Molly Elders Gallagher

N00160464

Vue College Application

CA2 WAF

Contents

[Introduction 2](#_Toc7390299)

[Vue Elements 2](#_Toc7390300)

[Vue Components 2](#_Toc7390301)

[Vue Router 2](#_Toc7390302)

[Vue Component Structure 4](#_Toc7390303)

[Content 4](#_Toc7390304)

[Methods 4](#_Toc7390305)

[Ajax Library & REST API 6](#_Toc7390306)

[Axios 6](#_Toc7390307)

[Testing & Debugging 7](#_Toc7390308)

[HTTP request client 7](#_Toc7390309)

[Vue dev tools 7](#_Toc7390310)

[Inspector debugger 7](#_Toc7390311)

[Npm run watch 7](#_Toc7390312)

# Introduction

This assignment required the development of a single page Vue application so that a user could view and manipulate data pulled from a database. This data is related to students, enrolments and courses of a college. This data is to be accessed through a REST API. This API supports a series of requests that must be implemented, and authorization is to be applied. The REST API will be implemented using Laravel. The front end of the application will be implemented using Vue. This front-end application will allow a user to log in, once this user has logged in, they can create, read, update and delete students, enrolments and courses. There are certain constraints applied to prevent some functionality to insure data integrity is kept within the database.

# Vue Elements

## Vue Components

A Vue component is a feature of vue.js that allows you to reuse basic HTML elements to reduce and eliminate the repetition of code. It also simplifies pages on a webpage to components being swapped in and out on a single page, this reduces the amount of code written. This can be done through a basic Vue file. Each component contains content, data and methods. These components are mapped to a path. As this is a single page application the use of components is very important. In the welcome page there are two component place holders. When a button is clicked with a different route the component mapped to that route will be called and will replace the component that is in the place holder. As the page will not refresh the components are simply swapped in and out. Everything passed the hashtag in the URL of the browser shows where the user is on the page and everything before the hashtag shows what page the user is on.

## Vue Router

In Vue to generate a single page application the routes are very important in relation to mapping components. Vue has a VueRouter class that creates a VueRouter object in this case that we have defined as router. Router contains an array or routes that we can user throughout the application.

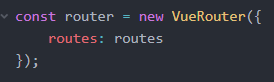


Figure VueRouter Object

As this application is a single page application this means the page will not refresh or change to a different page when a button is clicked, it will simply replace the component in the router-view place holder. A router-view is a Vue tag that is placed in the welcome.blade page, whenever a button is clicked and linked to a component the new component will search for this place holder and replace what is there.

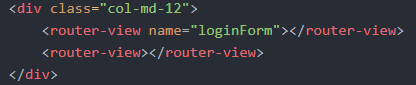


Figure Router View Placeholder

As seen in figure 2 loginForm is a nested component for when the application is loaded it will display the login component. If the user is logged in successfully this component will be replaced with the user home component.

As the page is never reloaded when each component is called the URL will update but the page will never change. This is made possible through the VueRouter object containing the routes. Each route is mapped to a component which is assigned a name a path and a component.

As said above within the URL everything before the # is the page the user is on and everything after the hash is where the user is on the page. After the # is the path that is mapped to the route and component, this indicates which component has been loaded.

These routes are mapped in the app.js file



Figure Routes Array

As seen above in figure three in the array of route objects, each object has a name a path and components. It is also possible for a route object to have children also seen in figure three. If the user is successfully logged in it will take them to the user home component. Only if your logged in and on the user home page will the user be able to access the user home children components. In this case to create read edit and delete enrollments.

This is all possible as the Vue object contains the Vue router object as a property along with the el mounted property. This el is directed to the id #app, within the element with the id #app the vue in tag attributes can be used such as v-for and v-if.

# Vue Component Structure

A component consists of content to be put on the browser, it is a Ui component that is displayed on the screen. A component contains content, methods and data.

Content

Within a component the structure starts with a template tag, within the template tag are a series of html tags that format the component to the way we want the elements to be displayed on the screen. Within these elements vue attributes are used to determine what show or hides in relation to the conditions used. In the example below there is an if condition, if the array is empty display its empty if not display the following.



Figure Vue attributes

Within this content structure variables are used to reduce hard coded data. These variables as seen in figure 4 come from the data function which is another feature a component is made up of.

Below the template tag there is a script tag that contains the JavaScript needed to be executed in this component. This script tag contains an export default function, this will be executed every time this component is used.

## Methods

After content methods is another factor that makes up a component. Within the default function methods can be created to use within the component itself, one of these functions is the mounted function. The mounted function is apart of the vue life cycle event. Mounted is a life cycle method, this means it is invoked and executed when the component is loaded or updated. Every time the component is loaded to the screen this function will be executed first even if its not called.

Data is another function within the script tag that is used. Data returns an object in which the objects properties can be used as variables throughout the component. As you can see in figure 5 the data function is returning an enrolments object which is an array of enrolments. In figure 6 you can see the properties of the enrolments object can be used as variables throughout the component.

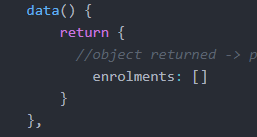


Figure data function

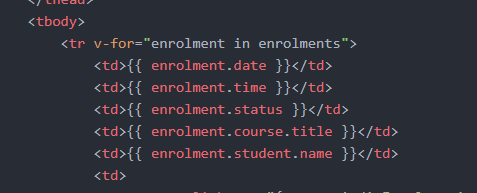


Figure data object properties used as variables

Other functions that can be created by the developer also goes within this script tag; these fucntions are stored in a methods array. And each function defined within this array can also be used within the component. For example within the component I created a deleteEnrolment method in which contained code to delete enrolment, this function can be called within the component within a button as seen in figure 7.



Figure Method used from default function

Style tags can also be used within the structure of a component. If a style tag contains the attribute scoped this means that the style defined within these tags apply to this component only.

# Ajax Library & REST API

## Axios

Axios is an Ajax library which we use in this Vue application. This http client library runs on both the server side and the client side. This means within Vue components Axios can be used to get data from a method or Lifecyle method. Axios being a promise-based HTTP client it can display and consume data from the REST API. An API is an interface that contains a number of definitions, protocols and methds that uses http requests to retrieve an manipulate data from the database using the principles of REST, this consist of POST, GET, PUT, PATCH and DELETE requests. The API created in this application is a Laravel controller. An example of using Axios with the rest API is the login.

When authenticating a user login with an API Laravel makes it easier using Laravel passport, this provides a serve implemented into our application. To determine a user has logged In successfully tokens are created and stored in local storage so it can be called again when another http request is sent. When the browser is loaded, and a user clicked the login button a post request is sent. This request will be in the form if a json object that will contain the email and password within the request body and a token within the header, this token is cross referenced to see if the token is already being used. The server will send a response of HTTP 1.1 200 OK, this means login has been successful

# Testing & Debugging

## HTTP request client

Testing the connection to the database through requests and responses were conducted by using chrome extensions. These chrome extensions are REST clients that allow you to send requests and examine the response. This was very helpful in the implementation on the login as testing the authentication using tokens within the header and to test if the request to access the CRUD functionality of the college data was only if the users are logged in. This process of testing led to a greater understanding of request headers, authentication, use of tokens and http responses.

## Vue dev tools

Chrome also contains a dev tool that can be installed, I found this very useful as it picked up on Vue attributes and characteristics and advised where they were coming from within the console. This dev tool aloud to vue the components and also vue the objects being returned and created.

## Inspector debugger

Within chrome the debugger was used to step through different lines of JavaScript code and into functions to see where the code was failing to execute. This identified the area of the problem and it could be fixed.

## Npm run watch

When using Vue node is needed to be installed, when editing JavaScript code, it needs to be compiled. With running npm run watch within the command prompt the files get compiled and watches the files so every time a change is made and saved it will be developed and compiles. If something goes wrong an error will appear in the command prompt window. This was also very useful in terms of debugging for more fatal errors made in the code that couldn’t be compiled successful.

Note: node modules and vendor file have been deleted from the application code