COS30043
Interface Design and Development

SWINBURNE UNIVERSITY OF TECHNOLOGY

Lecture 9 – Single Page Applications

2022 – Semester 1

KNOWING

Single-page Application (SPA)

- a web application that is contained in a single web page to provide user experience similar to desktop applications
- all code, HTML, CSS and JavaScript, is retrieved on initial page load, and appropriate resources are dynamically loaded as necessary based on response to user actions
- the entire page does not reload nor control transfer to another page
- interaction involves dynamic communication with the web server in the background

Contents

- Using Router for Menu in SPA
- Login
- Using Pagination in SPA
- Data Insert, Update and Delete

SPA - Route

- **Creating VueRouter**

```
const router = VueRouter.createRouter({
  history: VueRouter.createWebHashHistory(),
  routes: [
    {
      // login route
      path: '/login',
      component: Login,
      name:"login"
    },
```

SPA - Route (Continued)

```
// logout route
{
    path: '/logout',
    name:"logout"
},
// dashboard
{
     path: '/dashboard',
     component: Dashboard,
     name:  'dashboard'
  }  ]
})
```

SPA – Menu

```
app.component('app-nav', {

template: `
<div>
    <v-spacer></v-spacer>
    <v-btn>
          <router-link  to="/login"
          v-on:click="logout()“ replace>
               Logout<v-icon>mdi-logout</v-icon>
          </router-link>
    </v-btn>
</div>
```

SPA – Menu (Continued)

```
`,
    methods: {
      logout() {
          this.$root.logout();
      }
    }
});
```

Contents

- Using Router for Menu in SPA
- Login
- Using Pagination in SPA
- Data Insert, Update and Delete

SPA – Login component

```
// Creating Login component
const Login = {
// defining variables to be used in the component
  data() {
      return {
           input: {
                username: "",
                  password: ""
                },
           valid: false,
... }
```

N.B. "..." indicates lines of code which are not shown.

SPA – Login component (Continued)

```
methods: {
    login() {
        if (this.$refs.form.validate()) {
           ...


            const requestOptions = {
              method: 'POST',
              headers: {
                  'Content-Type': 'application/json'
                        },
              body: JSON.stringify({
                  username: this.input.username,
                  password: this.input.password
                  })
             };
```

SPA – Login component (Continued)

```
// Login check
fetch("resources/api_user.php/", requestOptions)
.then( response =>{
    //turning the response into the usable data
    return response.json( );
})
.then( data =>{
  //This is the data you wanted to get from url
        if (data == null) {// didn't find this username password pair
                self.msg="username or password incorrect.";
        }
        else{
                this.$emit("authenticated", true);
                        //$emit() function allows you to pass
                        //custom events up the component tree.
                this.$router.replace({ name: "dashboard" });
        }
})
.catch(error => {
        self.msg = "Error: "+error;
});      ...
```

N.B. "..." indicates lines of code which are not shown here

SPA – Login component (Continued)

```
template: `
...
<v-form ref="form" v-model="valid" lazy-validation>
      <v-text-field v-model="input.username"
            :counter="10" :rules="usernameRules"
            label="Username" required>
      </v-text-field>
      <v-text-field v-model="input.password"
            label="Password" type="password"
            :rules="passwordRules" required>
      </v-text-field>
      <v-btn :disabled="!valid" color="success"
            class="mr-4" @click="login()"> Login
      </v-btn>
      <v-btn color="error" class="mr-4" @click="reset">
          Reset
      </v-btn>
</v-form>
```

Contents

- Using Router for Menu in SPA
- Using Vuetify Tabs in SPA
- Using Pagination in SPA
- Data Insert, Update and Delete

Pagination using Vue JS Paginate

- **Link *vuejs-paginate.js* file in the HTML**

```
<script src="js/vuejs-paginate.js"></script>
```

- **Register *VuejsPaginate* component**

```
components: {
                paginate: VuejsPaginateNext,
},
```

N.B.“...” indicates lines of code which are not shown here

Pagination in SPA

```
const ReadMysql = {
  template: `
      ...
   <v-card-text>
    <ul class="list-group">
     <li class="list-group-item"
            v-for="person in getItems"
            :key="person.name" >
        <img v-bind:src='person.fpath'
            class="img-rounded"
            alt="smiley" height="30px" />
        {{person.name}} {{person.age}}
     </li>
    </ul>
   </v-card-text>
      ...
```

N.B. "..." indicates lines of code which are not shown here

Pagination in SPA (Continued)

```
...
<paginate
     :page-count="getPageCount"
     :page-range="5"
     :margin-pages= "1"
     :click-handler="clickCallback"
     :prev-text=" 'Prev' "
     :next-text="'Next'"
     :container-class="'pagination'"
     :page-class="'page-item'">
</paginate>
...
```

N.B."..." indicates lines of code which are not shown here

Pagination in SPA (Continued)

// Defining functions in *method* and *computed*

```
computed:{
    getItems: function() {
        let current = this.currentPage * this.perPage;
        let start = current - this.perPage;
        return this.persons.slice(start, current);
     },
     getPageCount: function() {
      return Math.ceil(this.persons.length /
this.perPage);
     }
  },
  methods:{
      clickCallback: function(pageNum){
        this.currentPage = Number(pageNum);
      }
    },
```

Contents

- Using Router for Menu in SPA
- Using Vuetify Tabs in SPA
- Using Pagination in SPA
- Data Insert, Update and Delete

Data Insert

```
...
<v-form>
<v-text-field label="Name" v-model="name1" /></v-text-field>
<v-text-field label="Age" v-model="age1" /></v-text-field>

<v-radio-group label="Smiley Color" v-model="imgVar">
     <v-radio label="White" value="1"></v-radio>
     <v-radio label="Yellow" value="2"></v-radio>
</v-radio-group>

<v-btn depressed
       v-on:click="postData(name1,age1, imgVar)"
       color="primary">
       Add
</v-btn>

</v-form>
...
```

N.B. "..." indicates the lines of code which are not shown here

Data Insert (Continued)

```
methods: {
  postData: function(nm, age, img) {
      //define url for api
      var postSQLApiURL = 'resources/apis.php/'

      var self = this;
      // POST request using fetch with error handling
      const requestOptions = {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json'
        },
        body: JSON.stringify({
          name: nm,
          age: age,
          fpath: 'img/smiley' + img + '.png'
        })
      };
```

Data Insert (Continued)

```
fetch(postSQLApiURL, requestOptions)
.then( response =>{
  //turning the response into the usable data
  return response.json( );
})
.then( data =>{
  //This is the data you wanted to get from url
   self.msg = "Data Inserted Successfully."  ;
})
.catch(error => {
   self.msg = 'There was an error!' + error;
});
```

Data Update

```
<v-form name="myForm2" class="form-horizontal">

    <v-text-field label="Name" v-model="name2" />
    </v-text-field>

    <v-text-field label="Age" v-model="age2" />
    </v-text-field>

    <v-btn depressed v-on:click="putData(name2,age2)"
       color="primary">
       Update
    </v-btn>

</v-form>

...
```

N.B. "..." indicates the lines of code which are not shown here

Data Update (Continued)

```
methods: {
   putData: function(nm, age) {
      //define url for api
      var putSQLApiURL ='resources/apis.php/name/'+nm;
      var self = this;
      // POST request using fetch with error handling
      const requestOptions = {
          method: 'PUT',
          headers: {
            'Content-Type': 'application/json'
          },
          body: JSON.stringify({
            name: nm,
            age: age
          })
    };
```

Data Update (Continued)

```
fetch(putSQLApiURL, requestOptions)
.then( response =>{
  //turning the response into the usable data
  return response.json( );
})
.then( data =>{
  //This is the data you wanted to get from url
  self.msg="successful";
})
.catch(error => {
  self.err=error
});
```

Data Delete

```
...
<v-form>

<v-text-field label="Name" v-model="name3" />
</v-text-field>

<v-btn depressed v-on:click="delData(name3)"
     color="primary">
     Delete
</v-btn>

</v-form>
...
```

N.B.“...” indicates lines of code which are not shown here

Data Delete (Continued)

```
...
methods: {

  delData: function(nm, age) {
      var delSQLApiURL ='resources/apis.php/name/' + nm;
      var self = this;
      // DELETE request using fetch with error handling
      const requestOptions = {
        method: 'DELETE',
        headers: {
          'Content-Type': 'application/json'
        },
        body: JSON.stringify({
          name: nm
        })
      };
```

N.B.“...” indicates lines of code which are not shown here

Data Delete (Continued)

```
fetch(delSQLApiURL, requestOptions)
.then( response =>{
      //turning the response into the usable data
      return response.json( );
})
.then( data =>{
  //This is the data you wanted to get from url
   self.msg = "Data deleted Successfully"

})
.catch(error => {
      self.msg = 'There was an error!';
      self.statusText = error;
});
```