Slots –

NamedSlot – used to render elements at specific place

For this we have to use v-slot in our template

Form – need to use v-model directive on form element for two way data binding

To validate vue forms you can make use of vue library vee-validate and with that you use schema yup that has predefined validation rules in it.

To install vee-validate for vue3 command is

Npm install vee-validate@next

To install yup

Npm install yup

Router - SPA – single page application –

Router are needed to design single page application – it avoids reloading of your page.

createRouter is function from vue-router library and it used to create the routers in vue

createWebHistory –

History Modes – 1) HashMode – it is created using createWebHashHistory() function and it adds # in the url everytime, this may create a problem during authentication.

http://myurl.com/#

2)HTML5Mode - this mode never make changes to baseURl, it will be sent to server as it is so this preferrable mode.

<a href=””/> - don’t use this for creating router-link – it reloads the page

So as we don’t want to reload the page instead of using <a href=””/> you should make use of <router-link>

<router-view/> - need to be used in your main file for placing the view generated by on click of router-link

Scoped style

Slots – vue implements content distribution API using <slot>. This will help you to distribute your contents at respective places.

NamedSlots – If you want to render or distribute contents at specific places then you should use NamedSlot. To use namedSlot you have to use v-slot directive on your template.

While creating slot you need provide specific name to it using name attribute.

Teleport – whenever components are rendered elements aree placed as part of nested divisions. So instead of placing them in nested divisions if we want place them outside component but in application DOM then it is possible using Teleport

{{}} – interpolation – it is one way data binding – it used to present data on UI.

Forms in vue – will use to accept data from use and to perform 2 way data binding

On fields of form we have to use v-model directive.

npm I vee-validate@next –save

We can form validation using vee-validate higher order components I.e Form, Field and ErrorMessage

npm install –save yup

Yup is JS schema builder that will help to validate fields using its built in rules

Router is used to create SPA. It avoids reloading of page. It will connect with server only when there is update in data or if data changes, to get the changes it makes asynchronous call with server and update data on UI.

createRouter – it is function to create routers

History Modes : it is created using createWebHistory().

Hash Mode – it uses # before actual URL internally passed. #does not have any importance on server but it may create a problem in SEO due to this it is not preferred.

HTML 5 Mode - preferred way. It keep url as it is as a base url.

<router-link> component is preferred to create links in place of <a href=””>

That avoids reloading of page

And to represent view from router configured components you have to <router-view> component.

<router-view> is a functional component that renders respective component depending matching router path.

Reference links

* Scoped Styles

<https://vue-loader.vuejs.org/guide/scoped-css.html>

Videos :

<https://www.youtube.com/watch?v=T0yVKIcBkK0>

* Introducing Slots

<https://v3.vuejs.org/guide/component-slots.html#slot-content>

Videos :

<https://www.youtube.com/watch?v=emi436qg9mg>

* Named Slots

<https://v3.vuejs.org/guide/component-slots.html#named-slots>

Videos :

<https://www.youtube.com/watch?v=8lbo7olMzVA>

* Slot Styles & Compilation

<https://vegibit.com/vue-slots-tutorial/>

Videos :

<https://www.youtube.com/watch?v=g_5jvUz7q80>

* Scoped Slots

<https://v3.vuejs.org/guide/component-slots.html#scoped-slots>

Videos :

<https://www.youtube.com/watch?v=i0E9miwWInY>

* Dynamic Components & Keeping Dynamic Components Alive

<https://v3.vuejs.org/guide/component-dynamic-async.html#dynamic-components-with-keep-alive>

<https://vegibit.com/vuejs-dynamic-components/>

* Teleporting Elements

<https://v3.vuejs.org/guide/teleport.html>

Using With Vue Component

<https://v3.vuejs.org/guide/teleport.html#using-with-vue-components>

Using Multiple Teleport on same target

<https://v3.vuejs.org/guide/teleport.html#using-multiple-teleports-on-the-same-target>

Videos :

<https://vueschool.io/lessons/vue-3-teleport?friend=vuejs>

<https://www.vuemastery.com/courses/vue-3-essentials/teleport/>

* Working with Fragments

<https://v3.vuejs.org/guide/migration/fragments.html>

Videos :

<https://www.youtube.com/watch?v=iC9smVwm7GE>

**Module 8 : Forms:**

* v-model & Inputs

<https://v3.vuejs.org/guide/forms.html>

* Working with v-model Modifiers and Numbers

<https://v3.vuejs.org/guide/forms.html#modifiers>

* v-model and Dropdowns

<https://v3.vuejs.org/guide/forms.html#select-options>

* Using v-model with Checkboxes & Radiobuttons

<https://v3.vuejs.org/guide/forms.html#checkbox-2>

<https://v3.vuejs.org/guide/forms.html#radio-2>

<https://v3.vuejs.org/guide/forms.html#radio>

* Adding Basic Form Validation

Videos :

<https://www.youtube.com/watch?v=ixOcve5PX-Q>

* V-model on Components

<https://medium.com/javascript-in-plain-english/vue-3-v-model-and-components-de2d39527f80>

<https://v3.vuejs.org/guide/forms.html#v-model-with-components>

Videos :

<https://www.youtube.com/watch?v=5TFelzoDMBQ>

**Module 9 : Routers:**

* What & Why?
* Routing Setup

<https://medium.com/javascript-in-plain-english/a-first-look-at-vue-router-in-vue3-253bc61b2cf5>

<https://www.codemag.com/Article/2101051/Routing-in-Vue3-Navigating-the-Options>

Videos :

<https://www.youtube.com/watch?v=I7pRp1Bwysw>

* Registering & Rendering Routes
* Navigating with router-link
* Passing Data with Route Params (Dynamic Segments)

<https://www.vuemastery.com/blog/vue-router-a-tutorial-for-vue-3/>

Videos :

<https://www.youtube.com/watch?v=-jeoyDJDsS>

<https://vueschool.io/lessons/vue-router-named-routes-and-params>

* Styling Active Links

Videos :

<https://www.youtube.com/watch?v=Nf8qlfgtfbY>

<https://vueschool.io/lessons/vue-router-active-class>

* Updating Params Data with Watchers

<https://forum.vuejs.org/t/rerendering-component-on-route-param-change-recalling-created-hooks/9536/6>

* Videos :

<https://www.youtube.com/watch?v=CF_T8xjzmg4>

* Passing Params as Props

<https://router.vuejs.org/guide/essentials/passing-props.html#boolean-mode>

* Videos :

<https://vueschool.io/lessons/how-to-pass-vue-router-params-as-props-to-components>

* Using Nested Routes

<https://www.codemag.com/Article/2101051/Routing-in-Vue3-Navigating-the-Options>

* Videos :

<https://vueschool.io/lessons/vue-router-nested-routes>

Assignments :

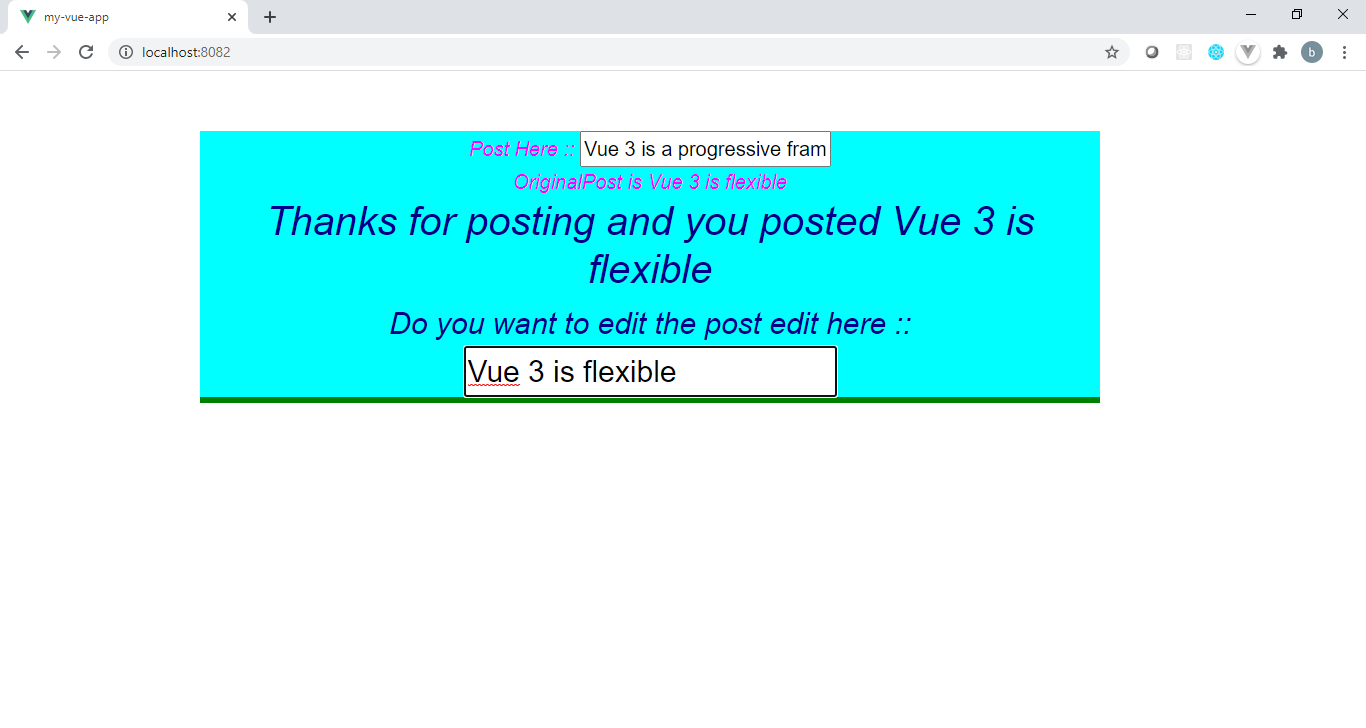
7.3 Consider problem statement 7.2 and generate below given view using PostComponent



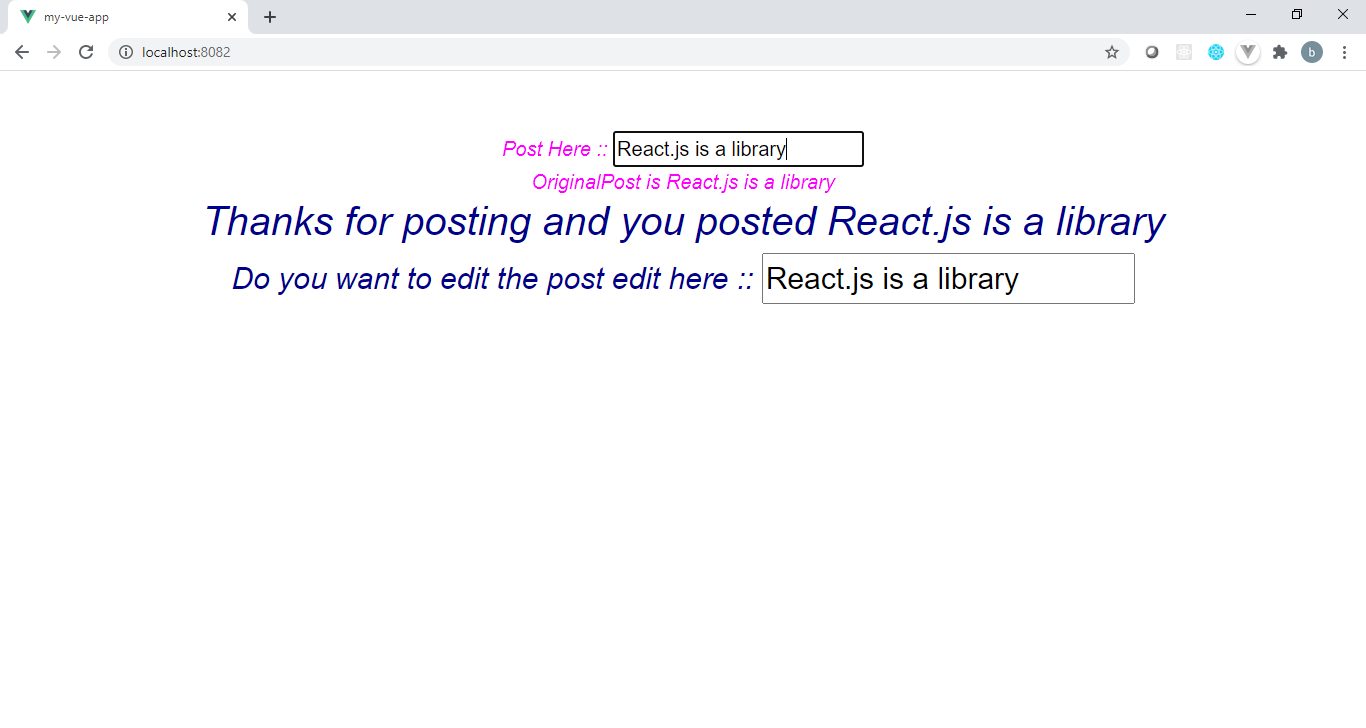
Enter the post here and pass it to BlogPost, if post is not null then BlogPost will get rendered and will generate below given view.



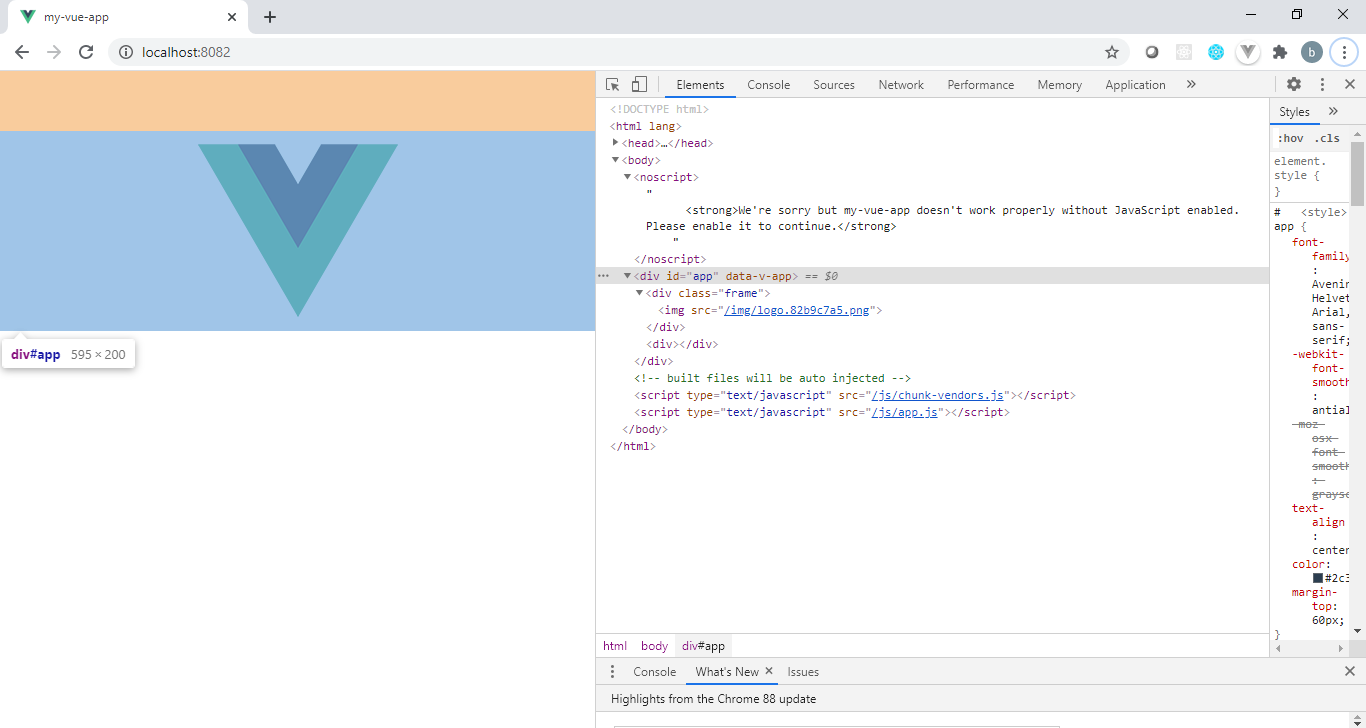
After editing the post in the above UI, everywhere the post value should be changed as given below



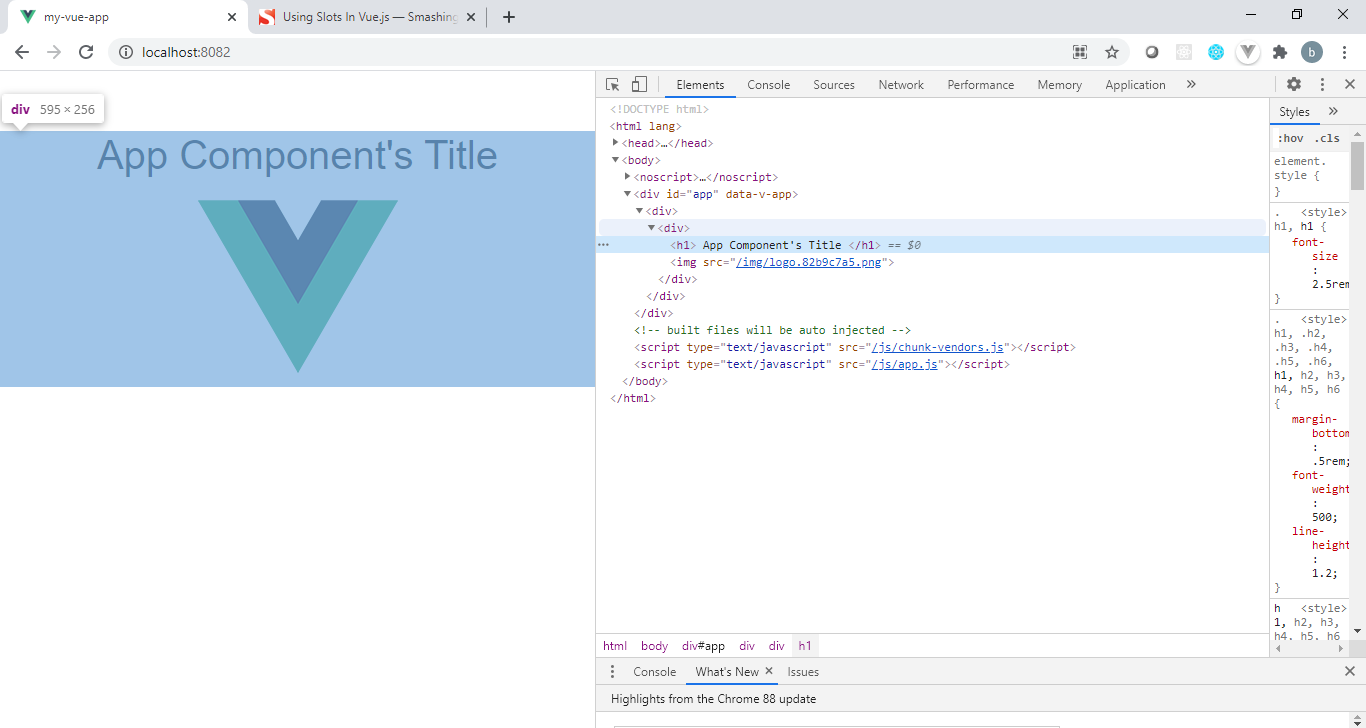
7.4 Consider problem statement 7.3, apply the styling to both components PostComponent and BlogPost. Apply different text-color, background-color, font-size, font style in each component. Make sure that Styling PostComponent should not impact on Styling of BlogPost. This problem statement should generate a view as given below.



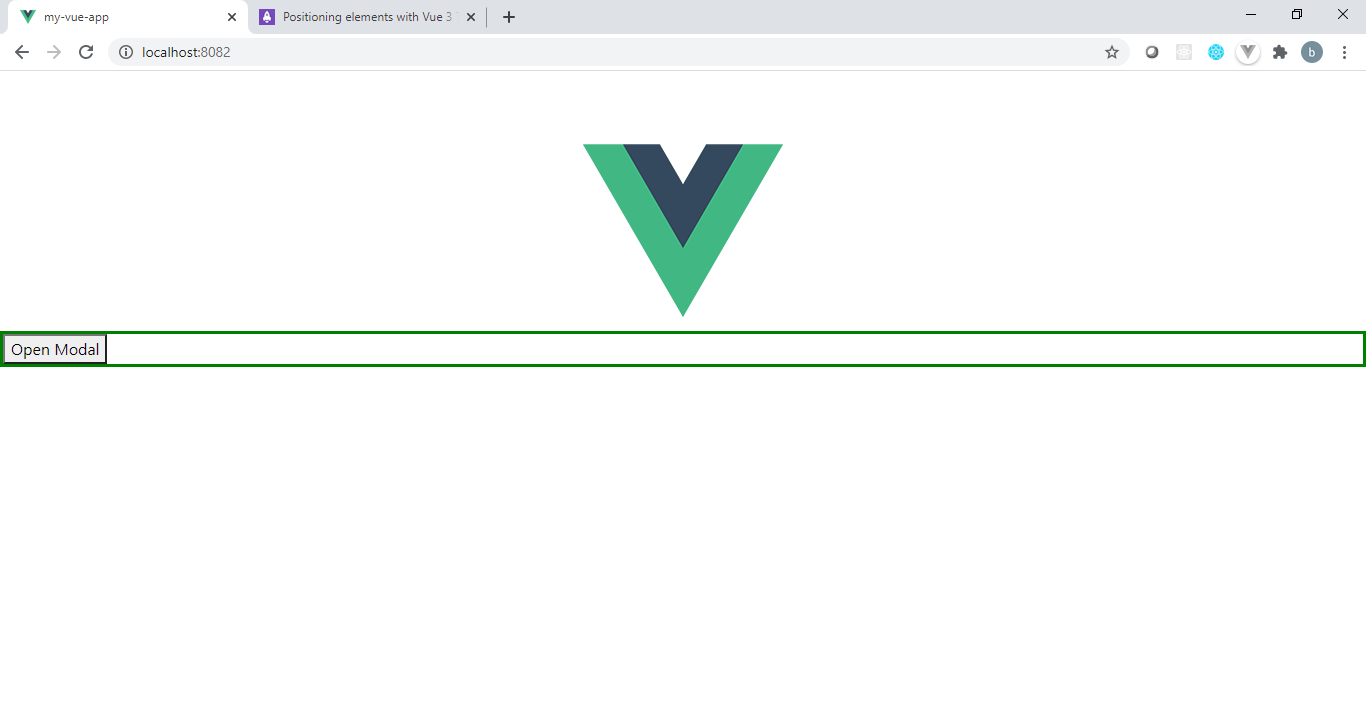
7.5 Create a FrameComponent that will render a empty division. App component will render FrameComponent by passing an <img> element as a children of it. Design FrameComponent’s template in such a manner that <img> children passed by App component while rendering FrameComponent will be placed in the division of FrameComponent. Check the below html structure when rendered App Component.



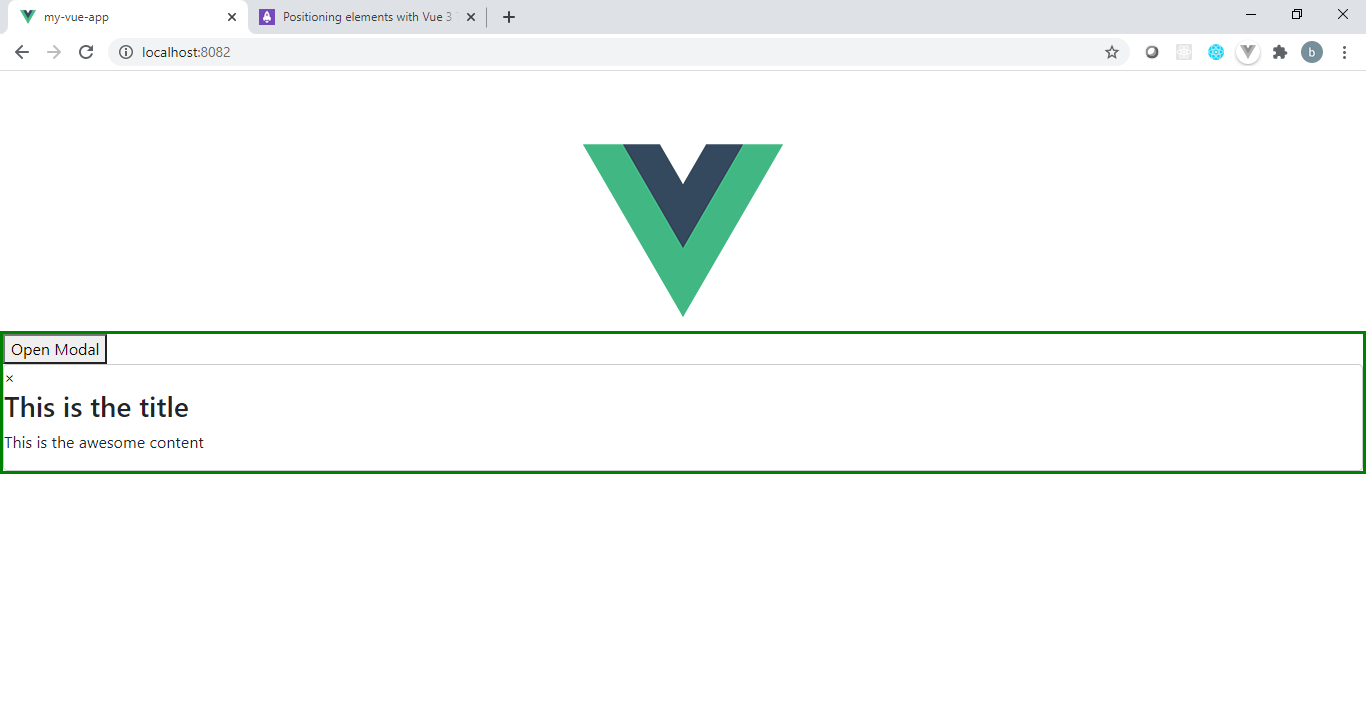
7.6 Create a component TitledFrame, that will render a <div><h1></h1></div>. App component from app.vue will render <TitledFrame> and it will pass a title and image to it. Title should be placed in <h1> of TitledFrame component and image should be placed in the division. App Component when rendered should generate below given view.



7.7 Create a Component MyComponent, that will render a button Open Modal, if data showModel from MyComponent will be false will generate a view as given below.



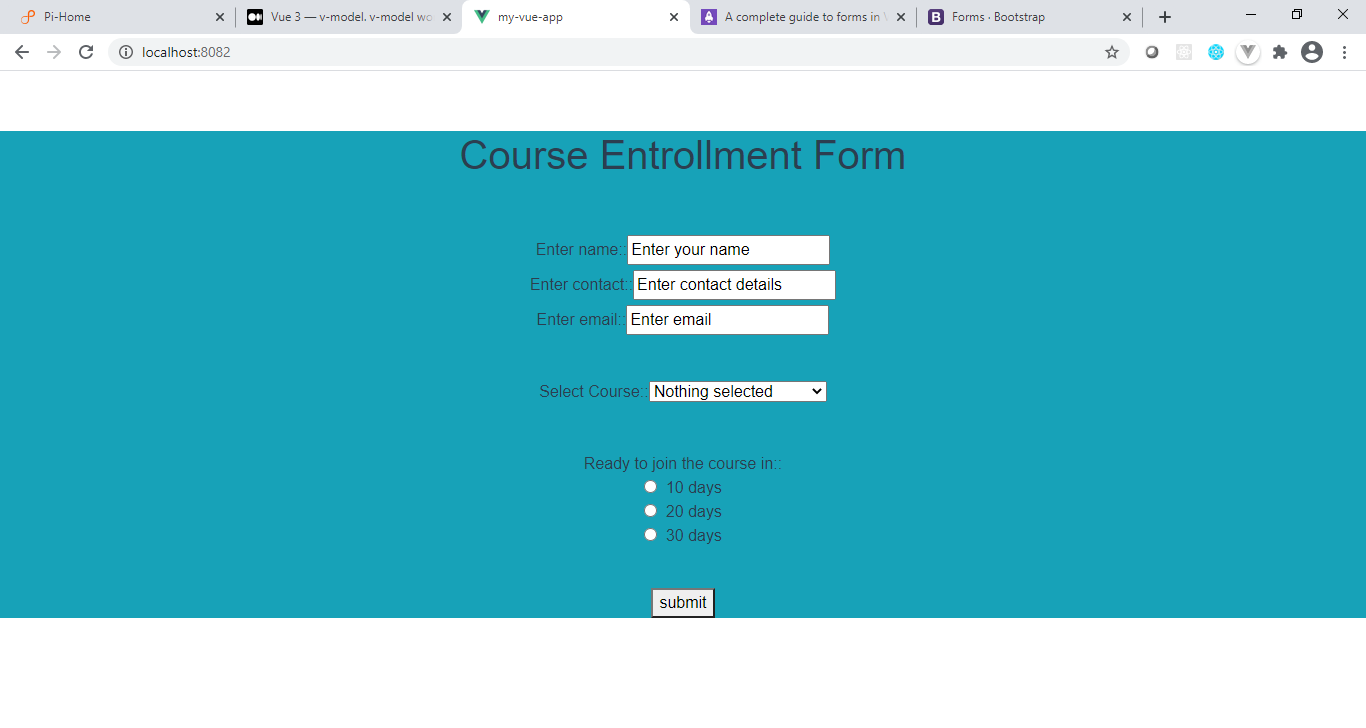
On click of button open model it should generate below view.



In index.html with <div id=”app”>, create one more division <div id=”myDiv”>, logo will rendered by App component in app container i.e. in division with id app. Make sure the open modal button and onClick of it whatever contents will get rendered should be placed in the division with id myDiv and this updates will happen when App component will render MyComponent. Execute the same using teleporting.

Module 8 : **Forms**

8.1 Create a course enrollment form as given below. List of course should be read only and displayed from FormComponent data.



On submit of above form it should display all entered values in alert box.

8.2 Apply validation to above form generated by FormComponent. It should display error message on respective fields if not entered the value or if pattern goes wrong for respective field.