

M04 MATERI 2

1. TUJUAN

CPMK : Mahasiswa mengetahui dan memahami konsep dasar React JS dan implementasinya dalam pengembangan web.

Sub-CPMK :

- a. Mahasiswa mampu menerapkan styling pada proyek ReactJS menggunakan teknik modern (CSS Modules atau Styled Components), serta memahami konsep lanjutan seperti hooks.
- b. Mahasiswa mampu membuat form dalam ReactJS, menerapkan validasi, dan mengintegrasikan API ke dalam aplikasi React untuk mengambil atau mengirim data.

2. DURASI WAKTU

1 pertemuan x 3 jam

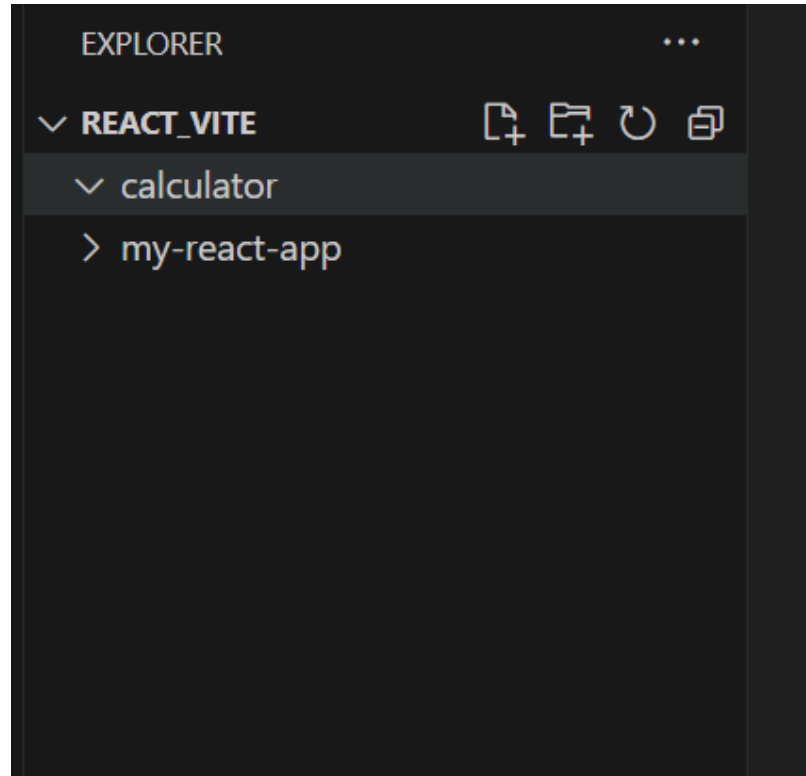
3. DASAR TEORI

React JS

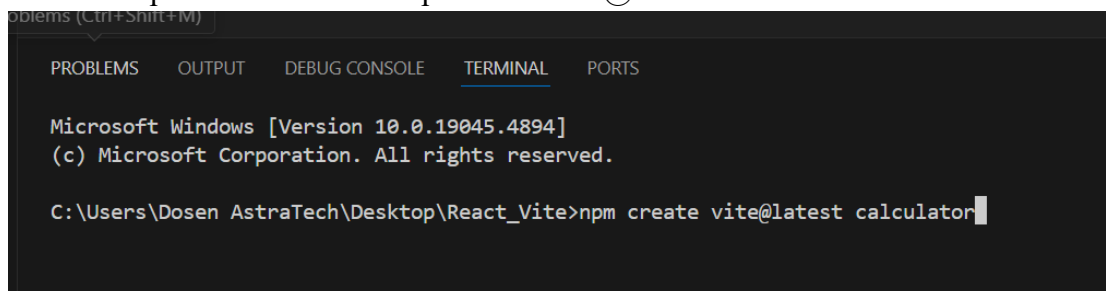
4. Percobaan

A. Membuat Kalkulator sederhana disertai validasi inputan.

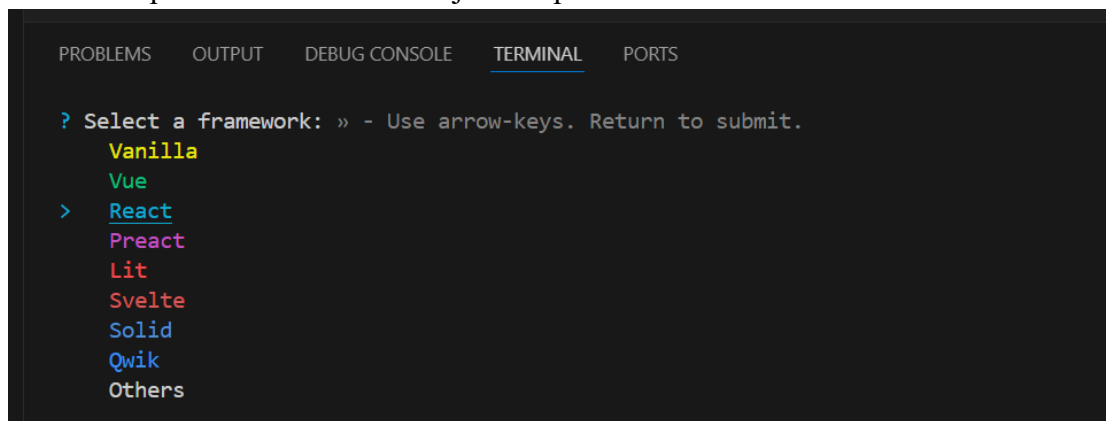
1. Buat folder baru dengan nama calculator

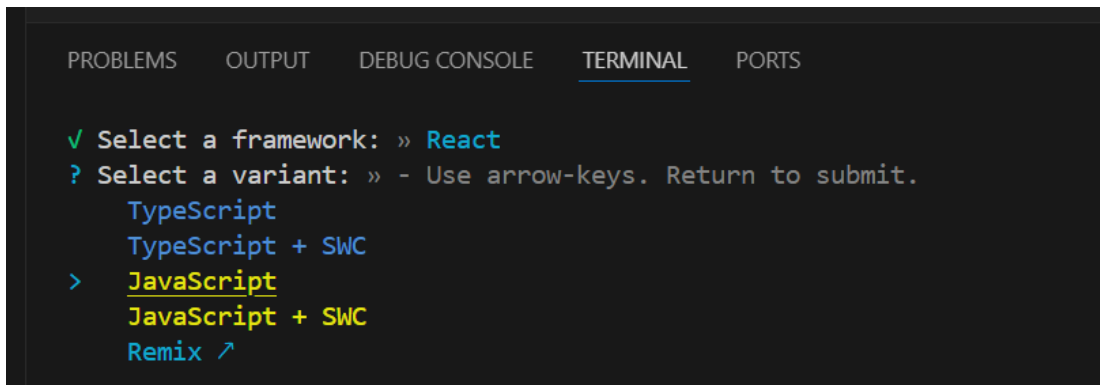


2. Kemudian pada terminal ketik `npm create vite@latest calculator`



3. Kemudian pilih react & kemudian javascript

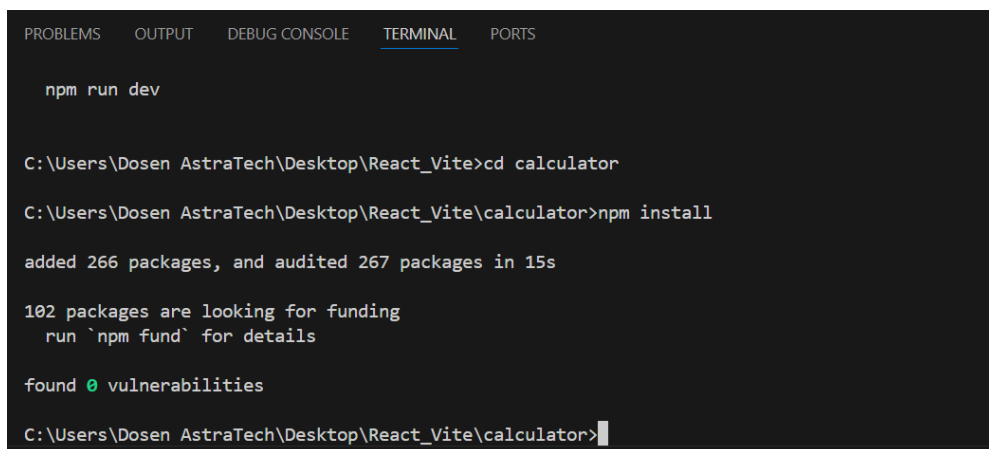




```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

✓ Select a framework: » React
? Select a variant: » - Use arrow-keys. Return to submit.
  TypeScript
  TypeScript + SWC
>  JavaScript
   JavaScript + SWC
   Remix ↗
```

4. Kemudian lakukan langkah-langkah yang tertera pada terminal yaitu `cd calculator`, `npm install`.



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

npm run dev

C:\Users\Dosen AstraTech\Desktop\React_Vite>cd calculator
C:\Users\Dosen AstraTech\Desktop\React_Vite\calculator>npm install

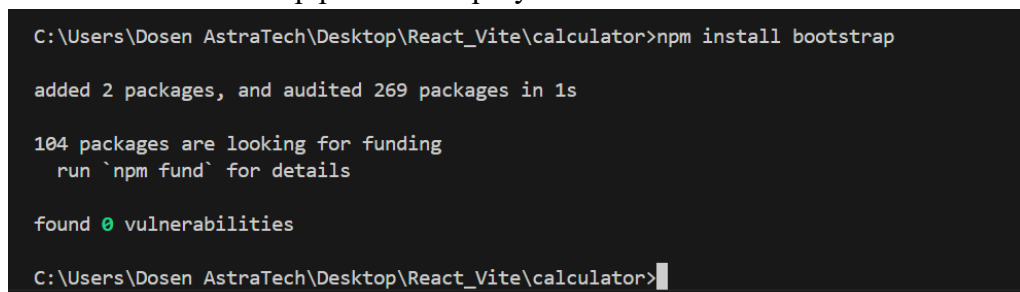
added 266 packages, and audited 267 packages in 15s

102 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\Users\Dosen AstraTech\Desktop\React_Vite\calculator>
```

5. Lakukan instalasi bootstrap pada folder proyek kita



```
C:\Users\Dosen AstraTech\Desktop\React_Vite\calculator>npm install bootstrap

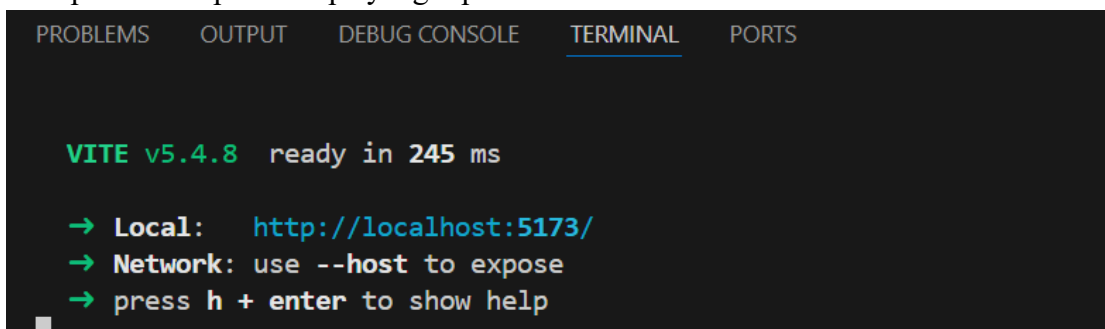
added 2 packages, and audited 269 packages in 1s

104 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\Users\Dosen AstraTech\Desktop\React_Vite\calculator>
```

6. Kemudian jalankan proyek kita dengan mengetikkan `npm run dev`. Port yang ditampilkan akan mempunyai kemungkinan berbeda satu sama lainnya. Penting untuk memperhatikan port berapa yang dipakai.

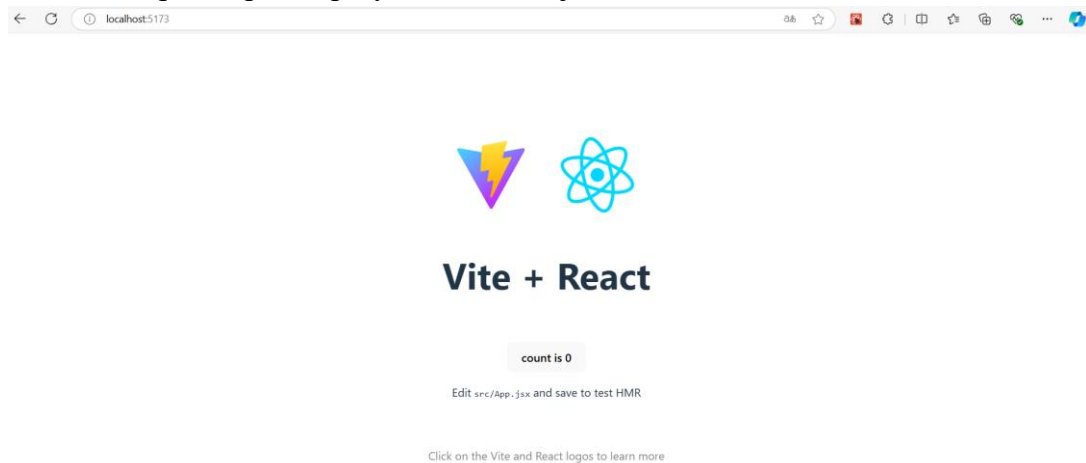


```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

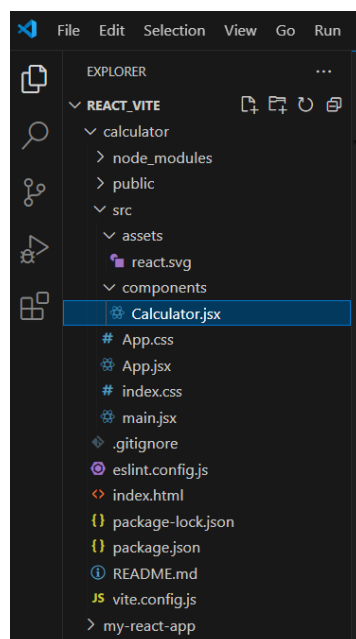
VITE v5.4.8 ready in 245 ms

→ Local:   http://localhost:5173/
→ Network: use --host to expose
→ press h + enter to show help
```

7. Contoh tampilan apabila proyek berhasil dijalankan



8. Kemudian buat sub-folder components di dalam folder src untuk menampung semua komponen yang akan kita buat. Kemudian buat file dengan nama calculator.jsx pada folder components



9. Kemudian pada file calculator.jsx ketikkan code seperti di bawah ini.

```
calculator > src > components > Calculator.jsx > Calculator > handleInputChange
1  import React, { useState } from 'react';
2
3  function Calculator() {
4    const [input, setInput] = useState('');
5    const [result, setResult] = useState('');
6    const [errorMessage, setErrorMessage] = useState('');
7
8    // Menghandle perubahan input
9    const handleInputChange = (e) => {
10     const value = e.target.value;
11     const validPattern = /^[0-9+-*/]*$/; // Regex untuk angka dan operator tanpa spasi
12
13     if (validPattern.test(value)) {
14       setInput(value);
15       setErrorMessage('');
16     } else {
17       setErrorMessage('Input hanya boleh berupa angka dan operator (+, -, *, /)');
18     }
19   };
20
21   // Fungsi untuk melakukan perhitungan berdasarkan operator
22   const calculate = () => {
23     try {
24       // Regex untuk mengekstrak angka dan operator
25       const pattern = /(\d+)([+\/-])(\d+)/;
26       const match = input.match(pattern);
27
28       if (match) {
29         const operand1 = parseFloat(match[1]);
30         const operator = match[2];
31         const operand2 = parseFloat(match[3]);
32
33         let calculationResult;
34
35         // Gunakan switch-case untuk operasi
36         switch (operator) {
37           case '+':
38             calculationResult = operand1 + operand2;
39             break;
40           case '-':
41             calculationResult = operand1 - operand2;
42             break;
43           case '*':
44             calculationResult = operand1 * operand2;
45             break;
46           case '/':
47             if (operand2 === 0) {
48               throw new Error('Tidak bisa membagi dengan nol!');
49             }
50             calculationResult = operand1 / operand2;
51             break;
52           default:
53             throw new Error('Operator tidak valid!');
54         }
55
56         setResult(calculationResult);
57       } else {
58         throw new Error('Input tidak valid! Gunakan format: angka operator angka');
59       }
60     } catch (error) {
61       setErrorMessage(error.message);
62       setResult('');
63     }
64   };
65
66   // Menghapus input dan hasil
67   const clear = () => {
68     setInput('');
69     setResult('');
70     setErrorMessage('');
71   };
72 }
```

```

72
73   return [
74     <div className="card mx-auto" style={{ maxWidth: '400px', padding: '20px' }}>
75       <div className="mb-3">
76         <input
77           type="text"
78           className="form-control"
79           placeholder="Masukkan operasi (contoh: 23+4, 4*1)"
80           value={input}
81           onChange={handleInputChange}
82         />
83       </div>
84       {errorMessage && (
85         <div className="alert alert-danger">
86           {errorMessage}
87         </div>
88       )}
89       <button
90         className="btn btn-primary btn-block mb-3"
91         onClick={calculate}
92         disabled={errorMessage !== '' || input === ''}
93       >
94         Hitung
95       </button>
96       <button
97         className="btn btn-danger btn-block mb-3"
98         onClick={clear}
99       >
100        Bersihkan
101      </button>
102      {result && (
103        <div className="alert alert-info">
104          <strong>Hasil: </strong>{result}
105        </div>
106      )}
107    </div>
108  ];
109 }
110
111 export default Calculator;

```

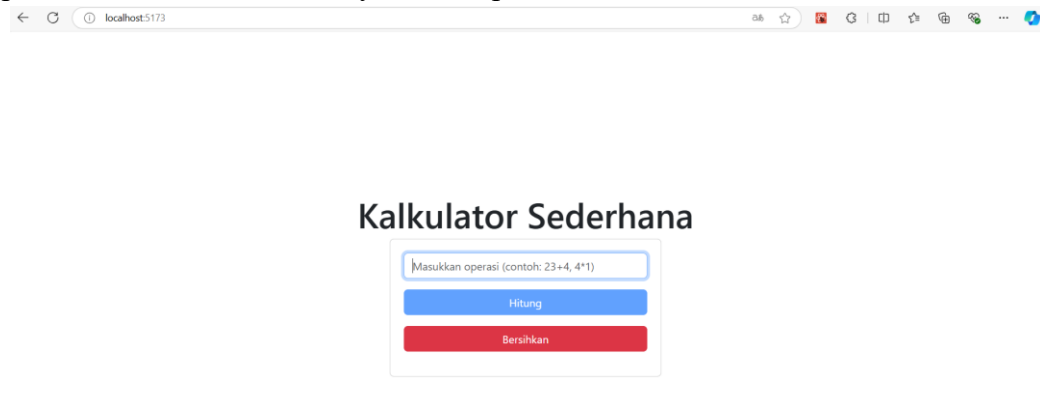
10. Kemudian Pada file App.jsx ketikkan kode seperti di bawah untuk bisa memakai komponen yang telah kita buat sebelumnya.

```

calculator > src > App.jsx > ...
1  import React from 'react';
2  import Calculator from './components/Calculator';
3  import 'bootstrap/dist/css/bootstrap.min.css'; // Pastikan Bootstrap diimpor
4  import './APP.css';
5
6  function App() {
7    return (
8      <div className="App" >
9        <div className="row justify-content-center">
10         <div className="col-md-12 text-center">
11           <h1>Kalkulator Sederhana</h1>
12           <Calculator />
13         </div>
14       </div>
15     </div>
16   );
17 }
18
19 export default App;

```

11. Apabila berhasil, maka hasilnya akan seperti berikut



12. Pada program ini kita tidak bisa mengetikkan inputan selain daripada angka dan operator perhitungan

Kalkulator Sederhana

Input hanya boleh berupa angka dan operator (+, -, *, /)

Hitung

Bersihkan

13. Ini adalah contoh perhitungan yang berhasil

Kalkulator Sederhana

2+3

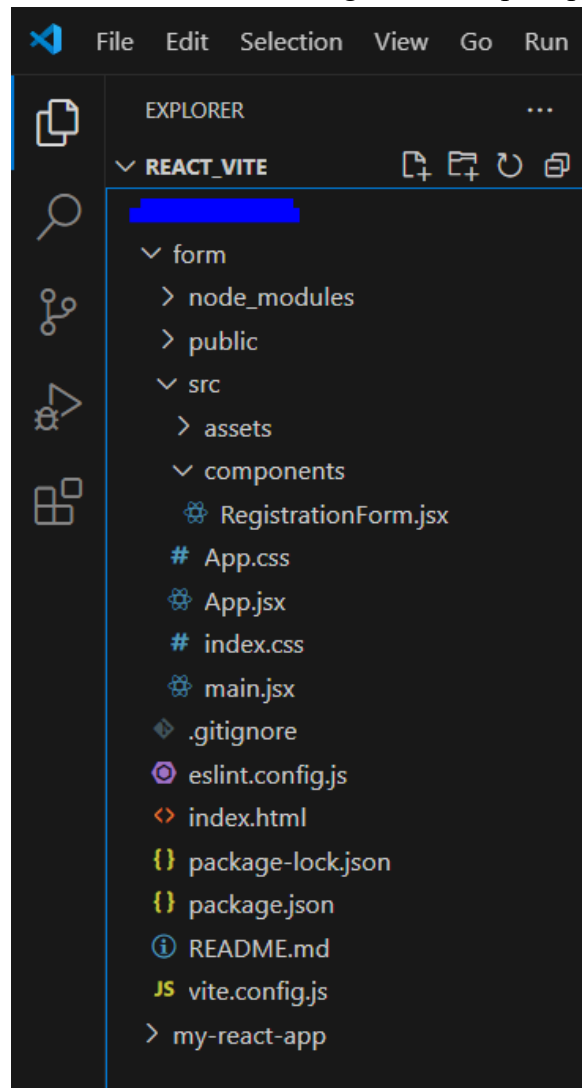
Hitung

Bersihkan

Hasil: 5

B. Membuat Form Registrasi Dengan Validasi

1. Buat struktur folder dengan nama form, kemudian lakukan persis sama seperti pada percobaan membuat kalkulator sederhana namun dengan nama seperti pada di bawah.



2. Buat file dengan nama RegistrationForm. Jsx pada folder components, lalu ketikkan kode sebagai berikut.

```
form > src > components > RegistrationForm.jsx > RegistrationForm
1  import React, { useState, useRef } from 'react';
2
3  const RegistrationForm = () => {
4    const [formData, setFormData] = useState({
5      username: '',
6      email: '',
7      password: '',
8      rePassword: '',
9      birthPlace: '',
10     birthDate: '',
11   });
12
13   const [errors, setErrors] = useState({});
14
15   const usernameRef = useRef();
16   const emailRef = useRef();
17   const passwordRef = useRef();
18   const rePasswordRef = useRef();
19   const birthPlaceRef = useRef();
20   const birthDateRef = useRef();
21
22   const validate = (name, value) => {
23     const newErrors = { ...errors };
24
25     switch (name) {
26       case 'username':
27         if (!value) {
28           newErrors.username = 'Username is required';
29         } else if (!/^[a-zA-Z\s]+$/i.test(value)) {
30           newErrors.username = 'Username Hanya Boleh Huruf Saja';
31         } else {
32           delete newErrors.username;
33         }
34         break;
35       case 'email':
36         if (!value) {
37           newErrors.email = 'Email is required';
38         } else if (!/^[a-zA-Z0-9+@-]+\.[a-zA-Z0-9+@-]+$/i.test(value)) {
39           newErrors.email = 'Email is invalid';
40         } else {
41           delete newErrors.email;
42         }
43         break;
```

```
44     case 'password':
45       if (!value) {
46         newErrors.password = 'Password is required';
47       } else {
48         delete newErrors.password;
49       }
50       break;
51     case 'rePassword':
52       if (value !== formData.password) {
53         newErrors.rePassword = 'Passwords must match';
54       } else {
55         delete newErrors.rePassword;
56       }
57       break;
58     case 'birthPlace':
59       if (!value) {
60         newErrors.birthPlace = 'Birth place is required';
61       } else {
62         delete newErrors.birthPlace;
63       }
64       break;
65     case 'birthDate':
66       if (!value) {
67         newErrors.birthDate = 'Birth date is required';
68       } else {
69         delete newErrors.birthDate;
70       }
71       break;
72     default:
73       break;
74   }
75
76   setErrors(newErrors);
77 };
78
79 const handleChange = (e) => {
80   const { name, value } = e.target;
81   setFormData({ ...formData, [name]: value });
82   validate(name, value);
83 };
```

```
84
85   const handleSubmit = (e) => {
86     e.preventDefault();
87     if (Object.keys(errors).length === 0) {
88       alert('Form submitted successfully!');
89       setFormData({
90         username: '',
91         email: '',
92         password: '',
93         rePassword: '',
94         birthPlace: '',
95         birthDate: '',
96       });
97       setErrors({});
98     } else {
99       const firstErrorKey = Object.keys(errors)[0];
100       switch (firstErrorKey) {
101         case 'username':
102           usernameRef.current.focus();
103           break;
104         case 'email':
105           emailRef.current.focus();
106           break;
107         case 'password':
108           passwordRef.current.focus();
109           break;
110         case 'rePassword':
111           rePasswordRef.current.focus();
112           break;
113         case 'birthPlace':
114           birthPlaceRef.current.focus();
115           break;
116         case 'birthDate':
117           birthDateRef.current.focus();
118           break;
119         default:
120           break;
121       }
122     }
123   };
124
```

```

125     return (
126     <div className="card mx-auto" style={{ maxWidth: '600px', padding: '20px', boxShadow: '0 4px 8px rgba(0, 0, 0, 0.1)' }}>
127     <h3 className="text-center mb-4">Registration Form</h3>
128     <form onSubmit={handleSubmit}>
129       <div className="mb-3 row">
130         <label htmlFor="username" className="col-sm-3 col-form-label">Username</label>
131         <div className="col-sm-9">
132           <input
133             type="text"
134             className={`form-control ${errors.username ? 'is-invalid' : ''}`}
135             id="username"
136             name="username"
137             ref={usernameRef}
138             value={formData.username}
139             onChange={handleChange}
140           />
141           {errors.username && <div className="invalid-feedback">{errors.username}</div>}
142         </div>
143       </div>
144
145       <div className="mb-3 row">
146         <label htmlFor="email" className="col-sm-3 col-form-label">Email</label>
147         <div className="col-sm-9">
148           <input
149             type="email"
150             className={`form-control ${errors.email ? 'is-invalid' : ''}`}
151             id="email"
152             name="email"
153             ref={emailRef}
154             value={formData.email}
155             onChange={handleChange}
156           />
157           {errors.email && <div className="invalid-feedback">{errors.email}</div>}
158         </div>
159       </div>

```

```

160
161       <div className="mb-3 row">
162         <label htmlFor="password" className="col-sm-3 col-form-label">Password</label>
163         <div className="col-sm-9">
164           <input
165             type="password"
166             className={`form-control ${errors.password ? 'is-invalid' : ''}`}
167             id="password"
168             name="password"
169             ref={passwordRef}
170             value={formData.password}
171             onChange={handleChange}
172           />
173           {errors.password && <div className="invalid-feedback">{errors.password}</div>}
174         </div>
175       </div>
176
177       <div className="mb-3 row">
178         <label htmlFor="rePassword" className="col-sm-3 col-form-label">Re-Password</label>
179         <div className="col-sm-9">
180           <input
181             type="password"
182             className={`form-control ${errors.rePassword ? 'is-invalid' : ''}`}
183             id="rePassword"
184             name="rePassword"
185             ref={rePasswordRef}
186             value={formData.rePassword}
187             onChange={handleChange}
188           />
189           {errors.rePassword && <div className="invalid-feedback">{errors.rePassword}</div>}
190         </div>
191       </div>

```

```

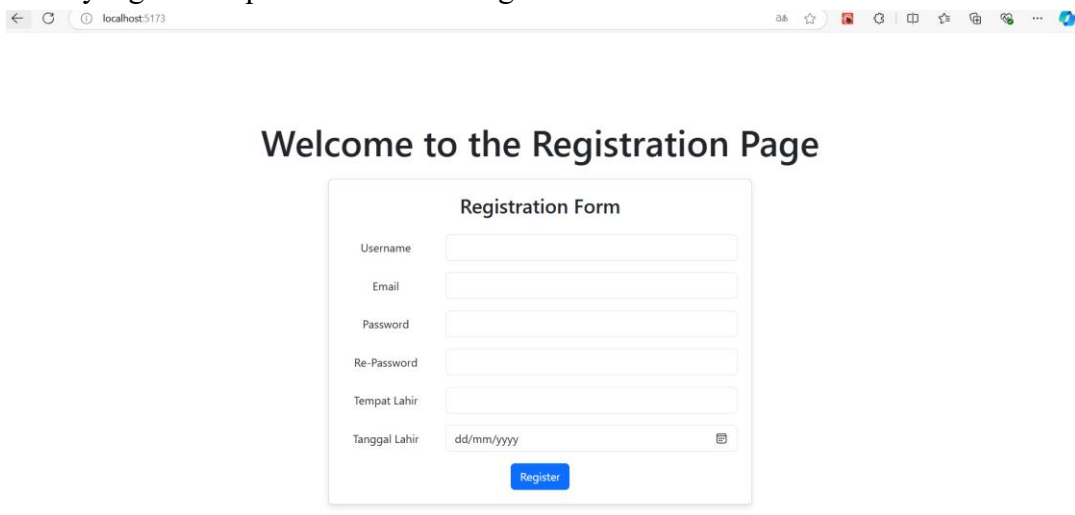
192
193 <div className="mb-3 row">
194   <label htmlFor="birthPlace" className="col-sm-3 col-form-label">Tempat Lahir</label>
195   <div className="col-sm-9">
196     <input
197       type="text"
198       className={`form-control ${errors.birthPlace ? 'is-invalid' : ''}`}
199       id="birthPlace"
200       name="birthPlace"
201       ref={birthPlaceRef}
202       value={formData.birthPlace}
203       onChange={handleChange}
204     />
205     {errors.birthPlace && <div className="invalid-feedback">{errors.birthPlace}</div>}
206   </div>
207 </div>
208
209 <div className="mb-3 row">
210   <label htmlFor="birthDate" className="col-sm-3 col-form-label">Tanggal Lahir</label>
211   <div className="col-sm-9">
212     <input
213       type="date"
214       className={`form-control ${errors.birthDate ? 'is-invalid' : ''}`}
215       id="birthDate"
216       name="birthDate"
217       ref={birthDateRef}
218       value={formData.birthDate}
219       onChange={handleChange}
220     />
221     {errors.birthDate && <div className="invalid-feedback">{errors.birthDate}</div>}
222   </div>
223 </div>
224
225 <div className="text-center">
226   <button type="submit" className="btn btn-primary">Register</button>
227 </div>
228 </form>
229 </div>
230
231 );
232 };
233
234 export default RegistrationForm;

```

3. Kemudian pada file App.jsx pada folder src, ketikkan kode sebagai berikut

```
form > src > App.jsx > ...
1  import React from 'react';
2  import RegistrationForm from './components/RegistrationForm';
3  import 'bootstrap/dist/css/bootstrap.min.css'; // Pastikan Bootstrap diimpor
4  import './App.css';
5
6  const App = () => {
7    return (
8      <div className="container mt-5">
9        <h1 className="text-center mb-4">Welcome to the Registration Page</h1>
10       <RegistrationForm />
11     </div>
12   );
13 };
14
15 export default App;
```

4. Hasil yang akan diperoleh adalah sebagai berikut



localhost:5173

Welcome to the Registration Page

Registration Form

Username

Email

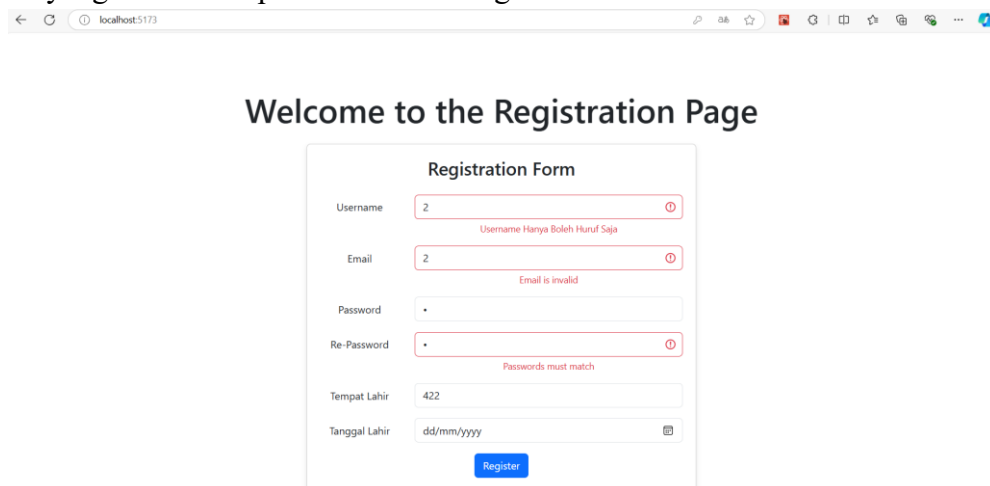
Password

Re-Password

Tempat Lahir

Tanggal Lahir

5. Pesan error yang akan ditampilkan adalah sebagai berikut



localhost:5173

Welcome to the Registration Page

Registration Form

Username Username Hanya Boleh Huruf Saja

Email Email is invalid

Password

Re-Password Passwords must match

Tempat Lahir

Tanggal Lahir

5. Latihan

Buatlah form registrasi dengan tampilan sebagai berikut menggunakan vite + react js dan tampilan menggunakan bootstrap.

Welcome to the Registration Page

Registration

Registration Form

Username

Email

Password

Re-Password

Tempat Lahir

Tanggal Lahir

dd/mm/yyyy

Register

Submitted Data:

Username: Not submitted yet

Email: Not submitted yet

Password: Not submitted yet

Re-Password: Not submitted yet

Tempat Lahir: Not submitted yet

Tanggal Lahir: Not submitted yet

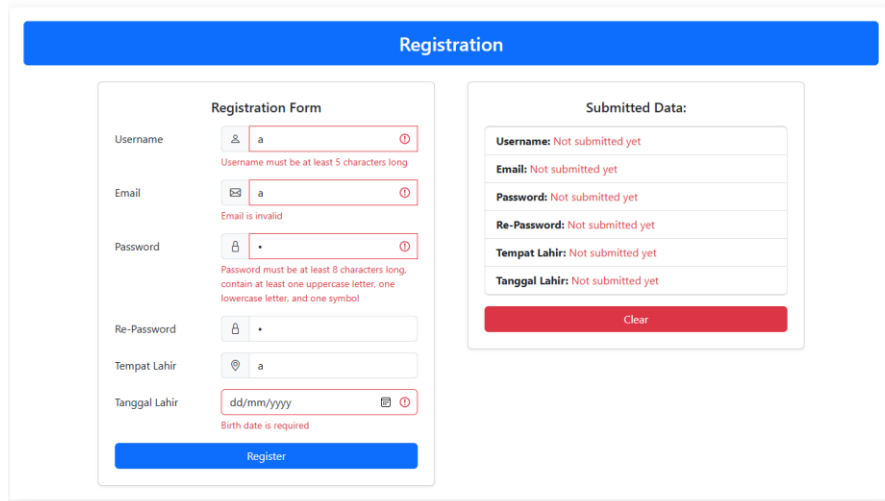
Clear

A. Tampilan Umum

- Buat tampilan semirip mungkin dengan yang telah ditentukan
- Tampilan harus memuat icon pada setiap input. Icon yang dipakai harus icon yang berasal dari framework bootstrap
- Untuk submitted data minimal harus menggunakan class *list-group-item* dari bootstrap
- Tampilan awal pada “submitted data” sebelum diinput seperti ditunjukkan pada gambar di atas

B. Tampilan Pesan Kesalahan harus bersifat langsung di saat pengisian meskipun tanpa menekan tombol register. Pesan kesalahan ditampilkan seperti dibawah ini.

Welcome to the Registration Page



The image shows a registration form titled "Registration" with a blue header. The form is divided into two main sections: "Registration Form" and "Submitted Data".

Registration Form:

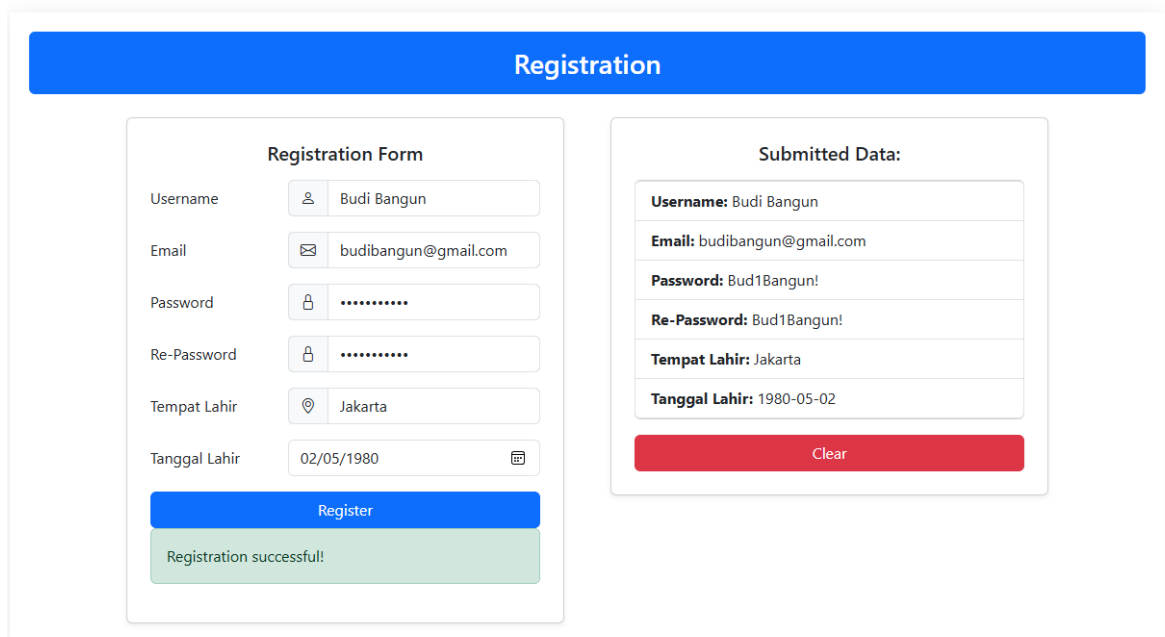
- Username:** Input field with "a". Error message: "Username must be at least 5 characters long".
- Email:** Input field with "a". Error message: "Email is invalid".
- Password:** Input field with "•". Error message: "Password must be at least 8 characters long, contain at least one uppercase letter, one lowercase letter, and one symbol".
- Re-Password:** Input field with "•".
- Tempat Lahir:** Input field with "a".
- Tanggal Lahir:** Input field with "dd/mm/yyyy" and a calendar icon. Error message: "Birth date is required".
- Register Button:** A blue button at the bottom.

Submitted Data:

- Username: Not submitted yet
- Email: Not submitted yet
- Password: Not submitted yet
- Re-Password: Not submitted yet
- Tempat Lahir: Not submitted yet
- Tanggal Lahir: Not submitted yet
- Clear Button:** A red button at the bottom.

C. Tampilan Layar Saat Berhasil

Welcome to the Registration Page



The image shows the same registration form as above, but now it displays successful registration data. The "Submitted Data" section is populated with the user's information.

Registration Form:

- Username:** Input field with "Budi Bangun".
- Email:** Input field with "budibangun@gmail.com".
- Password:** Input field with "••••••••".
- Re-Password:** Input field with "••••••••".
- Tempat Lahir:** Input field with "Jakarta".
- Tanggal Lahir:** Input field with "02/05/1980" and a calendar icon.
- Register Button:** A blue button at the bottom.
- Registration successful!** A green message box below the Register button.

Submitted Data:

- Username: Budi Bangun
- Email: budibangun@gmail.com
- Password: Bud1Bangun!
- Re-Password: Bud1Bangun!
- Tempat Lahir: Jakarta
- Tanggal Lahir: 1980-05-02
- Clear Button:** A red button at the bottom.

D. Validasi Form

- Semua inputan tidak boleh kosong
- Panjang username minimal 5 karakter, terdiri dari huruf . Username tidak boleh mengandung angka, maupun symbol lainnya kecuali spasi.
- Email harus diisi dengan format yang benar
- Password harus terdiri dari minimal 8 karakter, minimal 1 huruf besar, minimal 1 huruf kecil, minimal 1 angka, dan minimal 1 simbol.
- Re-password harus sama dengan password yang dituliskan sebelumnya

- E. Tombol clear akan menghapus semua inputan di form maupun di tampilan “submitted data”