## **Objectives**

* Explain React Forms validation

**React Forms Validation** is a process used to ensure that the data entered by the user into a form meets certain criteria before the form is submitted. It is important to prevent invalid or incomplete data from being sent to the server.

React provides the flexibility to perform both client-side and server-side validation. Client-side validation can be done using conditionals within the component or by using form libraries like Formik, Yup, or React Hook Form.

**Benefits:**

* Improves user experience.
* Prevents unnecessary server requests.
* Ensures clean and reliable data input.
* Identify the differences between React Form and HTML Form

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Feature | State Handling | Input Binding | Form Submission | Flexibility | Validation |
| HTML Form | Managed by the DOM | Done using the default HTML behavior | Uses default form submission | Limited customization | Simple HTML validation attributes |
| React Form | Managed using React state | Controlled via `value` and `onChange` props | Handled using event handlers (onSubmit) | Highly customizable with full control | Custom validation using JavaScript or libraries |

* Explain about controlled components

**Controlled Components** are React components where the form data is handled by the component’s state. The input element’s value is bound to the component’s state, and any user interaction updates the state.

**Example:**

<input type="text" value={this.state.name} onChange={this.handleChange} />

This provides better control and consistency across the application. It also allows integration of validation logic and dynamic behavior based on the input values.

* Identify various React Form input controls

React supports various form input controls, all of which can be managed using state to become controlled components. These include:

* **Text Input (<input type="text">)**: Used for single-line text input.
* **Textarea (<textarea>)**: Used for multi-line text input.
* **Checkbox (<input type="checkbox">)**: Used to represent a true/false value.
* **Radio Buttons (<input type="radio">)**: Used to select one option from multiple choices.
* **Select/Dropdown (<select> and <option>)**: Used to select a value from a dropdown menu.
* **File Input (<input type="file">)**: Used to upload files.

All these inputs can be controlled through React’s state and updated using onChange events.

* Explain how to handle React Forms

Handling forms in React involves creating state variables for form inputs and updating them using event handlers. The general process is:

1. Create a form with input elements.
2. Bind each input to a state variable using the value attribute.
3. Define an onChange handler to update the state whenever the user types.
4. Use the onSubmit handler to process the form data.

This method allows better control over form behavior and supports validation and conditional logic.

* Explain about submitting forms in React

To submit a form in React:

* The form should have an onSubmit event handler.
* Inside the handler, use event.preventDefault() to stop the default form submission behavior.
* Access the form data through the component’s state.
* Display messages, store data, or send it to a server using fetch or axios.

## **Notes**

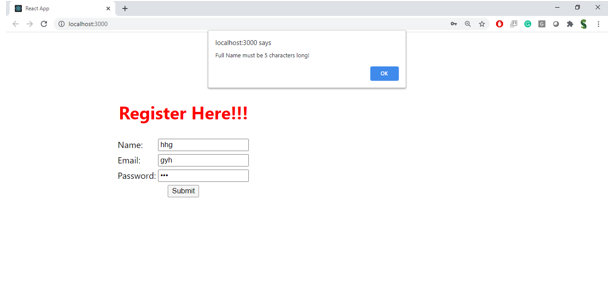
Estimated time to complete this lab: **60 minutes.**

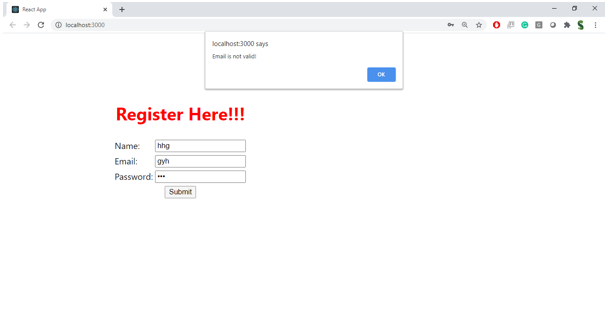
Create a React App named “mailregisterapp” which will have a component named “register.js”. Create a form which accepts the name, email and password and validate the fields as per the following:

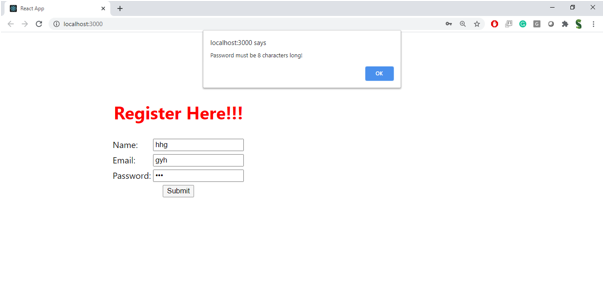
1. Name should have atleast 5 characters
2. Email should have @ and .
3. Password should have atleast 8 characters.

Ensure that validations are implemented through eventhandle and eventsubmit of a form.

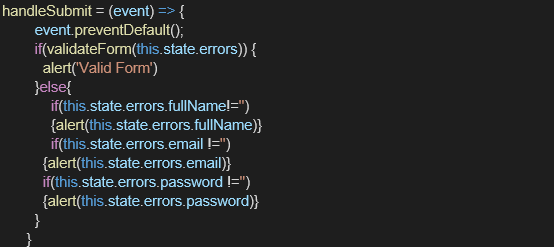
**Output Expected:**

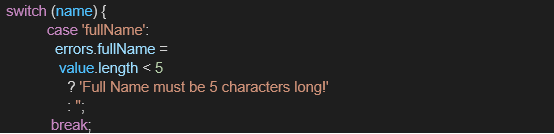


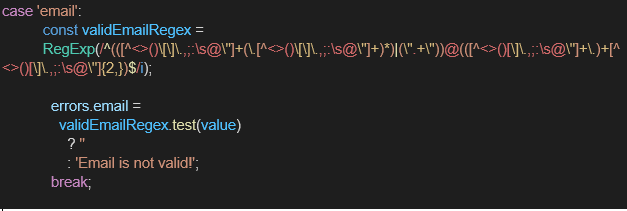




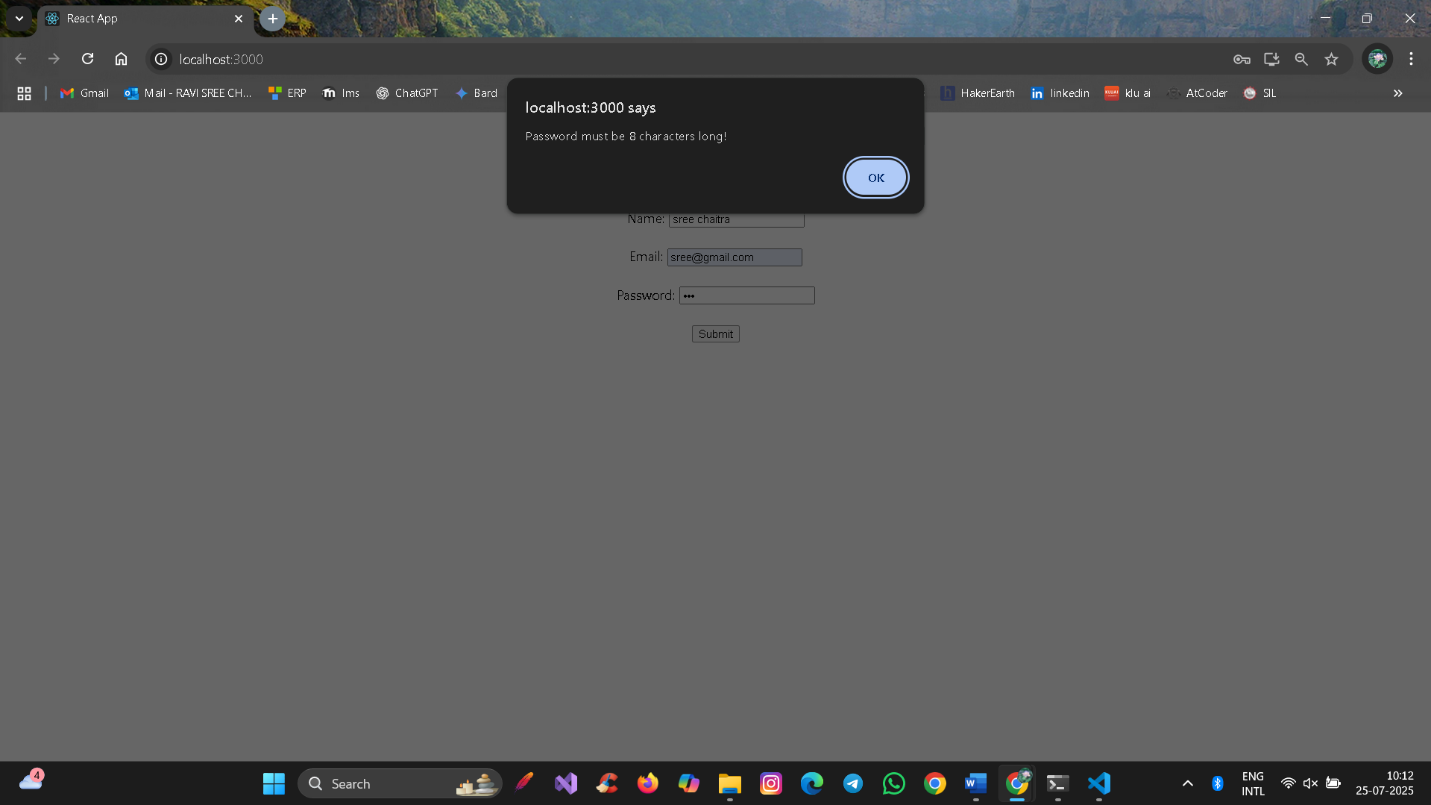
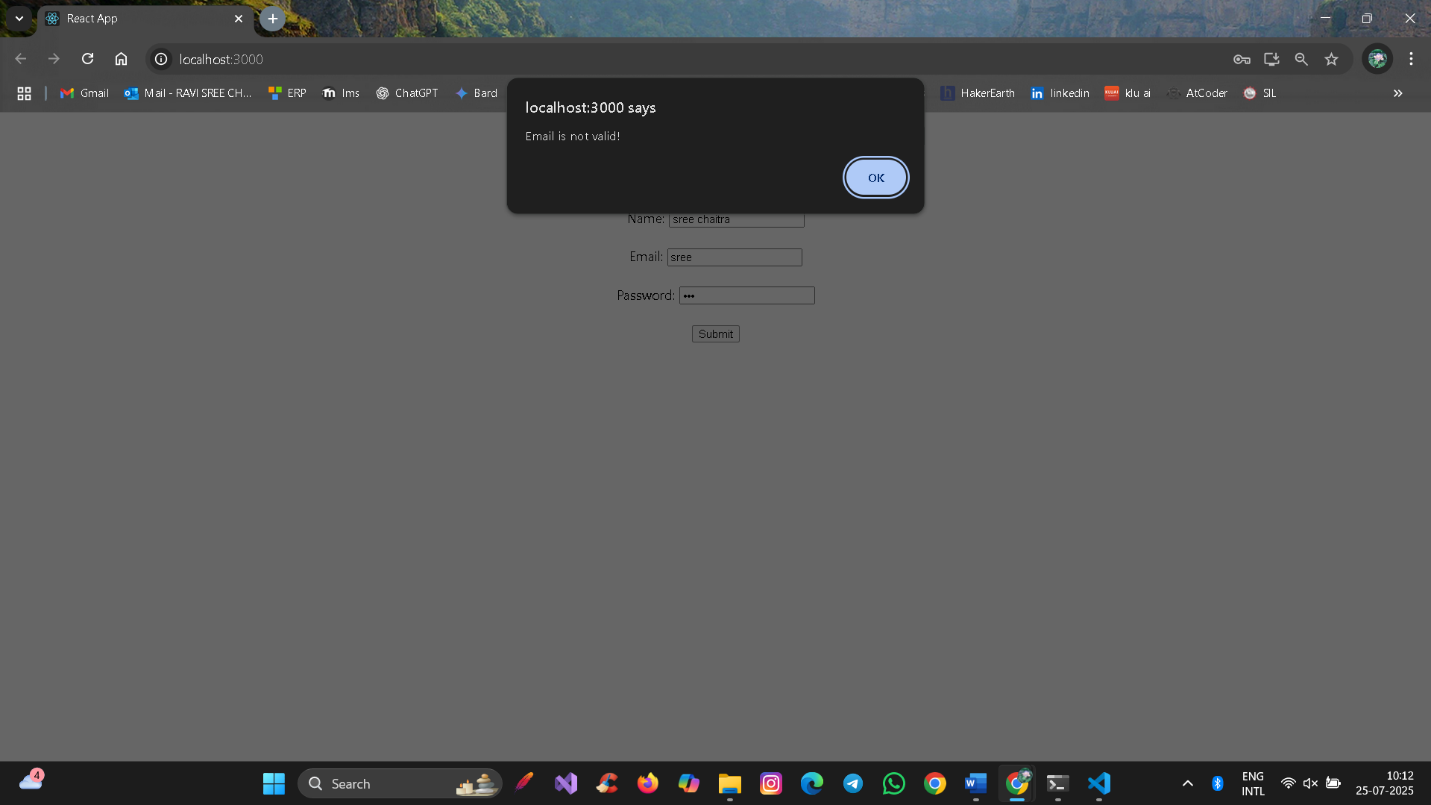
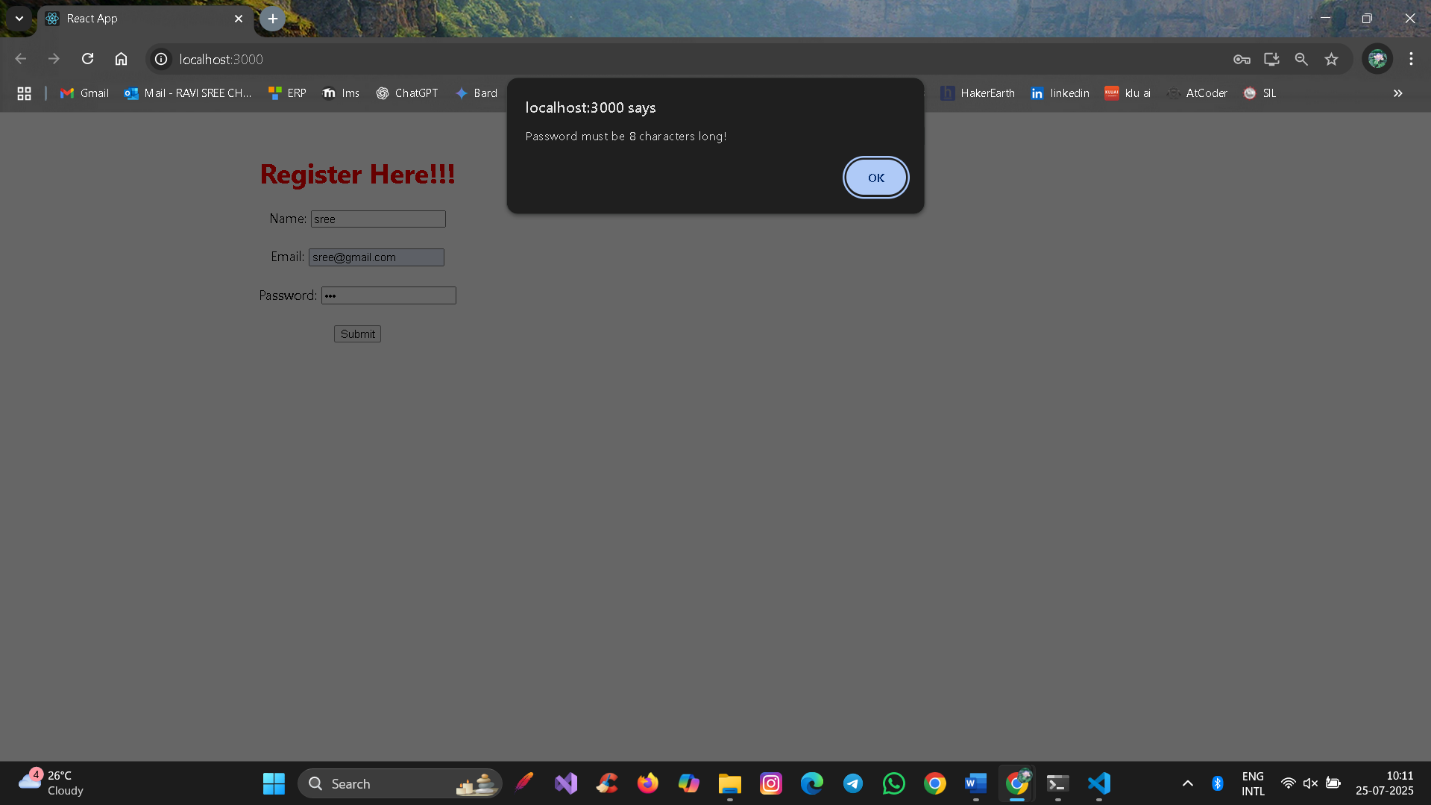
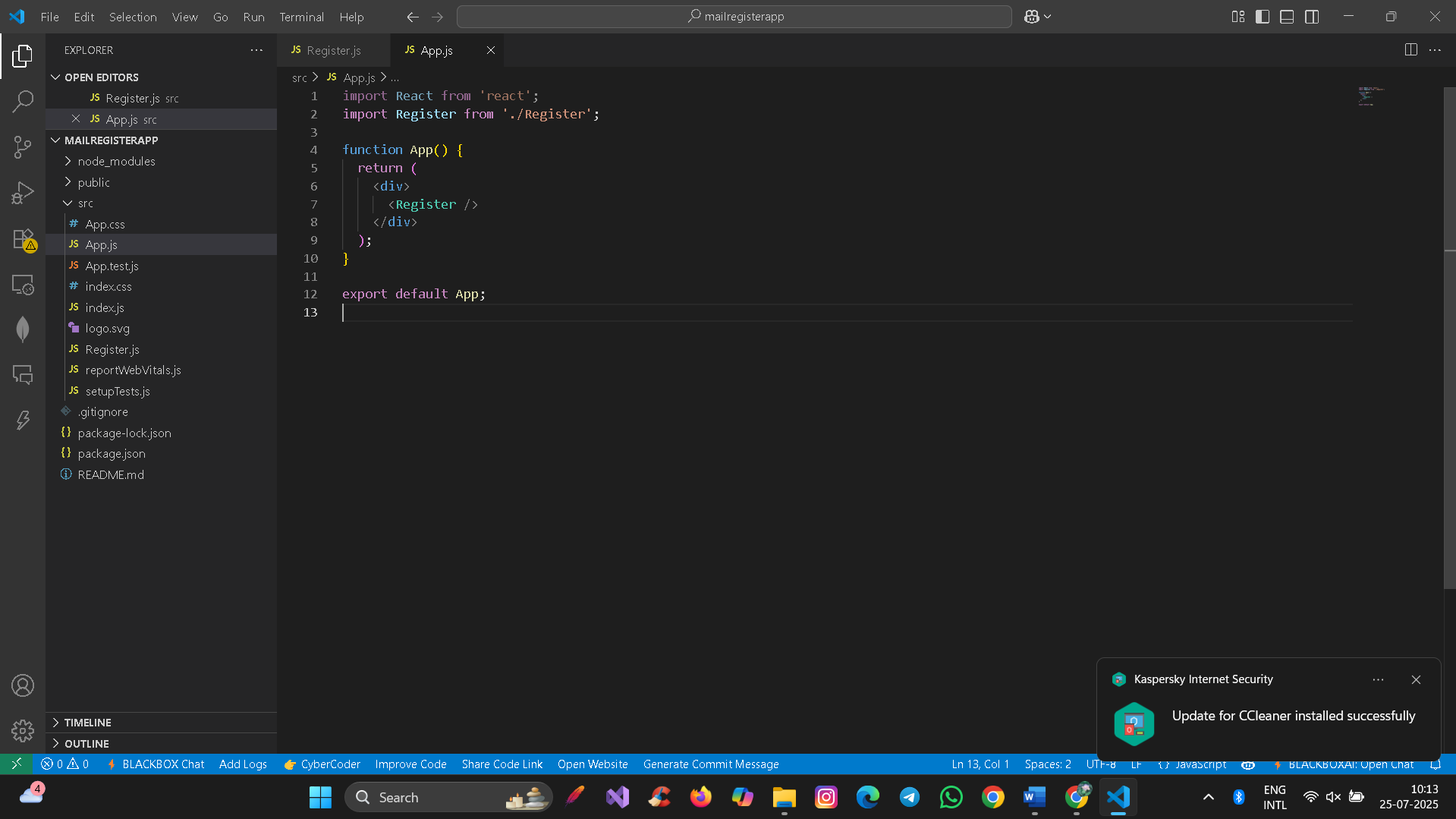
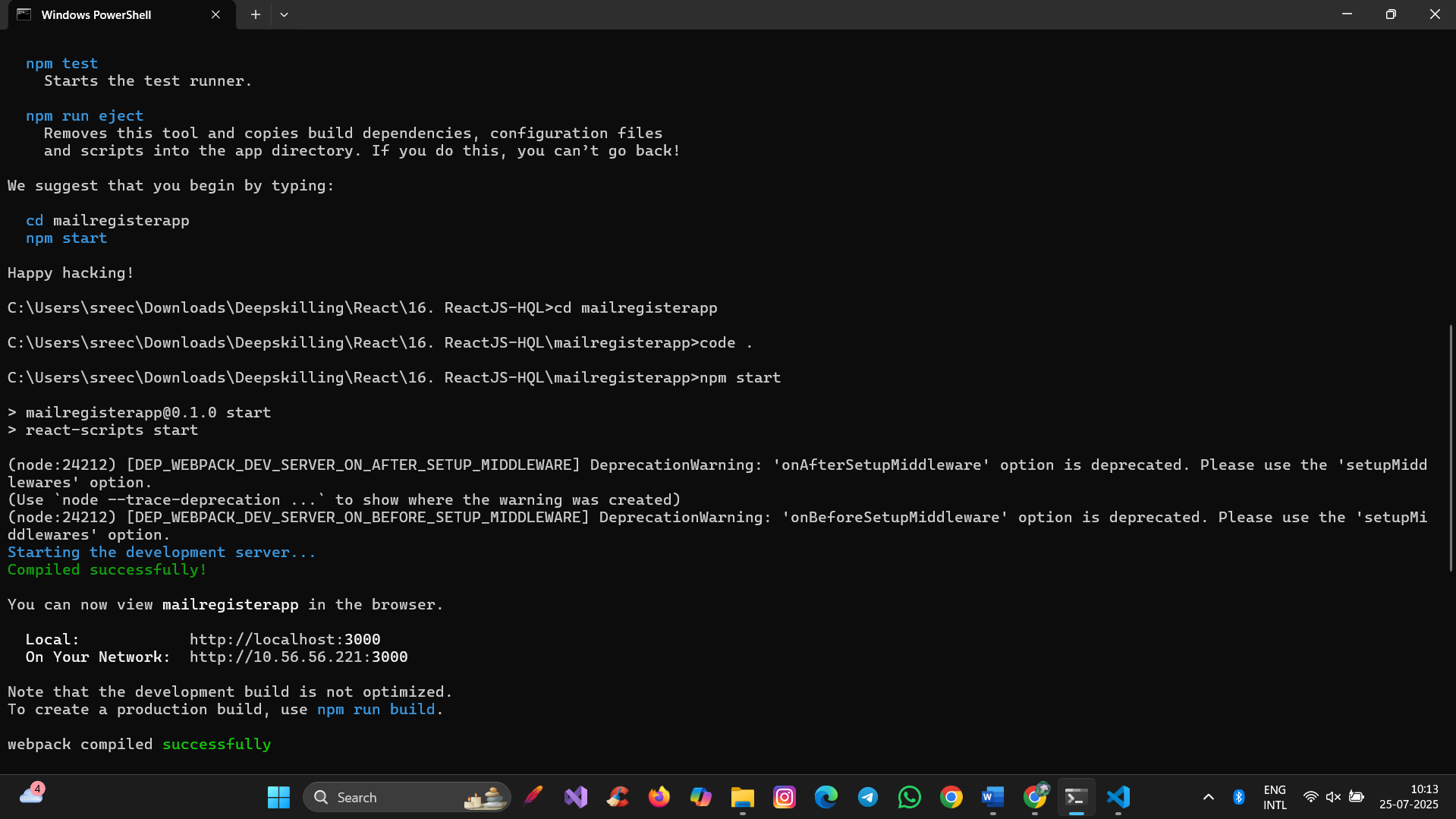
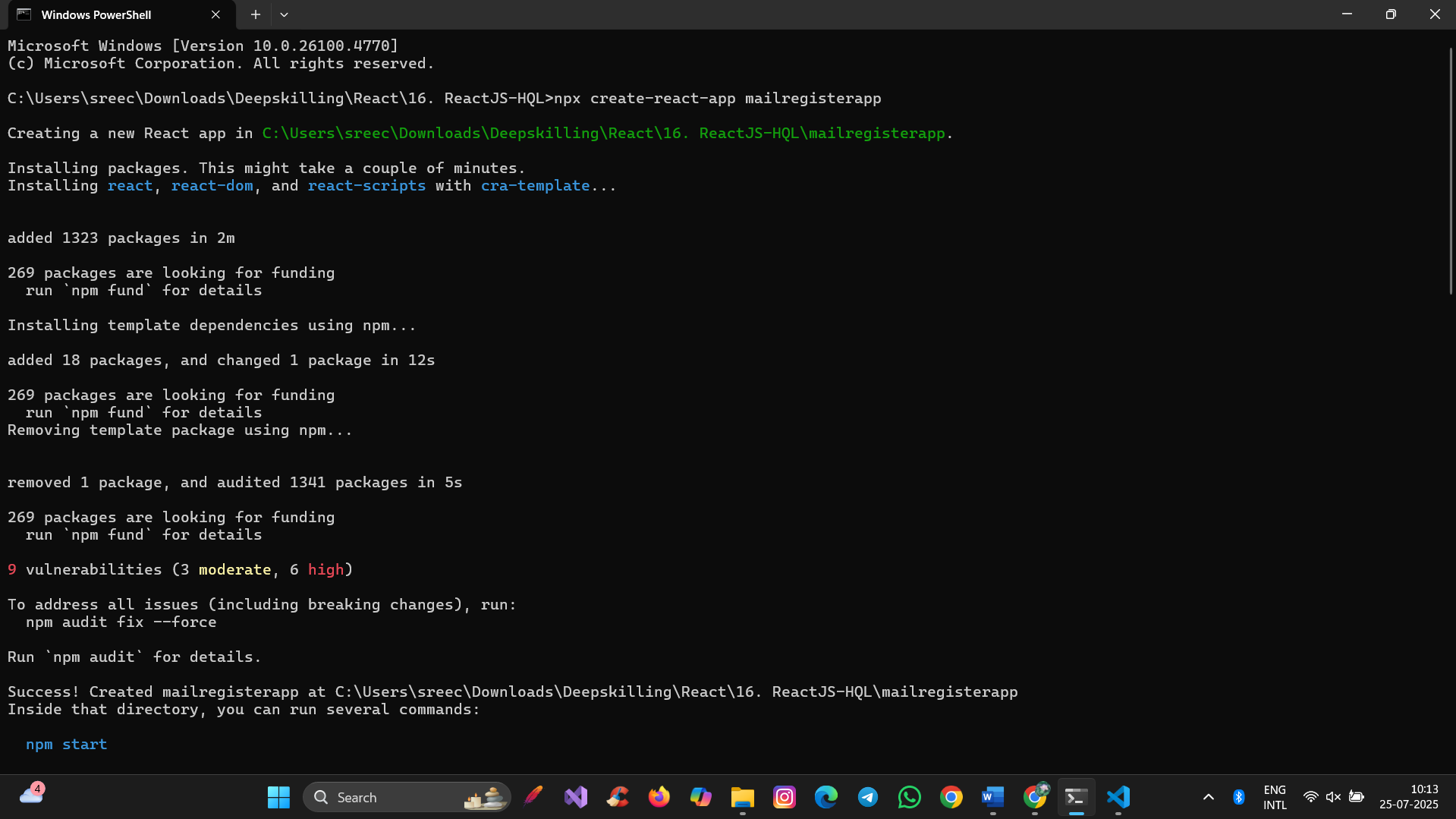
**Hint:**







**HandsOn:**

****