Vue JS Lab Guide

1. The time allotted for solving the exercises is 7.5 hours (1.25 hours per day, for 6 days).
2. It is expected that students will take more time than this (typically 30 minutes extra per day).
3. A Workshop Application shall be built by students, in the class, led by the instructor.
4. There is also a Meetings Application that will be built by students – This will take roughly 16 hours of effort.

Exercises on Vue JS, Vue Router & Vuex

1. Create a Vue root instance that manages a login form. It should have as children the following elements
a. A form with id 'login-form', and action set to '/login'
b. An input element to take username
c. An input element to take password
d. A submit button
You may add additional elements as children if you wish to. You may also enclose the inputs within a div etc. for styling purpose (these are optional).

Define the HTML first as a DOM template, and then using a string template, and render it in each case.

2. Define 2 root instances for 2 forms - one for login and the other for sign up.
- the login form is like in the previous exercise
- the signup form asks user to choose username, password and has an additional input that confirms password.
You need to render each in a separate root element.

3. Define components called LoginForm and SignupForm. They render UI as in the previous exercise. Render them on the page using elements of the component types. Now create a single component capable of rendering both login and signup forms – it takes a prop hasConfirmPassword and shows the UI accordingly.

4. Understand event flow in JavaScript - https://javascript.info/bubbling-and-capturing. Find out how to use "stop" event modifier in Vue - https://vuejs.org/v2/guide/events.html#Event-Modifier

5. Given the following product information, display it using a Vue JS instance (data should define product as a property with this as its value). The colors should be displayed in a list.

```
{
   name: 'Apple iPhone 12',
   price: 900,
   colors: [
      'Space Gray', 'White', 'Red'
   ]
}
```

6. Given the following products information, display it using a Vue JS instance. If colors is unavailable a message "The information about available colors is missing" should be displayed. The name of the phone should be set up as the title attribute value for the element displaying an item's information.

```
[
   {
      name: 'Apple iPhone 12',
```

```
        price: 900,
        colors: [
            'Space Gray', 'White', 'Red'
        ]
    },
    {
        name: 'Apple iPhone 11',
        price: 780
    },
    {
        name: 'Apple iPhone X',
        price: 650,
        colors: [
            'Space Gray', 'Yellow', 'Red'
        ]
    }
];
```

7. Sign up at https://openweathermap.org/ (Sign up), and view the API key at https://home.openweathermap.org/api_keys. You will need to use the current weather API - https://openweathermap.org/current (this is free with usage limits).

Define a component called WeatherReport that shows details of Wind, Cloudiness, Pressure, Humiditiy, Sunrise, Sunset and Geographical coordinates for a given city. Get some details for your city from https://openweathermap.org/city. These details are to be passed as props. You can also find details of weather by going to this URL - substitute city name with your city's name. Also substitute your api key.

```
api.openweathermap.org/data/2.5/weather?q={city name}&appid={your api key}
```

The weather data also contains a description of the weather and the name of the icon representing the weather condition - use this to display an image of the weather condition within the component. You can find details of how to get the URL of the icon (i.e. image) from the incoming data, here https://openweathermap.org/weather-conditions

Also define a component called SunriseSunset that shows Sunrise and Sunset timings. Use this within WeatherReport to show the details of sunrise and sunset.

__NOTE__:
a. The API key may take a few hours to generate. You will not require the API till you start working with the backend to fetch weather data and use it (will be done in forthcoming exercises). For a guide on using the API check https://openweathermap.org/guide

b. If you are able to get the API key, you can try making an Ajax call using Axios to fetch the current weather data for your city. Once the call is successful, you can render the WeatherReport element within the root (passing it the data you obtained using the API call as props).

8. Add styles to the WeatherReport component by showing the details inside a box with background color dictated by weather condition codes (group codes) - https://openweathermap.org/weather-conditions.
For example,
- If the weather is clear, show the details in lightblue background
- If the weather is rainy/drizzle - use dark blue
Similarly choose some colors for the other weather conditions.

9. Create an CitiesWeatherReport component that let's one view weather details for a group of cities. It takes as input (props) the weather details for a group of cities. It should display the weather details for each city. When doing so, use the WeatherReport component created in a previous exercise. Also make sure to give a unique key prop value for each WeatherReport element rendered.

```
const cities = [
  {
    "name": "New York",
    "weather": [
      {
        "id": 800,
        "main": "Clear",
        "description": "clear sky",
        "icon": "01n"
      }
    ],
    "main": {
      "temp": 281.52,
      "feels_like": 278.99,
      "temp_min": 280.15,
      "temp_max": 283.71,
      "pressure": 1016,
      "humidity": 93
    },
    ...
  },
  {
    "name": "New York",
    "weather": [
      {
        "id": 800,
        "main": "Clear",
        "description": "clear sky",
```

```
            "icon": "01n"
          }
        ],
        "main": {
          "temp": 281.52,
          "feels_like": 278.99,
          "temp_min": 280.15,
          "temp_max": 283.71,
          "pressure": 1016,
          "humidity": 93
        },
        ...
    },
    ...
]
```


__**NOTE**__: You can get weather details for multiple cities in one API call - check https://openweathermap.org/current#severalid. Extra credits if you make an Ajax call to fetch the weather data and then pass them as props to the CitiesWeatherReport component when rendering it.

10. Create a Calculator component. It has 2 input boxes that take user inputs (operand), and a dropdown that has 4 basic arithmetic operations - +, -, *, /. There is a button that user clicks after entering the operand values and selecting the operation. The complete expression, and the result should be displayed in the component.

11. Implement a WeatherTracker component that displays the weather for a list of cities. It render a CityInput component that accepts a city name and adds it to a list, as child. It also displays a CitiesWeatherReport element that shows weather reports for cities in the cities list that is maintained.

First decide what needs to be maintained in state. Which components need to work with the list of cities? Where will this state be maintained - WeatherTracker (parent), CityInput, or CitiesWeatherReport? Where will you define the method to add a city to the list of cities? How will CityInput be able to trigger this method?
Hint: You will need to fire a custom event from CityInput.

__**NOTE**__:
a. The created() method of CitiesWeatherReport needs to make an Ajax call to fetch weather details for every city in the list, every time it is rendered.
b. Fetching data for a city may be redundant, as previous Ajax calls may have fetched the data for existing cities. Try creating a data caching utility which maintains the data fetched for a particular city. Once an Ajax call to fetch data for a city is made freshly, the cached data is updated. If another call goes out for the same city within 15 minutes (say) on the same date, then the cached data may instead be used. This will take some amount of involved design and implementation. The data cache may be part of the state maintained by a component. This is completely optional.

12. We shall add filter functionality in the WeatherTracker component. Add a CityFilter component with a search input. User can type within this input and all cities that have the search term as a substring of the city name are displayed. For example, if "New" is typed, cities like "New York", "New Orleans" are displayed (assuming they exist in the list of added cities).

13. Add a delete button against each city list item in CitiesWeatherReport component. It should remove the city list item when clicked.

14. Add loading, and error state UIs for the Weather tracker application apart from the loaded state UI which already exists. You can display a loading message for CitiesWeatherReport as it fetches the data.

15. Use Vue CLI to create the Weather tracker application. Modularize and organize the code you have created in earlier exercises using a standalone HTML file. Create folders for components, services (that has methods to fetch data from OpenWeatherMap API, and data caching layer in case you have implemented that), models (like CurrentWeather - that models the response from the API), actions etc. You app should work like before.

16. We shall now break down the application into 3 screens.
* On top of each screen is a Navigation Menu - it has 2 links - "All Cities", and " All Reports". Clicking "All Cities" takes user to the landing page (http://localhost:3000/). Clicking "All Reports" shows the page displaying weather reports for all cities (http://localhost:3000/all)
* **http://localhost:3000/**. It shows a list of cities that have been added in the application. The CityInput component accepts the name of a new city. A new component called CitiesList is displayed below it, that shows the names of cititesadded to the application.
* **http://localhost:3000/:city**, where :city is a placeholder for a city's name. Thus an actual route could be http://localhost:3000/new+york or http://localhost:3000/sunnyvale etc. The component displays the weather details for the specified city only.
* **http://localhost:3000/all** - The CitiesWeatherReport component that displays weather report of all added cities is displayed (only limited weather information is displayed) Clicking any one city within this list, displays the detailed weather report for that city (http://localhost:3000/:city)

17. Introduce Vuex into the Weather tracker application.
* Store the cities list and weather data fetched for cities in the Vuex store.
* When making a call to fetch data for a list of list of cities, check whether each city's weather data is already available in the store. If so, exclude that city automatically.