

Destian Ardan Alfatanu

H1D022045

Shift E

Laporan Responsi

1. Registrasi

The image displays three screenshots of a mobile application's registration page, titled 'Registrasi'.

Top Left Screenshot: Shows the initial state of the registration form. The title is 'Daftarkan Diri Anda Sekarang!' (Register Yourself Now!). Below the title is a subtitle 'Isi form di bawah ini untuk mendaftar' (Fill out the form below to register). The form contains three input fields: 'Nama' (Name), 'Email', and 'Password'. A 'Registrasi' button is located below the fields. At the bottom, there is a link 'Sudah punya akun? Login' (Already have an account? Login).

Top Right Screenshot: Shows the form filled with sample data. The 'Nama' field contains 'Fatan', the 'Email' field contains 'fatan@gmail.com', and the 'Password' field contains a masked password '*****'. The 'Registrasi' button is still present, and the 'Login' link remains at the bottom.

Bottom Screenshot: Shows the form with validation errors. The 'Nama' field has a red error message 'Nama harus diisi' (Name must be filled). The 'Email' field has a red error message 'Email harus diisi' (Email must be filled). The 'Password' field has a red error message 'Password harus diisi' (Password must be filled). The 'Registrasi' button is still present, and the 'Login' link remains at the bottom.

Gambar di kiri atas adalah tampilan awal dari Halaman Registrasi yang belum diisi dengan data diri. Gambar di kanan atas adalah Halaman Registrasi yang sudah diisi dengan data diri. Gambar di kiri bawah adalah

tampilan registrasi ketika kolomnya belum diisi tapi sudah klik tombol Registrasi. Setelah data diri telah sepenuhnya diisi langsung saja klik tombol Registrasi.

```
1 // Membuat Textbox nama
2 Widget _namaTextField() {
3   return TextFormField(
4     decoration: const InputDecoration(labelText: "Nama", labelStyle: TextStyle(color: Colors.black)),
5     keyboardType: TextInputType.text,
6     controller: _namaTextboxController,
7     style: const TextStyle(fontFamily: 'Georgia', color: Colors.black),
8     validator: (value) {
9       if (value!.isEmpty) {
10         return 'Nama harus diisi';
11       }
12       return null;
13     },
14   );
15 }
16
17 // Membuat Textbox email
18 Widget _emailTextField() {
19   return TextFormField(
20     decoration: const InputDecoration(labelText: "Email", labelStyle: TextStyle(color: Colors.black)),
21     keyboardType: TextInputType.emailAddress,
22     controller: _emailTextboxController,
23     style: const TextStyle(fontFamily: 'Georgia', color: Colors.black),
24     validator: (value) {
25       if (value!.isEmpty) {
26         return 'Email harus diisi';
27       }
28       return null;
29     },
30   );
31 }
32
33 // Membuat Textbox password
34 Widget _passwordTextField() {
35   return TextFormField(
36     decoration: const InputDecoration(labelText: "Password", labelStyle: TextStyle(color: Colors.black)),
37     keyboardType: TextInputType.text,
38     obscureText: true,
39     controller: _passwordTextboxController,
40     style: const TextStyle(fontFamily: 'Georgia', color: Colors.black),
41     validator: (value) {
42       if (value!.isEmpty) {
43         return "Password harus diisi";
44       }
45       return null;
46     },
47   );
48 }
```

Kode diatas berfungsi untuk menampilkan kolom inputan di Halaman Registrasi

```

1 // Membuat Tombol Registrasi
2 Widget _buttonRegistrasi() {
3   return ElevatedButton(
4     child: const Text("Registrasi", style: TextStyle(fontFamily: 'Georgia', color: Colors.white)), // Ubah warna tulisan tombol menjadi putih
5     onPressed: () {
6       var validate = _formKey.currentState!.validate();
7       if (validate) {
8         if (!_isLoading) _submit();
9       }
10    },
11    style: ElevatedButton.styleFrom(
12      backgroundColor: Colors.grey[700], // Warna tombol
13      padding: const EdgeInsets.symmetric(vertical: 12.0, horizontal: 20.0), // Padding untuk tombol
14    ),
15  );
16 }

```

Kode diatas berfungsi diatas membuat tombol Registrasi

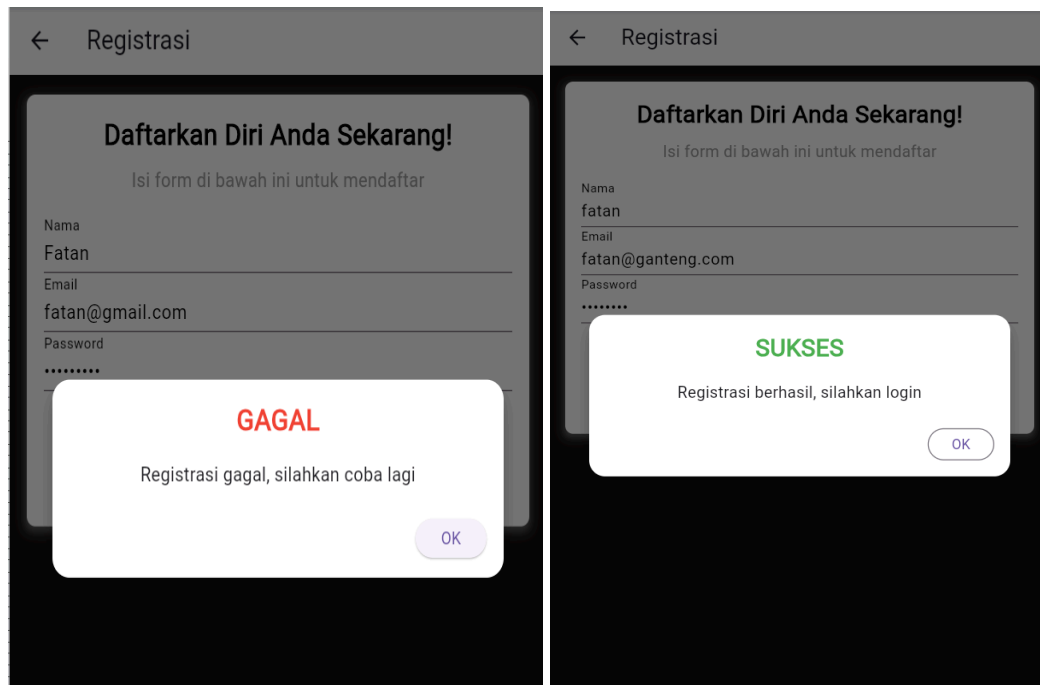
```

1 void _submit() {
2   _formKey.currentState!.save();
3   setState(() {
4     _isLoading = true;
5   });
6   RegistrasiBloc.registrasi(
7     nama: _namaTextboxController.text,
8     email: _emailTextboxController.text,
9     password: _passwordTextboxController.text,
10  ).then((value) async {
11    if (value.code == 200) {
12      // Tindakan setelah registrasi berhasil
13      Navigator.pop(context); // Kembali ke halaman login
14    } else {
15      showDialog(
16        context: context,
17        barrierDismissible: false,
18        builder: (BuildContext context) => const WarningDialog(
19          description: "Registrasi gagal, silahkan coba lagi",
20        ),
21      );
22    }
23  }, onError: (error) {
24    print(error);
25    showDialog(
26      context: context,
27      barrierDismissible: false,
28      builder: (BuildContext context) => const WarningDialog(
29        description: "Registrasi gagal, silahkan coba lagi",
30      ),
31    );
32  });
33 }

```

Kode diatas berfungsi ketika tombol Registrasi di klik maka akan memproses data yang telah diinputkan. Jika berhasil melakukan registrasi maka akan langsung

diarahkan ke halaman Login. Dan jika error maka akan muncul alert “Registrasi gagal, silahkan coba lagi” dan kembali ke halaman registrasi



Gambar di kiri adalah tampilan ketika kita gagal melakukan registrasi yang disebabkan oleh inputan yang tidak lengkap atau formatnya salah. Sedangkan Gambar di kanan adalah tampilan ketika kita telah berhasil melakukan registrasi yang selanjutnya diarahkan ke halaman login

2. Login

Login

Sudahkah Anda Stress Hari Ini?

Silahkan login untuk mengurangi stress Anda!

Email

Password

Login

Belum punya akun? [Registrasi](#)

Login

Sudahkah Anda Stress Hari Ini?

Silahkan login untuk mengurangi stress Anda!

Email

fatan@ganteng.com

Password

.....

Login

Belum punya akun? [Registrasi](#)

Login

Sudahkah Anda Stress Hari Ini?

Silahkan login untuk mengurangi stress Anda!

Email

fatan@gmail.com

Password

.....

Login

GAGAL

Login gagal, silahkan coba lagi

OK

Login

Sudahkah Anda Stress Hari Ini?

Silahkan login untuk mengurangi stress Anda!

Email

Email harus diisi

Password

Password harus diisi

Login

Belum punya akun? [Registrasi](#)

Gambar di kiri atas adalah tampilan ketika baru mengakses Halaman Login. Gambar di kanan atas adalah tampilan ketika kita mengisi form. Gambar di kanan bawah adalah tampilan ketika belum isi data dan pencet tombol login. Dan gambar di kiri bawah adalah tampilan ketika kita gagal melakukan login karena salah menginputkan data

```

1 // Membuat Textbox email
2 Widget _emailTextField() {
3   return TextFormField(
4     decoration: const InputDecoration(labelText: "Email", labelStyle: TextStyle(color: Colors.black)),
5     keyboardType: TextInputType.emailAddress,
6     controller: _emailTextEditingController,
7     style: const TextStyle(fontFamily: 'Georgia', color: Colors.black),
8     validator: (value) {
9       // Validasi harus diisi
10      if (value!.isEmpty) {
11        return 'Email harus diisi';
12      }
13      return null;
14    },
15  );
16 }
17
18 // Membuat Textbox password
19 Widget _passwordTextField() {
20   return TextFormField(
21     decoration: const InputDecoration(labelText: "Password", labelStyle: TextStyle(color: Colors.black)),
22     keyboardType: TextInputType.text,
23     obscureText: true,
24     controller: _passwordTextEditingController,
25     style: const TextStyle(fontFamily: 'Georgia', color: Colors.black),
26     validator: (value) {
27       // Validasi harus diisi
28       if (value!.isEmpty) {
29         return "Password harus diisi";
30       }
31       return null;
32     },
33   );
34 }

```

Kode diatas berfungsi untuk membuat kolom email dan password yang masing masing kolomnya mempunyai validator apakah kolomnya sudah terisi atau belum. Jika kolom email tidak diisi maka akan muncul pesan “Email harus diisi”, dan jika kolom password tidak diisi maka akan muncul pesan “Password harus diisi”.

```

1 // Membuat Tombol Login
2 Widget _buttonLogin() {
3   return ElevatedButton(
4     child: const Text("Login", style: TextStyle(fontFamily: 'Georgia', color: Colors.white)),
5     onPressed: () {
6       var validate = _formKey.currentState!.validate();
7       if (validate) {
8         if (!_isLoading) _submit();
9       }
10    },
11    style: ElevatedButton.styleFrom(
12      backgroundColor: Colors.grey[700], // Warna tombol
13    ),
14  );
15 }

```

Kode diatas berfungsi untuk membuat tombol Login yang jika dipencet maka akan terhubung ke kode di bawah ini

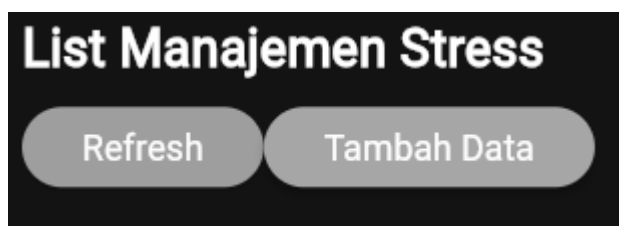
```

1 void _submit() {
2   _formKey.currentState!.save();
3   setState(() {
4     _isLoading = true;
5   });
6   LoginBloc.login(
7     email: _emailTextboxController.text,
8     password: _passwordTextboxController.text)
9   .then((value) async {
10    if (value.code == 200) {
11      await UserInfo().setToken(value.token.toString());
12      await UserInfo().setUserID(int.parse(value.userID.toString()));
13      Navigator.pushReplacement(context,
14        MaterialPageRoute(builder: (context) => const StressPage()));
15    } else {
16      showDialog(
17        context: context,
18        barrierDismissible: false,
19        builder: (BuildContext context) => const WarningDialog(
20          description: "Login gagal, silahkan coba lagi",
21        ));
22    }
23  }, onError: (error) {
24    print(error);
25    showDialog(
26      context: context,
27      barrierDismissible: false,
28      builder: (BuildContext context) => const WarningDialog(
29        description: "Login gagal, silahkan coba lagi",
30      ));
31  });
32 }

```

Kode di atas aktif ketika tombol Login di klik. Jika login berhasil dilakukan maka akan menuju ke halaman utama yaitu List Manajemen Stress. Namun jika login gagal dilakukan maka akan muncul alert “Login gagal, silahkan coba lagi”.

3. Create Data



Jika di klik Tambah Data maka akan mengarah ke halaman Edit

← Edit Stress

Stress Factor

Coping Strategy

Stress Level

0

Save Changes

← Edit Stress

Stress Factor

Server Down

Coping Strategy

Cuddling

Stress Level

1000

Save Changes

☰ Manajemen Stress

List Manajemen Stress

Refresh

Tambah Data

ID	Stress Factor	Coping Strategy	Stress Level
1	Workload	Meditation	2
2	Overheat	Healing	5
3	Server Down	Cuddling	1000

Gambar di kiri adalah tampilan ketika kita baru saja mengklik tombol Tambah Data, form yang tersedia masih kosong. Gambar di kanan adalah tampilan form ketika sudah diisi. Jika kita klik Save Changes maka akan tersimpan dan akan ditampilkan di halaman List Manajemen Stress. Kita juga akan diarahkan langsung ke halaman tersebut.


```

1 ElevatedButton(
2   onPressed: () {
3     // Arahkan ke halaman EditStressPage untuk menambah data baru
4     Navigator.push(
5       context,
6       MaterialPageRoute(
7         builder: (context) => EditStressPage(
8           stress: Stress(), // Buat objek Stress baru untuk ditambahkan
9         ),
10      ),
11    );
12  },
13  child: const Text('Tambah Data'),
14  style: ElevatedButton.styleFrom(
15    backgroundColor: Colors.grey,
16    foregroundColor: Colors.white,
17  ),
18 ),

```

kode diatas adalah tombol untuk menambahkan data yang jika di pencet maka akan masuk ke halaman edit tetapi tidak ada nilai yang tersimpan.

```

1 children: [
2   TextField(
3     controller: _stressFactorController,
4     decoration: const InputDecoration(labelText: 'Stress Factor', labelStyle: TextStyle(color: Colors.white)),
5     style: const TextStyle(color: Colors.white),
6   ),
7   TextField(
8     controller: _copingStrategyController,
9     decoration: const InputDecoration(labelText: 'Coping Strategy', labelStyle: TextStyle(color: Colors.white)),
10    style: const TextStyle(color: Colors.white),
11  ),
12  TextField(
13    controller: _stressLevelController,
14    decoration: const InputDecoration(labelText: 'Stress Level', labelStyle: TextStyle(color: Colors.white)),
15    keyboardType: TextInputType.number,
16    style: const TextStyle(color: Colors.white),
17  ),
18  const SizedBox(height: 20),
19  ElevatedButton(
20    onPressed: _saveChanges,
21    child: const Text('Save Changes'),
22    style: ElevatedButton.styleFrom(
23      backgroundColor: Colors.grey,
24      foregroundColor: Colors.white,
25    ),
26  ),
27 ],

```

Kode di atas berisi formulir kosong untuk mengisi inputan baru yang jika di klik “Save Changes” maka data yang telah diinputkan akan di tampilkan di Halaman List Data Stress

4. Read Data

☰

Manajemen Stress

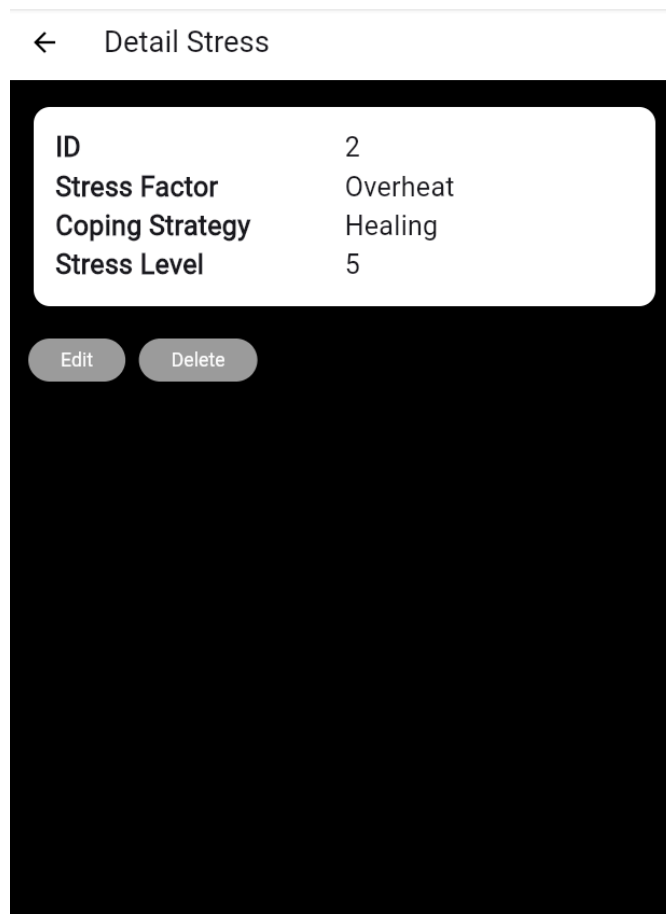
List Manajemen Stress

Refresh

Tambah Data

ID	Stress Factor	Coping Strategy	Stress Level
1	Workload	Meditation	2
2	Overheat	Healing	5

Ini adalah tampilan dari List Manajemen Stress. Jika data di klik maka akan dialihkan ke halaman detail.



Ini adalah Halaman Detail yang menampilkan isi salah satu data dengan detail dan terdapat tombol edit untuk menuju ke halaman edit serta tombol delete untuk menghapus data tersebut

```

1  child: Table(
2      children: [
3          TableRow(
4              children: [
5                  const Text('ID', style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold)),
6                  Text('${stress.id}', style: const TextStyle(fontSize: 20)),
7              ],
8          ),
9          TableRow(
10             children: [
11                 const Text('Stress Factor', style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold)),
12                 Text('${stress.stressFactor}', style: const TextStyle(fontSize: 20)),
13             ],
14         ),
15         TableRow(
16             children: [
17                 const Text('Coping Strategy', style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold)),
18                 Text('${stress.copingStrategy}', style: const TextStyle(fontSize: 20)),
19             ],
20         ),
21         TableRow(
22             children: [
23                 const Text('Stress Level', style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold)),
24                 Text('${stress.stressLevel}', style: const TextStyle(fontSize: 20)),
25             ],
26         ),
27     ],
28 ),

```

Kode diatas berfungsi untuk menampilkan semua data di halaman Detail. Data yang ditampilkan berbentuk tabel agar lebih mudah dibaca. Tabel tersebut berisikan 4 kolom.

5. Update Data

← Edit Stress

Stress Factor

Server Down

Coping Strategy

Healing

Stress Level

1

Save Changes

☰ Manajemen Stress

List Manajemen Stress

Refresh

Tambah Data

ID	Stress Factor	Coping Strategy	Stress Level
1	Workload	Meditation	2
3	Server Down	Healing	1

Gambar di kiri adalah pengisian kolom di halaman edit dan gambar di kanan adalah hasil dari inputan edit tersebut ketika sudah mengklik tombol save changes

```
1  children: [  
2      ElevatedButton(  
3          onPressed: () {  
4              Navigator.push(  
5                  context,  
6                  MaterialPageRoute(  
7                      builder: (context) => EditStressPage(stress: stress),  
8                  ),  
9              ).then((updatedStress) {  
10                 if (updatedStress != null) {  
11                     onUpdate(updatedStress); // Panggil fungsi untuk memperbarui data  
12                     Navigator.pop(context); // Kembali ke halaman sebelumnya  
13                 }  
14             });  
15         },  
16         child: const Text('Edit'),  
17         style: ElevatedButton.styleFrom(  
18             backgroundColor: Colors.grey,  
19             foregroundColor: Colors.white,  
20         ),  
21     ),  
22 ],
```

Kode diatas berfungsi sebagai kolom di halaman Edit dan dapat diisi dengan inputan yang nantinya inputan tersebut dapat tersimpan dan diperbarui di halaman List Manajemen Stress.

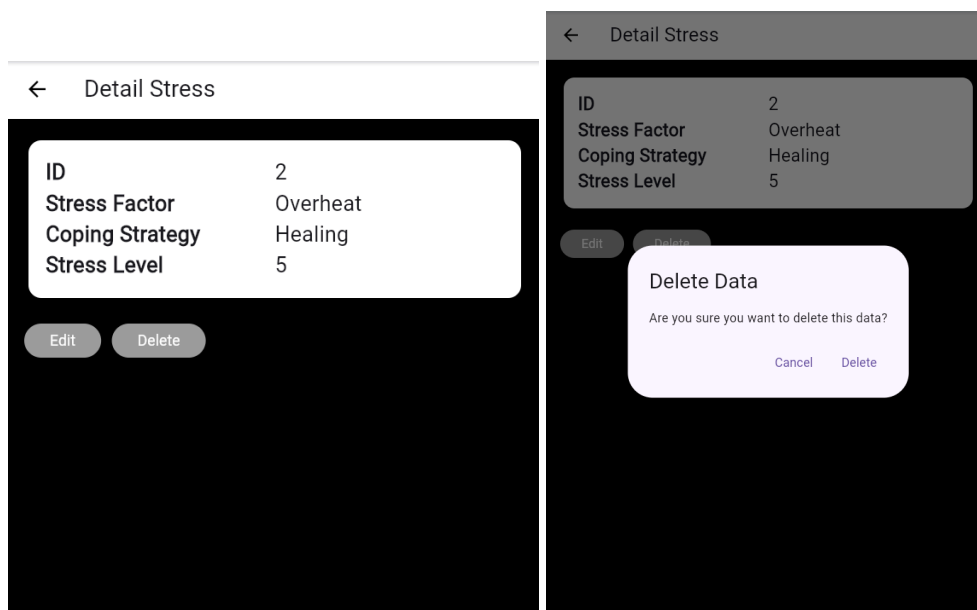
```

1 void _saveChanges() {
2   // Mengirimkan data yang telah diperbarui kembali ke halaman List Stress
3   Navigator.pop(
4     context,
5     Stress(
6       id: widget.stress.id,
7       stressFactor: _stressFactorController.text,
8       copingStrategy: _copingStrategyController.text,
9       stressLevel: int.tryParse(_stressLevelController.text),
10    ),
11  );
12 }

```

Kode diatas berguna untuk mengirimkan inputan data yang telah diisi sebelumnya dan kemudian ditampilkan di halaman List Manajemen Stress.

6. Delete Data



Ketika kita klik tombol Delete di halaman detail maka akan muncul alert seperti gambar di kanan atas. Dan jika kita klik Delete maka data tersebut akan langsung terhapus dan hilang dari tampilan List Manajemen seperti pada gambar di bawah ini

Manajemen Stress

List Manajemen Stress

Refresh

Tambah Data

ID	Stress Factor	Coping Strategy	Stress Level
1	Workload	Meditation	2
3	Server Down	Cuddling	1000

```

1  ElevatedButton(
2      onPressed: () {
3          _deleteStress(context);
4      },
5      child: const Text('Delete'),
6      style: ElevatedButton.styleFrom(
7          backgroundColor: Colors.grey,
8          foregroundColor: Colors.white,
9      ),
10 )

```

kode diatas membentuk sebuah tombol “DELETE” yang jika di klik akan menghapus data yang ada.

```

1 void _deleteStress(BuildContext context) {
2   // Konfirmasi penghapusan data
3   showDialog(
4     context: context,
5     builder: (BuildContext context) {
6       return AlertDialog(
7         title: const Text('Delete Data'),
8         content: const Text('Are you sure you want to delete this data?'),
9         actions: <Widget>[
10           TextButton(
11             onPressed: () {
12               Navigator.of(context).pop();
13             },
14             child: const Text('Cancel'),
15           ),
16           TextButton(
17             onPressed: () {
18               onDelete(stress);
19               Navigator.of(context).pop(); //
20               Navigator.pop(context); // Kembali ke halaman list stress setelah menghapus data
21             },
22             child: const Text('Delete'),
23           ),
24         ],
25       );
26     },
27   );
28 }

```

Untuk mengaktifasi tombol “DELETE” di atas, diperlukan kode agar ketika di klik mengakibatkan action yang kita inginkan.