Task 3: Weather Forecast Website

In this project, you will make a web application to check out the weather forecast for the current day and for the next few days. You will use an API to fetch real-time data and then add it to your application. The user will input his/her location and the weather forecast for the next 5 days will be displayed. In addition, a feature to automatically detect the location can add to the versatility of the project.

Skills Required – JavaScript, Node.js, ReactJS.

Creating a weather forecast website involves using APIs to fetch real-time weather data based on user input or location detection. We can use technologies like Node.js for backend logic and React.js for frontend development. Below are the steps to create this project:

Step 1: Set Up Your Development Environment

1. Install Node.js: Download and install Node.js from the official website.
2. Set Up React App: Use Create React App to set up a new React.js project.

```
npx create-react-app weather-forecast-app
```

3. Install Required Packages:

```
cd weather-forecast-app
npm install axios react-geocode
```

Step 2: Get Weather Data from API

1. Sign up for a free API key from a weather service provider like OpenWeatherMap
2. Create a service file (weatherService.js) to handle API calls using Axios.

```
// weatherService.js

import axios from 'axios';

const API_KEY = 'your_api_key';

const BASE_URL = 'https://api.openweathermap.org/data/2.5/forecast';

const getWeatherForecast = async (location) => {

 try {

   const response = await axios.get(`${BASE_URL}?q=${location}&appid=${API_KEY}`);

   return response.data;

 } catch (error) {

   throw error;

 }

};
```

```
export default getWeatherForecast;
```

Step 3: Create Components in React

1. Create a form component (**WeatherForm.js**) to input the location.

```
// WeatherForm.js

import React, { useState } from 'react';

const WeatherForm = ({ onSubmit }) => {

  const [location, setLocation] = useState('');

  const handleSubmit = (e) => {

    e.preventDefault();

    onSubmit(location);

  };

  return (

    <form onSubmit={handleSubmit}>

      <input type="text" value={location} onChange={(e) => setLocation(e.target.value)} />

      <button type="submit">Get Forecast</button>

    </form>

  );

};

export default WeatherForm;
```

2. Create a weather display component (**WeatherDisplay.js**) to show forecast data.

```
// WeatherDisplay.js
import React from 'react';
const WeatherDisplay = ({ forecast }) => {
  return (
    <div>
      {forecast && (
        <ul>
          {forecast.map((item, index) => (
            <li key={index}>{item.main.temp}°C - {item.weather[0].description}</li>
          ))}
        </ul>
      )}
    </div>
  );
};
export default WeatherDisplay;
```

Step 4: Integrate Components and API

1. Update **App.js** to integrate form submission and weather data display.

```
// App.js
import React, { useState } from 'react';
import WeatherForm from './WeatherForm';
import WeatherDisplay from './WeatherDisplay';
```

```
import getWeatherForecast from './weatherService';
function App() {
  const [forecastData, setForecastData] = useState(null);
  const fetchWeatherData = async (location) => {
    try {
      const data = await getWeatherForecast(location);
      setForecastData(data.list);
    } catch (error) {
      console.error('Error fetching weather data:', error);
    }
  };
  return (
    <div className="App">
      <h1>Weather Forecast App</h1>
      <WeatherForm onSubmit={fetchWeatherData} />
      <WeatherDisplay forecast={forecastData} />
    </div>
  );
}
export default App;
```

Step 5: Run Your Application

1. Start the development server:

```
npm start
```

2. Open browser and go to the website to see weather forecast app in action.

Additional Features

- **Location Detection:** We can use the **react-geocode** package to automatically detect the user's location based on their IP address or browser location.
- **Styling:** Add CSS or a UI framework like Bootstrap to style application and make it more visually appealing.
- **Extended Forecast:** Modify the API call to fetch an extended forecast for more than 5 days if needed.

By following these steps and adding additional features as per requirements, can create a weather forecast web application using JavaScript, Node.js, and React.js.