1. Walk everyone through the project. Be sure to call out:

   A. This is the exact same starter code you will receive for your capstone

   B. Database has been previously installed using scripts

   C. The Java project is the same code we built during the M2 web services review (we can test with postman to validate it works)

   D. We will cover login/register and web service calls on 2nd Review Day

2. Walk through the homes project that we want to achieve. Load the final solution and run. Be sure to point out that the backend server needs to be started first. BUT, we won't be using it today. We will hook that up on Thursday. We will just grab some JSON to temporarily put inside our component.

3. Walk through the starting code. Point out that I have added images to the assets folder since we don't have a place to store them. Also show them that the app.vue file has a starter nav bar but we aren't going to use that one. We will build ours later inside the header. Be sure to show them the <router-view> tag and the route configurations inside the router. We will cover these later.

4. So let's start by building a header and footer and add them to the app.vue file:

   A. Let's create a header: **TheHeader.vue**

```
<template>
  <div>
     <p>{{pageTitle}}</p>
  </div>
</template>

<script>
export default {
   name: 'the-header',
   data() {
      return {
         pageTitle: 'Java Green Home Search'
      };
   }
```

```
}
</script>

<style>

</style>
```

B. Let's create a footer: **TheFooter.vue**

```
<template>
  <div>
     <p>&#169; {{ copyText }}</p>
  </div>
</template>

<script>
export default {
   name: 'the-footer',
   data() {
      return {
         copyText: 'Java Green Enterprises LLC.'
      };
   }
}
</script>

<style>

</style>
```

C: Ask the students how to include them so that they appear on all pages.

```
App.vue

<template>
```

```
  <div id="app">
   <the-header></the-header>
   <router-view />
   <the-footer></the-footer>
  </div>
</template>

<script>

import TheHeader from './components/TheHeader'
import TheFooter from './components/TheFooter'

export default {


  name: 'App',
  components: {
     TheHeader,
     The Footer
  }
}
</script>
```

5. Let’s add an image to the header:

```
 Let’s add an image to the header…

<img src="../assets/java-green.png"/>


img {
   width: 15%;
   height: auto;
  display: block;
  margin-left: auto;
  margin-right: auto;

}
```

6. Let’s add a Nav bar to the header:

```
    <nav>
      <a href="#">Home</a>
      <a href="#">About</a>
      <a href="#">Search Homes</a>
   </nav>
```

7. Let’s add some CSS styling (add one at a time)

```
nav {
    display: flex;
    justify-content: center;
}
nav a {
    padding: 15px 25px;
    background-color: #16b065;
    Color: black;
}
nav a:hover {
    background-color: #56e38b;
    color:white;
}
```

8. Now let's go work on the **HomeSearch.vue** component

```
<template>
  <div>
    <p> Placeholder text </p>
  </div>
</template>

<script>
export default {
   name: 'home-search',
   data() {
      return {};
   }
}
</script>

<style scoped>

</style>
```

9. Now let's add this to a View called **SearchForZip.vue**

```
<template>
   <home-search />
</template>

<script>

import HomeSearch from '../components/HomeSearch.vue';

export default {
  components: {
     HomeSearch
  },

}
</script>

<style scoped>

</style>
```

10. Now let’s set up a route so we can add it to our Nav Bar. We need to add the import and the router entry

```
import SearchByZip from '../views/SearchByZip.vue'

  {
    path: "/search",
    name: "search",
    component: SearchByZip,
    meta: {
      requiresAuth: false
    }
  },
```

11. Now we need to add the route to the navigation in the header

```
<router-link v-bind:to="{ name: 'search' }">Home Search</router-link>
```

12. Let’s go ahead and create a route back to the home page while we are here.

```
<router-link v-bind:to="{ name: 'home' }">Home</router-link>
```

13. Let's go back to the **HomeSearch** Component and start adding some functionality. We will start by adding in the table that will hold our home information. (Show the students the table builder I used: Go to: https://divtable.com/table-styler/

```
<div id="main-div">
  <div class="divTable minimalistBlack">
    <div class="divTableHeading">
      <div class="divTableRow">
        <div class="divTableHead">A</div>
        <div class="divTableHead">B</div>
        <div class="divTableHead">C</div>
        <div class="divTableHead">D</div>
        <div class="divTableHead">E</div>
      </div>
    </div>
    <div class="divTableBody">
      <div class="divTableRow">
        <div class="divTableCell">Data A</div>
        <div class="divTableCell">Data B</div>
        <div class="divTableCell">Data C </div>
        <div class="divTableCell">Data D</div>
        <div class="divTableCell">Data E</div>
      </div>
    </div>
  </div>
</div>
```

14. Now let's add the matching CSS:

```
div.minimalistBlack {
   margin:auto;
  border: 2px solid #06B712;
  width: 80%;
  text-align: left;
  border-collapse: collapse;
}
.divTable.minimalistBlack .divTableCell, .divTable.minimalistBlack .divTableHead {
  border: 1px solid #000000;
  padding: 5px 4px;
}
```

```
.divTable.minimalistBlack .divTableBody .divTableCell {
  font-size: 14px;
}
.divTable.minimalistBlack .divTableHeading {
  background: #1DFF2C;
  background: -moz-linear-gradient(top, #55ff61 0%, #33ff41 66%, #1DFF2C 100%);
  background: -webkit-linear-gradient(top, #55ff61 0%, #33ff41 66%, #1DFF2C 100%);
  background: linear-gradient(to bottom, #55ff61 0%, #33ff41 66%, #1DFF2C 100%);
  border-bottom: 3px solid #0F9A39;
}
.divTable.minimalistBlack .divTableHeading .divTableHead {
  font-size: 15px;
  font-weight: bold;
  color: #109902;
  text-align: left;
}
.minimalistBlack .tableFootStyle {
  font-size: 14px;
}
/* DivTable.com */
.divTable{
    display: table;
    table-layout: fixed;
}
.divTableRow { display: table-row; }
.divTableHeading { display: table-header-group;}
.divTableCell, .divTableHead { display: table-cell;}
.divTableHeading { display: table-header-group;}
.divTableFoot { display: table-footer-group;}
.divTableBody { display: table-row-group;}
```

The table should be somewhat styled. If we want to fix the hover color on the main menu, how would we do it?

15. Now let's add the search box:

```
<div id="searchHomes">
    <label for="zip">Enter Your Zip Code To Find Your Next Dream Home:</label>
    <input type="text" name="zip"/>
</div>
```

16. OK, components are placed, but it looks kind of jammed together. Let's add some CSS fixes:

```
#main-div {
    margin: 30px;
}

#search {
   margin: 30px;
}
```

17. That fixed the spacing around the table, now let's clean up the search bar. I HATE the default text boxes.

```
input[type=text] {
  margin: 30px;
  width: 15%;
  padding: 12px 20px;
  box-sizing: border-box;
  border: 2px solid green;
  border-radius: 6px;
}
```

18. Now that the page looks a little better, let's go set up the sample data. In our next review session we will get this data from the web service. However, I will get this test data from the actual web service by running it in Postman. (Go to Postman)

    Then add the data to the state property in the Vuex store just below the auth token stuff.

```
[
  {
    "mlsNumber": "1000",
    "imageName": "1000.jpg",
    "address": {
```

```
        "addressLine": "123 Elm Street",
        "city": "Columbus",
        "state": "Ohio",
        "zipCode": 43323
    },
    "numberOfBedrooms": 3.0,
    "numberOfBathrooms": 4.0,
    "price": 250000.00,
    "description": "Freddies Nightmare will come to your life in the classic home. Sweet dreams!"
},
{
    "mlsNumber": "1001",
    "imageName": "1001.jpg",
    "address": {
        "addressLine": "125 Elm Street",
        "city": "Columbus",
        "state": "Ohio",
        "zipCode": 43323
    },
    "numberOfBedrooms": 3.0,
    "numberOfBathrooms": 3.0,
    "price": 100000.00,
    "description": "Nice house but the neighbor is a little strange."
},
{
    "mlsNumber": "1002",
    "imageName": "1002.jpg",
    "address": {
        "addressLine": "777 Oak Street",
        "city": "Dublin",
        "state": "Ohio",
        "zipCode": 43017
    },
    "numberOfBedrooms": 5.0,
    "numberOfBathrooms": 3.0,
    "price": 350000.00,
    "description": "Charming house far away from those Elm street weirdos."
},
{
    "mlsNumber": "1003",
```

```
    "imageName": "1003.jpg",
    "address": {
      "addressLine": "555 E Main Street",
      "city": "Columbus",
      "state": "Ohio",
      "zipCode": 43203
    },
    "numberOfBedrooms": 25.0,
    "numberOfBathrooms": 10.0,
    "price": 450000.00,
    "description": "This house was featured on the reality tv show: Kevin Flipped My House."
  },
  {
    "mlsNumber": "1004",
    "imageName": "1004.jpg",
    "address": {
      "addressLine": "127 Slider Lane",
      "city": "Columbus",
      "state": "Ohio",
      "zipCode": 43210
    },
    "numberOfBedrooms": 25.0,
    "numberOfBathrooms": 10.0,
    "price": 150000.00,
    "description": "Next door to the White Castle, always be on the lookout for Harold and Kumar!"
  },
  {
    "mlsNumber": "1005",
    "imageName": "1005.jpg",
    "address": {
      "addressLine": "127 Crime Fighters Blvd",
      "city": "Columbus",
      "state": "Ohio",
      "zipCode": 43210
    },
    "numberOfBedrooms": 25.0,
    "numberOfBathrooms": 10.0,
    "price": 750000.00,
    "description": "Rumour has it batman used to live here before upgrading to the bat cave!"
  }
```

```
]
```

19. Now let’s go back to our HomeSearch component and add a data property to hold the zip code that the user will enter:

```
data() {
  return {
     zipFilter: ‘’
  };
},
```

20. Now let’s add a v-model to bind zipFilter to the input box

```
<input name="zip" type="text" v-model="zipFilter"/>
```

21. Now we need to write a computed property to get the data from the Vuex store and filter it based on the zipFilter value

```
computed: {
  filteredHomes() {
    const homes = this.$store.state.homes;
    return homes.filter(home => {
      return this.zipFilter ==’’ ? true : this.zipFilter == home.addres.zipCode;
    });
  }
}
```

22. Now that we have a computed property ready, we need to modify our template to loop through and update the row data in our table.

```
<div class="divTableRow" v-for="home in filteredHomes" v-bind:key="home.mlsNumber">
```

23. Now that we have the computed properties in place… Let's go bind the data to the Table

24. Now let's add the image

```
<img v-bind:src=home.imageName />
```

But this gives us a broken link for the image, so we have to go back to the store and fix the image links:

imageName: require('../assets/1000.jpg')
(file paths need to be wrapped in a require function)

25. Fix and CSS

```
img {
    width: 150px;
    height: auto;
}
```

26. What if we wanted to NOT show the empty table if there are no results? Ask the students what could we try?

    Well, we shouldn't add a v-if to the v-for because the v-for will have a higher priority, so it will render an empty table… What we can do, is add it to the parent div and test for filteredHomes.length.

    <div class="divTable minimalistBlack" v-if="filteredHomes.length > 0">
27.
28. Now the table should disappear….