. Goldschmidt, "3D Printing Microfluidic Models - 5 Most Interesting Projects," 26 July 2019. [Online]. Available: https://all3dp.com/2/3d-printing-microfluidic-models-most-interesting-projects/. [Accessed 4 July 2021].

[7] Y. Zhang, "Frontend Development, Online 3D-Microfluidics Designing Platform," Unknown. [Online]. Available: https://www.ei.tum.de/en/eda/theses-jobs/open-positions/. [Accessed 4 July 2021].

[8] Wikipedia, "HTML," 29 June 2021. [Online]. Available: https://en.wikipedia.org/wiki/HTML. [Accessed 3 July 2021].

[9] ljhsherman, "40 HTML interview questions for beginners & experienced," 23 February 2021. [Online]. Available: https://www.hackertrail.com/talent/frontend/html/40-html-interview-questions-for-beginners-experienced/. [Accessed 3 July 2021].

[10] Wikipedia, "CSS," 30 June 2021. [Online]. Available: https://en.wikipedia.org/wiki/CSS. [Accessed 3 July 2021].

[11] Wikipedia, "JavaScript," 21 June 2021. [Online]. Available: https://simple.wikipedia.org/wiki/JavaScript. [Accessed 3 July 2021].

[12] M. Contributors, "What is JavaScript?," 19 June 2021. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript. [Accessed 3 July 2021].

[13] Unknown, "11 Most In-Demand Programming Languages in 2021," 2021. [Online]. Available: https://bootcamp.berkeley.edu/blog/most-in-demand-programming-languages/. [Accessed 3 July 2021].

[14] Unknown, "Simple Custom Dialog Popup Plugin With jQuery," 2014. [Online]. Available: https://www.jqueryscript.net/other/Simple-Custom-Dialog-Popup-Plugin-With-jQuery.html. [Accessed 3 July 2021].

[15] M. Pekarsky, "Does your web app need a front-end framework?," 3 February 2020. [Online]. Available: https://stackoverflow.blog/2020/02/03/is-it-time-for-a-front-end-framework/. [Accessed 5 July 2021].

[16] V. Editors, "Vue v2 Guide," Unknown. [Online]. Available: https://vuejs.org/v2/guide/. [Accessed 4 July 2021].

[17] A. Gore, "Migrating a Vue.js App to Vuex," 4 Feb 2020. [Online]. Available: https://dzone.com/articles/migrating-a-vuejs-app-to-vuex. [Accessed 4 July 2021].

[18] S. Khan, "What is a JavaScript Library?," Unknown. [Online]. Available: https://generalassemb.ly/blog/what-is-a-javascript-library/. [Accessed 4 July 2021].

[19] Unknown, "What is jQuery?," Unknown. [Online]. Available: https://www.javatpoint.com/what-is-jquery. [Accessed 3 July 2021].

[20] Wikipedia, "jQuery," 17 June 2021. [Online]. Available: https://en.wikipedia.org/wiki/JQuery. [Accessed 3 July 2021].

[21] C. Castiglione, "jQuery vs. JavaScript | What’s the difference?," 19 September 2016. [Online]. Available: https://learn.onemonth.com/jquery-vs-javascript/. [Accessed 3 July 2021].

[22] T. Editors, "Runtime Environment (RTE)," Unknown. [Online]. Available: https://www.techopedia.com/definition/5466/runtime-environment-rte. [Accessed 4 July 2021].

[23] Unknown, "Node.js," 5 July 2021. [Online]. Available: https://en.wikipedia.org/wiki/Node.js. [Accessed 5 July 2021].

[24] Unknown, "npm (software)," 4 July 2021. [Online]. Available: https://en.wikipedia.org/wiki/Npm_(software). [Accessed 5 July 2021].

[25] V. Editors, "Installation," 23 September 2020. [Online]. Available: https://cli.vuejs.org/guide/installation.html. [Accessed 5 July 2021].

[26] HP, "Printables; Graph Paper: Wide," Unknown. [Online]. Available: https://printables.hp.com/us/en/printable/graph-paper-wide-productivity-worksheets-hp. [Accessed 7 July 2021].

[27] rafaesc, "Drawing Complex Canvas Graphics With Angular – ng2-konva," 23 December 2017. [Online]. Available: https://angularscript.com/drawing-complex-canvas-graphics-angular-ng2-konva/. [Accessed 8 July 2021].

[28] Dribble, "Side Menu," 2021. [Online]. Available: https://dribbble.com/tags/side_menu. [Accessed 7 July 2021].

[29] K. Editors, "HTML5 canvas Rect Tutorial," Unknown. [Online]. Available: https://konvajs.org/docs/shapes/Rect.html. [Accessed 7 July 2021].

[30] W. Chang, 17 August 2016. [Online]. Available: https://shiny.rstudio.com/articles/modal-dialogs.html. [Accessed 7 July 2021].

[31] Sunflower, 3 September 2020. [Online]. Available: https://www.sinocalife.com/ways-of-vue-components-communication-and-how-to-use-vuex-state-manager. [Accessed 8 July 2021].