System Description

The Referee Accountability Squad (RAS) responsive web app is a small app designed to provide information on current members, rule knowledge testing and have room for expansion into further resources.

For greater detail on the RAS including product summary, product purpose etc please see project proposal.

The system was built using HTML, CSS, Javascript and Vue.js. It incorporated responsive features such as scaling of elements to suit screen size (using meta HTML tag and CSS @media tag), user input binding, transitioning into view of modal elements, 'stickying' of the menu bar upon scrolling and providing feedback based on user input, to name a few.

Challenges

Vue CLI and single-file components

A significant attempt was made to work within the Vue CLI and build using single-file components, with some headway made. However, this was not commenced until a large portion of the project was already completed, and to transfer the project into the CLI and single-file form proved complex and challenging, and as such, this was abandoned. In general, this would be the preferred and more accepted method as this allows for greater modularity.

Member Page

This page was the most significant in terms of Vue usage, with props and a specific members component incorporated. The members component and prop usage was challenging, and a single-file template would have been ideal rather than one long string – as far as coding practices go, this is not efficient and not easily readable nor alterable. This was the point at which CLI and single-file components were considered but unfortunately given the completed nature of the majority of the project, it was deemed too late to adjust.

Deviations from Proposal

Features Adjusted

- Member Slideshow – this was altered from a slideshow to a view-all tile style approach, with individual member information visible upon hover. This was considered a more user-friendly approach to help find a specific member swiftly.

Features Removed/Pushed to next Phase:

- Search bar – Issues as discussed above with incorporating single-file components and working with the Vue CLI, meant that incorporating a search bar and potentially also working within a back-end was too complex for this project within the time frame.
- Social feeds – the research around this indicated largely working with pre-made Vue packages, which was not conducive to the task requirements of showing individual Vue capabilities, and there was also no established social feed that seemed relevant to link to.
- Registration/Login – these features would involve the usage of a back-end which was beyond the scope of this task. It seemed unnecessary to include the Vue elements when the overall outcome would be a non-event.

Coding Concept Requirements

Conditional Rendering & Event Handling

This was used in:

- the **rule-quiz** element
- showing/hiding of pop-up modals
- mobile navigation menu.

Having pop-up modals for member information, contact form submission and mobile navigation menu allows the screen space to be used efficiently, with only information that is relevant at the time being visible (e.g. the information of the member the mouse is hovering over). By using event handling and conditional rendering to determine when the pop-up modals were visible, Vue assisted in creating a visually clean and intuitive page for predictable information/navigation access.

The rule quiz element conditionally rendered simple text on the basis of an event – the user clicking a radio checkbox, which was linked to functions. Using Vue allows the user to receive instant feedback, which grows a connection between them and RAS – positive feedback if correct, a learning experience if not. The code snippets are shown below:

Vue

```
//rule quiz element. Checks response to rules quiz and displays message
new Vue({
  el: '#rule-quiz',
  data: {
    correct: false,
    incorrect: false
  },
  methods: {
    //if the true button is checked, the incorrect answer response is shown/
correct answer response hidden
    trueClicked: function() {
      this.correct = false;
      this.incorrect = true;
    },
    //if the false button is checked, the correct answer response is shown/
incorrect answer response hidden
    falseClicked: function() {
      this.correct = true;
      this.incorrect = false;
    }
  }
})
```

HTML

```
<!--Quiz question radio checkbox form with Vue answer check-->
      <div id="rule-quiz" class="quiz">
        <center><form name="quiz">
          A5 shoots the ball with his foot on the three-point line
          and the basket is successful. The score shall count for two points.
          <br><label for="true">True</label>
          <input type="radio” name="answer" value="true" v-
on:click="trueClicked">
```

```
          <br><label for="false">False</label>
          <input type="radio" name="answer" value="false" v-
on:click="falseClicked">
          <p v-if="correct">Your answer is correct!</p>
          <p v-else-if="incorrect">Your answer is wrong.</p>
        </form></center>
      </div>
```

Form Input Bindings & Enter/Leave Transitions

The contact page submission involved form input bindings, as well as a fade transition for appearance of the modal upon submission. By hiding the contact form upon submission and personalising the response, the site engages the user and ensures their experience feels individualised, as well as allowing them to check if the correct email was entered. Transitions (CSS available in .zip file) were also present in the member page modal (fade transition) and the mobile navigation menu (sliding transition):

Vue

```
//contact page element that responds to user input - hides completed form and
leaves thankyou message
var contactPage = new Vue({
  el: '#contact',
  data: {
    name: '',
    mail: '',
    context: '',
    complete: false,
  },
    methods: {
    processForm: function() {
      this.complete = true,
    }
  }
})
```

HTML

```
<div id="contact" class="row content">
              <p class="contact">Submit your name and email to be registered t
o our mailing list! You can also provide feedback here.</p>
              <form v-on:submit.prevent="processForm" v-if="complete == false
">
                <br><label for="name">Name: </label><input v-model="name" widt
h="50%" required>
                <br><label for="mail">Email: </label><input id="mail" v-
model="mail" placeholder="example@example.com" width ="50%" required>
                <br><label for="context">Tell us about yourself: </label>
                <br><textarea v-model="context" width="50%" rows="10"></
textarea>
                <br><input type="submit">
                <br>{{name}}
              </form>
              <transition name="fade">
```

```
                <div class="modal submitted" v-show="complete">
                  Hello {{name}}! Thanks for getting in touch. We will contact
you at {{email}} shortly to respond to your query.
                </div>
              </transition>
            </div>
```

Component and Prop Usage

Props were used in the member page, with values coded into the HTML as the component was re-used. It is logical to have a standardised structure, style and reactivity for each member being displayed – the use of components, with props that would allow easy addition of new members, the member page looks uniform, clean and is predictable in how each member icon will react to a mouseover. Following this methodology helps future developers working on the site easily integrate future members.

Vue

```
//Globally registered
//member template component that shows an information modal upon mouse hover
var memberPage = Vue.component('members', {
  props: ['name', 'association', 'image'],
  data: function() {
    return {
      info: false,
    }
  },
  methods: {
    mouseOver: function() {
      this.info = true;
    },
    mouseLeave: function() {
      this.info = false;
    }
  },
  template: '<div class="members"><img class="member" v-bind:src="image" v-
on:mouseover="mouseOver" v-on:mouseleave="mouseLeave"><transition name="fade">
<div class="modal" v-show="info"><img class="popup" v-
bind:src="image"><br>Name: {{name}}<br>Association: {{association}}<br>2020 Go
als: Officiate NBL1 Women<br>Greatest Achievement: 2020 U18 State Championship
s Gold Medal</div></transition></div>'
});
```

HTML (Singular Example)

```
<members name="Camellia Menzies" image="assets/images/Camellia-referee.jpg" as
sociation="Brisbane"></members>
```