Assignment :- 6B (React JS)

Aim :- WAP to implement the concept of forms and events.

Create a React JS registration Form consisting of textbox, textarea, selection input, check box, radio button, submit button and reset button handling onSubmit, onClick and keyDown events.

Theory :- Creating a React.js registration form that incorporates various input elements (textbox, textarea, selection input, checkbox, radio button) and handles events such as `onSubmit`, `onClick`, and `onKeyDown` involves several key concepts. Below, I'll provide you with a theoretical overview of these concepts:

1. React Components : In React, user interfaces are built using components. Components are reusable, self-contained pieces of UI that can be composed to create complex applications. For this assignment, you'll need to create a registration form component that encapsulates all the form elements and their behavior.

2. State Management : React components often have state that determines their behavior and appearance. State is managed using the `useState` hook (for functional components) or `this.state` (for class components). You'll use state to keep track of the data entered into the form fields and whether the popup should be displayed.

3. Form Elements : Your registration form will include various form elements:
   - Text Input : For capturing the user's name and email address.
   - Text area : For capturing additional messages or comments.
   - Select Input : For allowing users to choose their gender.
   - Checkbox : For agreeing to terms and conditions.
   - Radio Buttons : For selecting options (e.g., gender).

4. Event Handling :
   - `onChange`: Used to capture user input as they type. For instance, you'll use it to update the state as the user types into text inputs, textarea, or selects options in the form.
   - `onClick`: Applied to buttons to trigger an action when the button is clicked. In your case, it will be used for the reset button.
   - `onSubmit`: Applied to the form itself to handle form submission. When the user clicks the submit button, this event is triggered, allowing you to process the form data.
   - `onKeyDown`: You can use this event to capture key presses on specific elements. For instance, you can use it to trigger an action when a specific key is pressed.

5. Conditional Rendering : You'll use conditional rendering to display the popup/modal when certain conditions are met. In your case, the popup will be displayed when the form is successfully submitted.

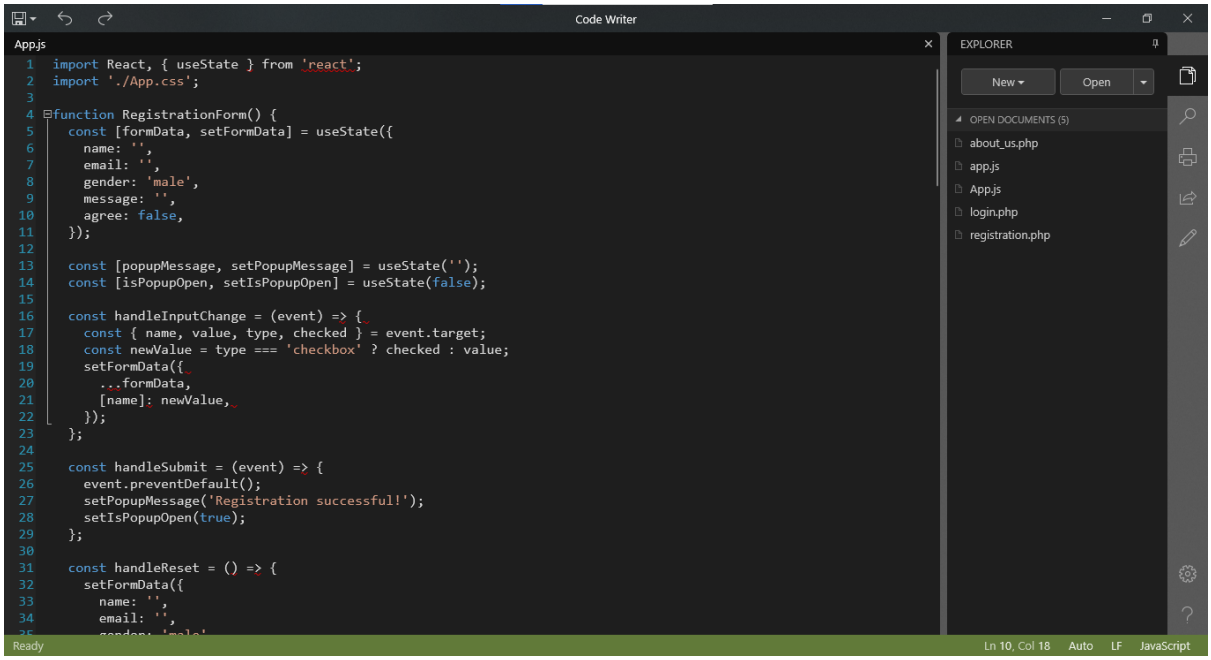
6. CSS Styling : CSS is used to style the form elements and the popup/modal. Proper styling enhances the user experience by making the form visually appealing.

7. Handling Form Submission : Inside the `onSubmit` event handler, you can perform actions such as data validation and sending the data to a server. In your example, it displays a popup message indicating successful registration.

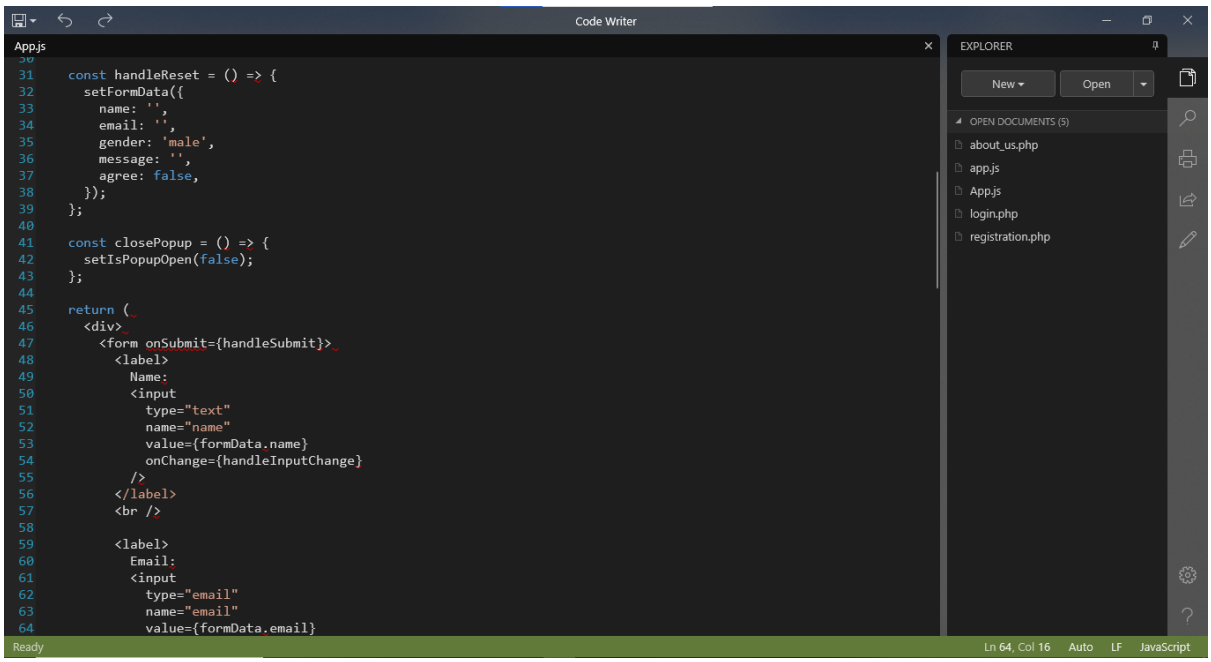
8. Popup/Modal : The popup/modal is a UI element that appears on top of the main content, usually to provide additional information or feedback to the user. It's controlled by state and can be shown or hidden based on user actions or conditions.

By understanding these concepts, you can create a React registration form that meets the requirements of your assignment. Remember to carefully structure your React components, manage state effectively, handle events correctly, and apply appropriate CSS styling to make your form and popup visually appealing and functional.

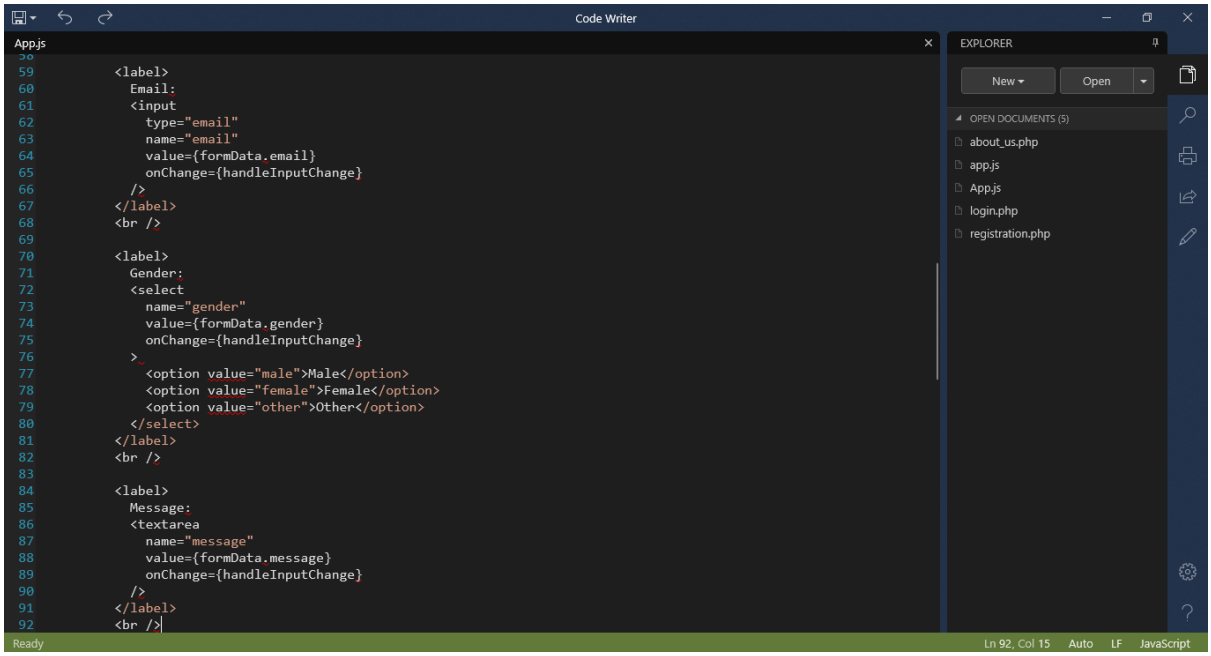
Code :-




```
import React, { useState } from 'react';
import './App.css';

function RegistrationForm() {
  const [formData, setFormData] = useState({
    name: '',
    email: '',
    gender: 'male',
    message: '',
    agree: false,
  });

  const [popupMessage, setPopupMessage] = useState('');
  const [isPopupOpen, setIsPopupOpen] = useState(false);

  const handleInputChange = (event) => {
    const { name, value, type, checked } = event.target;
    const newValue = type === 'checkbox' ? checked : value;
    setFormData({
      ...formData,
      [name]: newValue,
    });
  };

  const handleSubmit = (event) => {
    event.preventDefault();
    setPopupMessage('Registration successful!');
    setIsPopupOpen(true);
  };

  const handleReset = () => {
    setFormData({
      name: '',
      email: '',
```



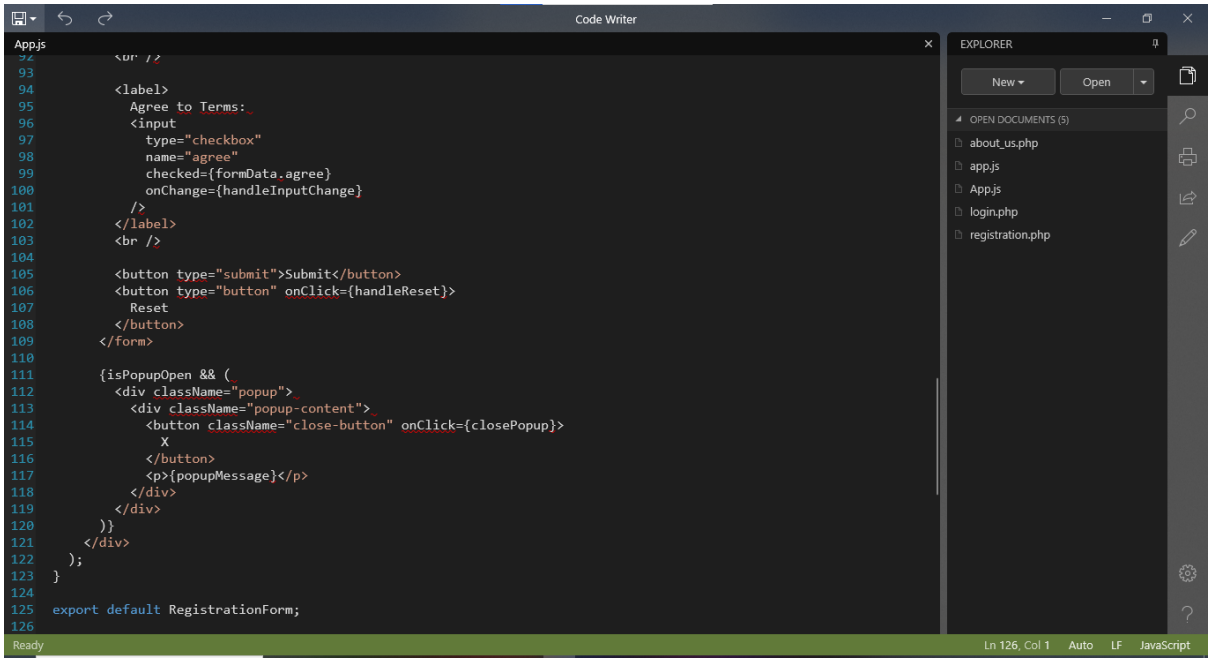

```
  const handleReset = () => {
    setFormData({
      name: '',
      email: '',
      gender: 'male',
      message: '',
      agree: false,
    });
  };

  const closePopup = () => {
    setIsPopupOpen(false);
  };

  return (
    <div>
      <form onSubmit={handleSubmit}>
        <label>
          Name:
          <input
            type="text"
            name="name"
            value={formData.name}
            onChange={handleInputChange}
          />
        </label>
        <br />

        <label>
          Email:
          <input
            type="email"
            name="email"
            value={formData.email}
```


Code Writer
App.js
EXPLORER
New
Open
OPEN DOCUMENTS (5)
about_us.php
app.js
App.js
login.php
registration.php
<label>
Email:
<input
type="email"
name="email"
value={formData.email}
onChange={handleInputChange}
/>
</label>

<label>
Gender:
<select
name="gender"
value={formData.gender}
onChange={handleInputChange}
>
<option value="male">Male</option>
<option value="female">Female</option>
<option value="other">Other</option>
</select>
</label>

<label>
Message:
<textarea
name="message"
value={formData.message}
onChange={handleInputChange}
/>
</label>

Ready
Ln 92, Col 15
Auto
LF
JavaScript


Code Writer
App.js
EXPLORER
New
Open
OPEN DOCUMENTS (5)
about_us.php
app.js
App.js
login.php
registration.php
<label>
Agree to Terms:
<input
type="checkbox"
name="agree"
checked={formData.agree}
onChange={handleInputChange}
/>
</label>

<button type="submit">Submit</button>
<button type="button" onClick={handleReset}>
Reset
</button>
</form>
{isPopupOpen && (

<button className="close-button" onClick={closePopup}>
X
</button>
{popupMessage}

)}

);
}
export default RegistrationForm;
Ready
Ln 126, Col 1
Auto
LF
JavaScript

App - Notepad

File Edit Format View Help

```
form {
  max-width: 400px;
  margin: 0 auto;
  padding: 20px;
  border: 1px solid #ccc;
  background-color: #f9f9f9;
  border-radius: 5px;
  box-shadow: 0px 0px 5px rgba(0, 0, 0, 0.3);
}

label {
  display: block;
  margin-bottom: 10px;
  font-weight: bold;
}

input[type="text"],
input[type="email"],
select,
textarea {
  width: 100%;
  padding: 10px;
  margin-bottom: 15px;
  border: 1px solid #ccc;
  border-radius: 4px;
  font-size: 16px;
}

button[type="submit"],
button[type="button"] {
  background-color: #007bff;
  color: #fff;
  border: none;
  padding: 10px 20px;
  border-radius: 4px;
```

App - Notepad

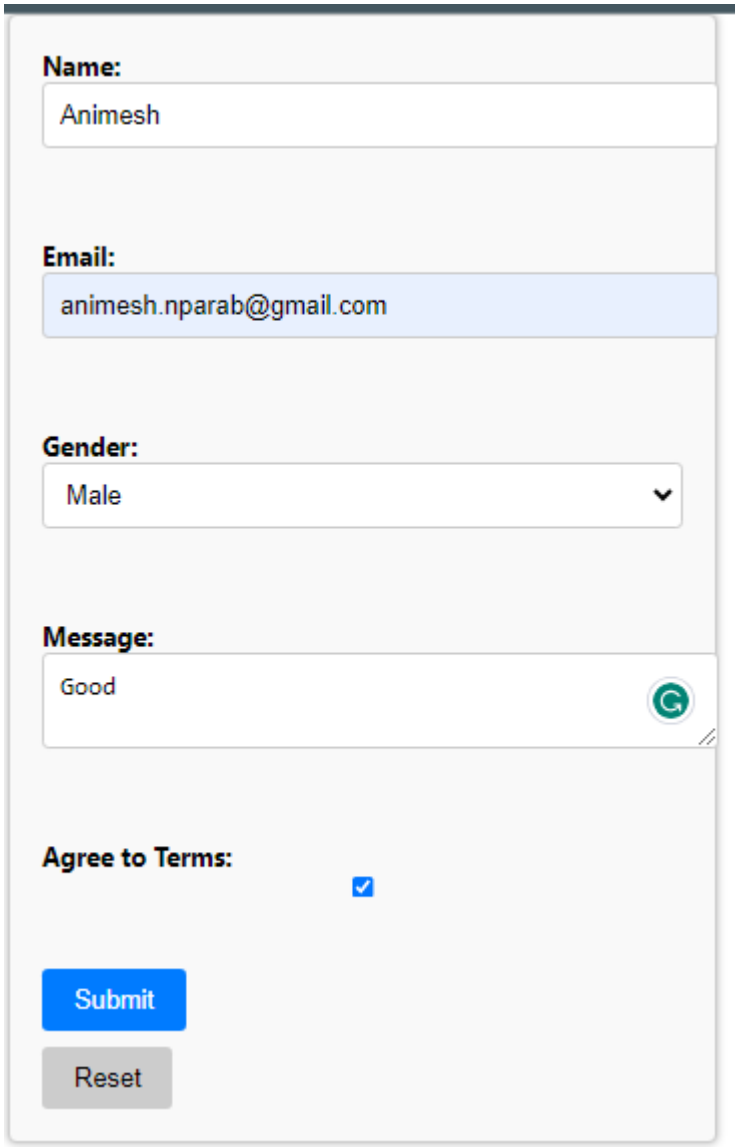
File Edit Format View Help

```
button[type="submit"],
button[type="button"] {
  background-color: #007bff;
  color: #fff;
  border: none;
  padding: 10px 20px;
  border-radius: 4px;
  font-size: 16px;
  cursor: pointer;
  margin-right: 10px;
}


button[type="button"] {
  background-color: #ccc;
  color: #000;
}


label[for="agree"] {
  font-weight: normal;
}


form > * {
  display: flex;
  flex-direction: column;
}

form > * + * {
  margin-top: 10px;
}

form button {
  align-self: flex-start;
```

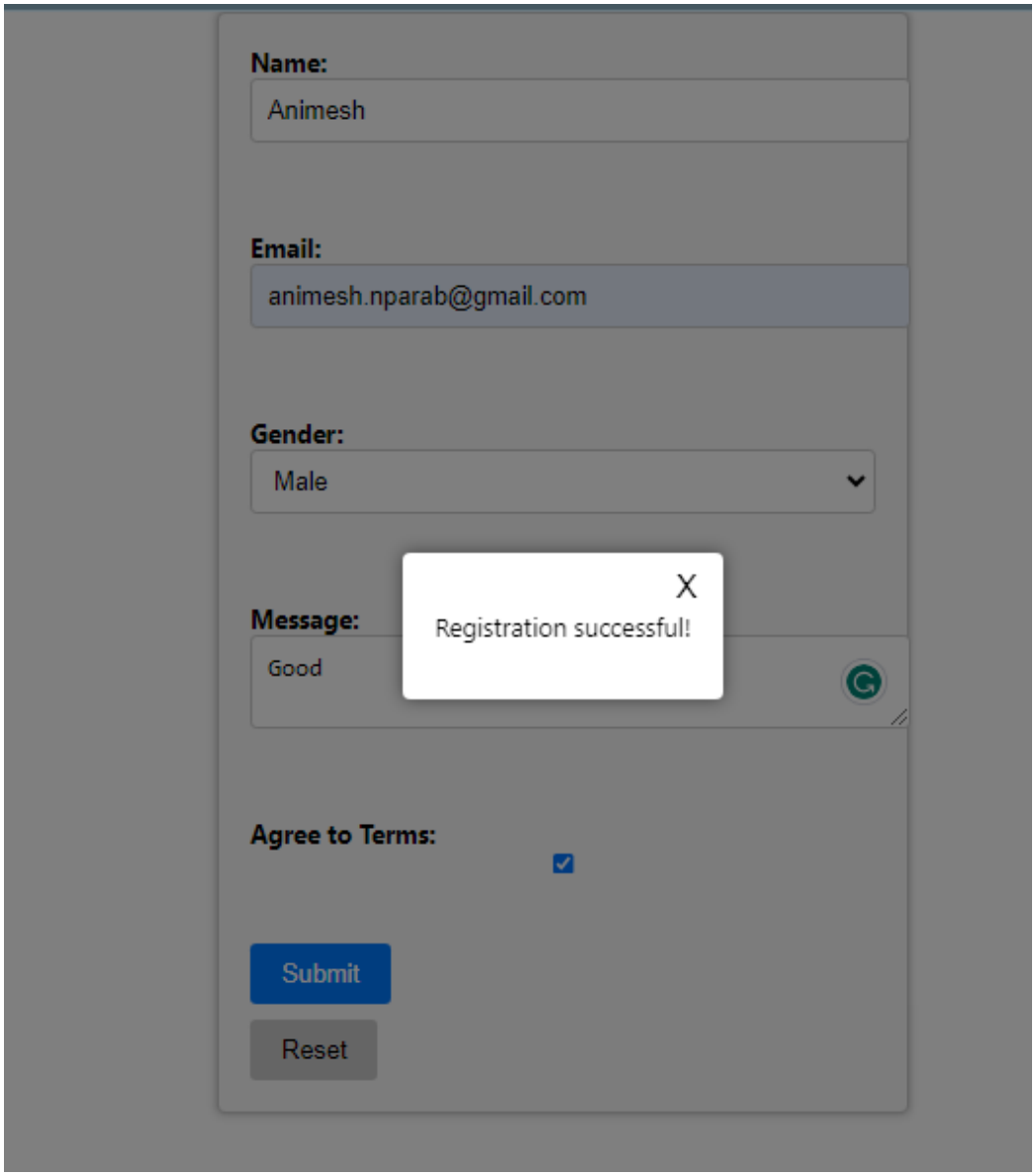
Ln 54, Col 11 | 100% | Unix (LF) | UTF-8


App - Notepad
File Edit Format View Help
}
.popup {
position: fixed;
top: 0;
left: 0;
width: 100%;
height: 100%;
background: rgba(0, 0, 0, 0.5);
display: flex;
justify-content: center;
align-items: center;
}
.popup-content {
background: #fff;
padding: 20px;
border-radius: 5px;
box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.5);
text-align: center;
position: relative;
}
.close-button {
position: absolute;
top: 10px;
right: 10px;
background: transparent;
border: none;
cursor: pointer;
font-size: 20px;
}
Ln 100, Col 1
100%
Unix (LF)
UTF-8

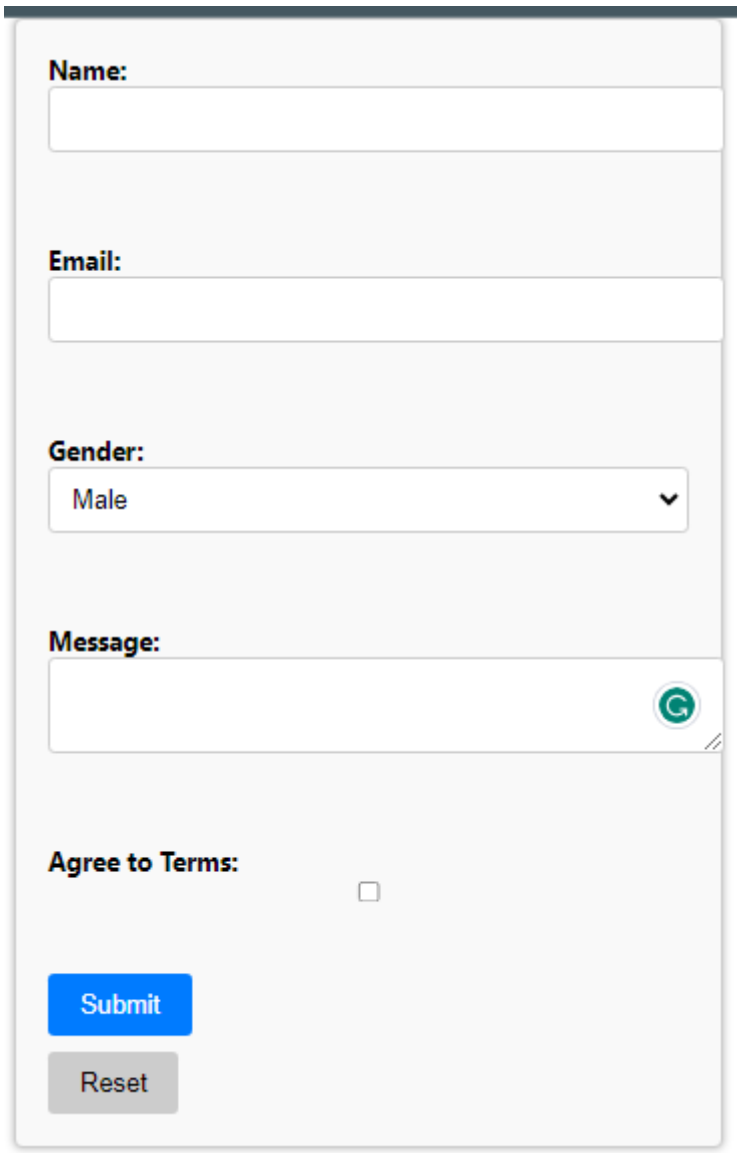
Code output :-


Name:
Animesh
Email:
animesh.nparab@gmail.com
Gender:
Male
Message:
Good
Agree to Terms:
Submit
Reset

After clicking Submit button



After clicking Reset Button



Conclusion :- In this assignment, we implemented a React.js registration form that demonstrates several fundamental concepts of React development, including form handling, event management, and state management.