

Exam 4 - Part 2: The Plant Store

This is a closed-notes exam. You may only use the notes and files you uploaded to GitHub at the end of Day 1. No talking, no looking at other projects or searching online.

- ✅ This is your chance to show how well you can apply what you've practiced.
- ✅ The project must be completed and pushed to GitHub by the end of class.

Project Overview:

You are building a Plant Store with:

- A Home page that welcomes users
- A Plants page that displays a list of plants
- A Nav bar that lets users navigate between pages
- A Dropdown filter to show only indoor or outdoor plants
- A list of plant cards built using props
- An event emit from the Nav to App.vue that filters the plants
- Two simple Vue Router routes (/ and /plants)
- No dynamic params, no API calls
- Bootstrap is optional

Step-by-Step Instructions:

1. Create a new Vue project

Use: `npm create vue@latest`

Choose: Router: YES, Pinia: NO, TypeScript: NO, Testing: NO

File Structure

```
src/  
  components/  
    Nav.vue  
    PlantCard.vue  
  views/  
    Home.vue  
    PlantList.vue  
App.vue  
main.js  
router/  
  index.js
```

Step 2: Clean up App.vue and router/index.js

1. In `App.vue`, replace the template with a basic layout and remove the `<script>` markup. Your completed `App.vue` should look like this:

```
src > App.vue > ...
1  <template>
2  |   <h1>Plant Store App</h1>
3  |   <router-view />
4  </template>
```

2. Then open `src/router/index.js`
 - temporarily set up an empty route list
 - remove `HomeView` import
 - `index.js` should now look like this

```
src > router > JS index.js > ...
1  import { createRouter, createWebHistory } from 'vue-router'
2
3  const router = createRouter({
4    history: createWebHistory(import.meta.env.BASE_URL),
5    routes: [],
6  })
7
8  export default router
```

3. **(optional)** Remove the `import './assets/main.css` from `main.js` if you don't wish to have the default styling

```
src > JS main.js > ...
1  // import './assets/main.css'
2
3  import { createApp } from 'vue'
4  import App from './App.vue'
5  import router from './router'
6
7  const app = createApp(App)
8
9  app.use(router)
10
11 app.mount('#app')
```

✅ **Test:** Your app should compile and display “Plant Store App” in the browser — no errors, no missing imports.

Step 3: Create Home.vue and add it to the router

1. Create `src/views/Home.vue`:

```
▼ Home.vue ×
src > views > ▼ Home.vue > ...
1  <template>
2  |   <h2>Welcome to the Plant Store!</h2>
3  |   </template>
4
```

2. Open `router/index.js`
 - **add** an import for the `Home.vue`
 - **add** the route:

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

const routes = [
  { path: '/', component: Home }
]
```

✅ **Test:** Visiting `/` should now display

Plant Store App

Welcome to the Plant Store!

Step 4: Create `PlantList.vue` and add it to the router

1. Create (or open if you already created it) `src/views/PlantList.vue`:
 - add a template:

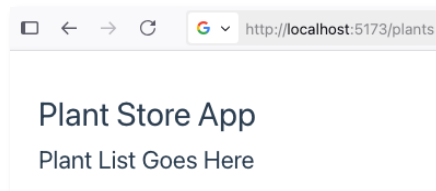
```
PlantList.vue X
src > views > PlantList.vue > ...
1 <template>
2 <h2>Plant List goes here</h2>
3 </template>
```

2. In `router/index.js`, add:

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

const routes = [
  { path: '/', component: Home },
  { path: '/plants', component: PlantList }
]
```

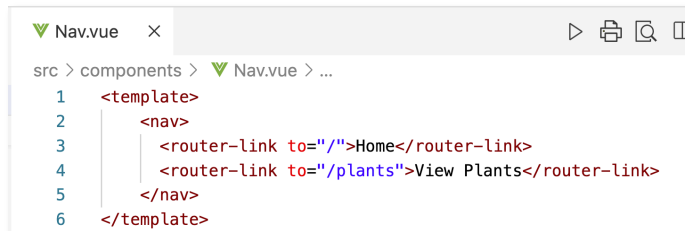
✅ **Test:** Visiting `/plants` should show the heading from `PlantList.vue`.



Font formatting varies based on use of `main.css` in Step 2.3.

Step 5: Create a Navigation Component that Utilizes Routes

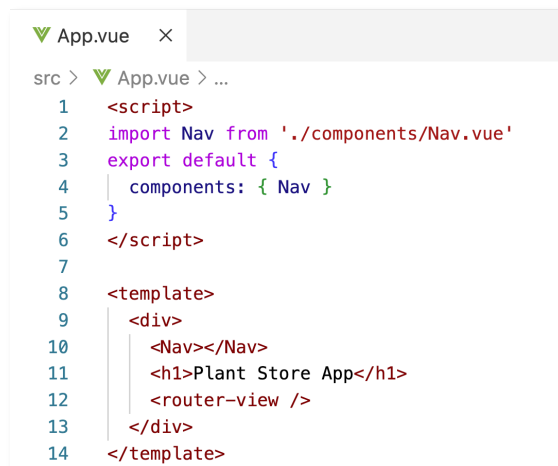
1. Create (or open if you already created it) `Nav.vue` and add a template:



```
1 <template>
2   <nav>
3     <router-link to="/">Home</router-link>
4     <router-link to="/plants">View Plants</router-link>
5   </nav>
6 </template>
```

2. Update `App.vue`:

- Create a `<script>` block
 1. Import the `Nav.vue`
 2. Register the `Nav` component
- Update the template with the `Nav` markup



```
1 <script>
2   import Nav from './components/Nav.vue'
3   export default {
4     components: { Nav }
5   }
6 </script>
7
8 <template>
9   <div>
10    <Nav></Nav>
11    <h1>Plant Store App</h1>
12    <router-view />
13  </div>
14 </template>
```

✅ **Test:** You should be able to click back and forth between Home and Plants views.

Step 6: Add Plant Data in `App.vue`

1. **Create a hardcoded plants array in `App.vue`**
 - In `App.vue` add the `data` method that returns an array of plant objects:

```

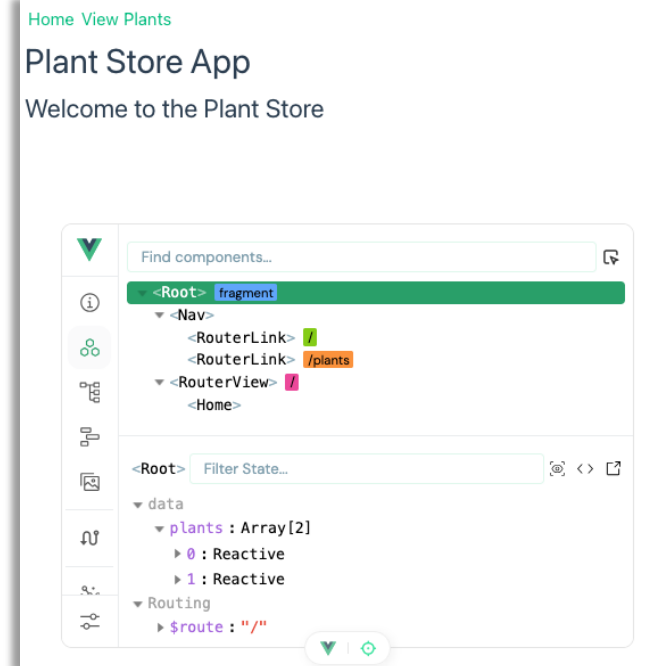
data() {
  return {
    plants: [
      { id: 1, name: "Snake Plant", type: "Indoor", inStock: true },
      { id: 2, name: "Rose", type: "Outdoor", inStock: false }
    ]
  }
}

```

2. Pass the array into the view:

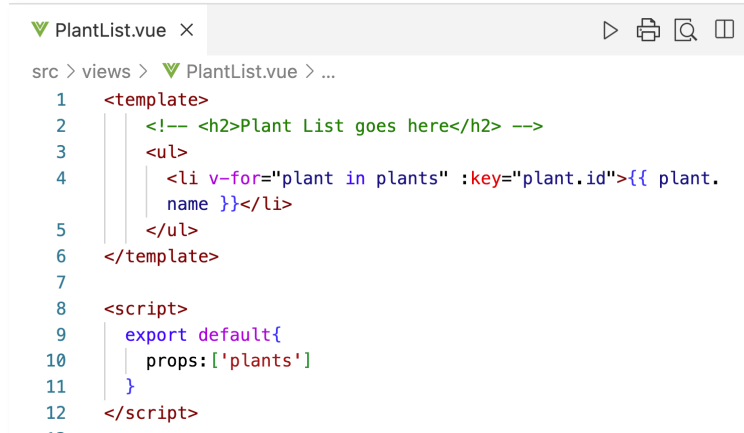
```
<router-view :plants="plants" />
```

✅ **Test:** App runs without errors.
Check in the Vue Dev tools for the data.



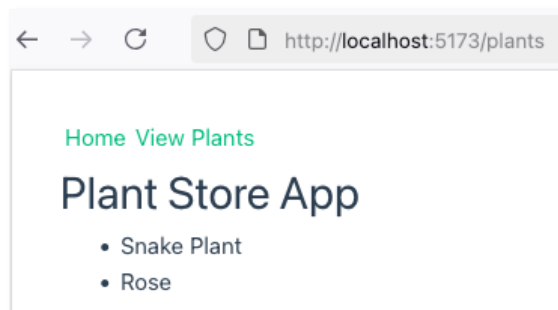
Step 7: Receive and display plant names in PlantList.vue

1. Add `props: ['plants']`
2. Use `v-for` to display names:



```
PlantList.vue X
src > views > PlantList.vue > ...
1 <template>
2   <!-- <h2>Plant List goes here</h2> -->
3   <ul>
4     <li v-for="plant in plants" :key="plant.id">{{ plant.
      name }}</li>
5   </ul>
6 </template>
7
8 <script>
9   export default{
10     props: ['plants']
11   }
12 </script>
```

✓ **Test:** Visiting `/plants` should show plant names.



Step 8: Create and implement PlantCard.vue

1. Create (or open if you already created it) `PlantCard.vue`:
 - Add the `<script>` block with the default object
 - Add the template

```
PlantCard.vue X
src > components > PlantCard.vue > ...
1  <script>
2  export default {
3    props: ['plant']
4  }
5  </script>
6
7  <template>
8    <div>
9      <h3>{{ plant.name }}</h3>
10     <p>Type: {{ plant.type }}</p>
11     <p v-if="plant.inStock">In Stock</p>
12     <p v-else>Out of Stock</p>
13   </div>
14 </template>
```

2. Update `PlantList.vue`
 - Remove the ``
 - Add the `<PlantCard>` markup and send the plant prop:

```
<template>
  <div>
    <PlantCard v-for="plant in plants" :key="plant.id" :plant="plant" />
  </div>
</template>
```

- Import and register the `PlantCard`

```
<script>
import PlantCard from "../components/PlantCard.vue"
export default{
  props:['plants'],
  components:{PlantCard}
}
</script>
```



✓ Test: Plant info should now show using cards.

Step 9: Add dropdown filter to Nav.vue and emitter

1. Open `Nav.vue`
 - Update the template to include the `<select>` drop down markup :

```
vue-project > src > components > Nav.vue
1  <template>
2    <nav>
3      <router-link to="/">Home</router-link>
4      <router-link to="/plants">View Plants</router-link>
5      <!-- Add the Select Drop down -->
6      <select @change="$emit('filter-plants', $event.target.value)">
7        <option value="All">All</option>
8        <option value="Indoor">Indoor</option>
9        <option value="Outdoor">Outdoor</option>
10     </select>
11   </nav>
12 </template>
```


2. Open App.vue:

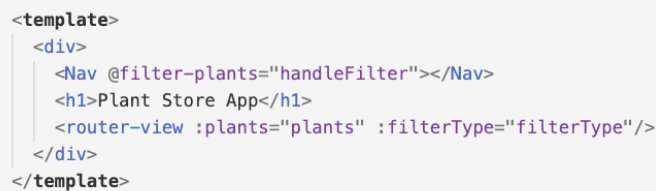
- add a `filterType` property inside the `data()` section and set its initial value to "All".
- Add a `methods` property, and define a method inside it called `handleFilter()`.

✅ This allows your app to react to the user's filter selection from the dropdown.



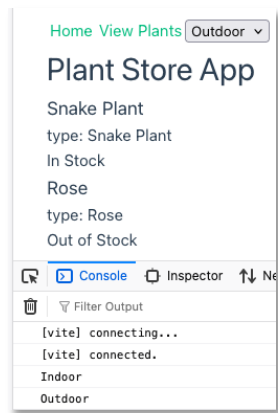
```
App.vue
vue-project > src > App.vue
1 <script>
2 import Nav from './components/Nav.vue'
3 export default {
4   components: { Nav },
5   data() {
6     return {
7       plants: [
8         { id: 1, name: "Snake Plant", type: "Indoor", inStock: true },
9         { id: 2, name: "Rose", type: "Outdoor", inStock: false }
10      ],
11      // Add filterType
12      filterType: 'All'
13    }
14  },
15  // add methods property and function
16  methods: {
17    handleFilter(type) {
18      this.filterType = type;
19      console.log(this.filterType);
20    }
21  }
22 }
23 </script>
```

- Update `<nav>` in the template to include the event listener (`@filter-plants`)
- Update `<router-view>` in the template with the attribute `“filterType”`



```
<template>
  <div>
    <Nav @filter-plants="handleFilter"></Nav>
    <h1>Plant Store App</h1>
    <router-view :plants="plants" :filterType="filterType"/>
  </div>
</template>
```

✓ **Test:** Changing the dropdown updates the filterType and displays either “Indoor” or “Outdoor” in the console.



Step 10: Filter plants in PlantList.vue

1. Add `filterType` to `props`: `['plants', 'filterType']`
2. Create (if not done already) a `<div>` inside your `<template>` to act as a wrapper for each plant card.
3. Move the `v-for="plant in plants"` and `:key="plant.id"` directives to that `<div>`.
4. Inside that `<div>`, place the `<PlantCard>` component.
5. Pass the plant object to `<PlantCard>` using `:plant="plant"`.
6. Add a `v-if` to `<PlantCard>` using the condition:
`v-if="filterType === 'All' || plant.type === filterType"`

- The full list of plants is always available, but Vue only renders the plant cards that match the selected filter.
- If "All" is selected, Vue renders every plant card.
- If "Indoor" or "Outdoor" is selected, Vue renders only the cards where the plant's type matches the selected filter.

```
PlantList.vue x
vue-project > src > views > PlantList.vue
1  <template>
2    <div v-for="plant in plants" :key="plant.id">
3      <PlantCard v-if="filterType === 'All' || plant.type === filterType"
4        :plant="plant" />
5    </div>
6  </template>
7  <script>
8    import PlantCard from "../components/PlantCard.vue"
9    export default{
10     props:['plants', 'filterType'],
11     components:{PlantCard}
12   }
13 </script>
```

✅ **Test:** Changing the dropdown filters which plants are displayed on the page

