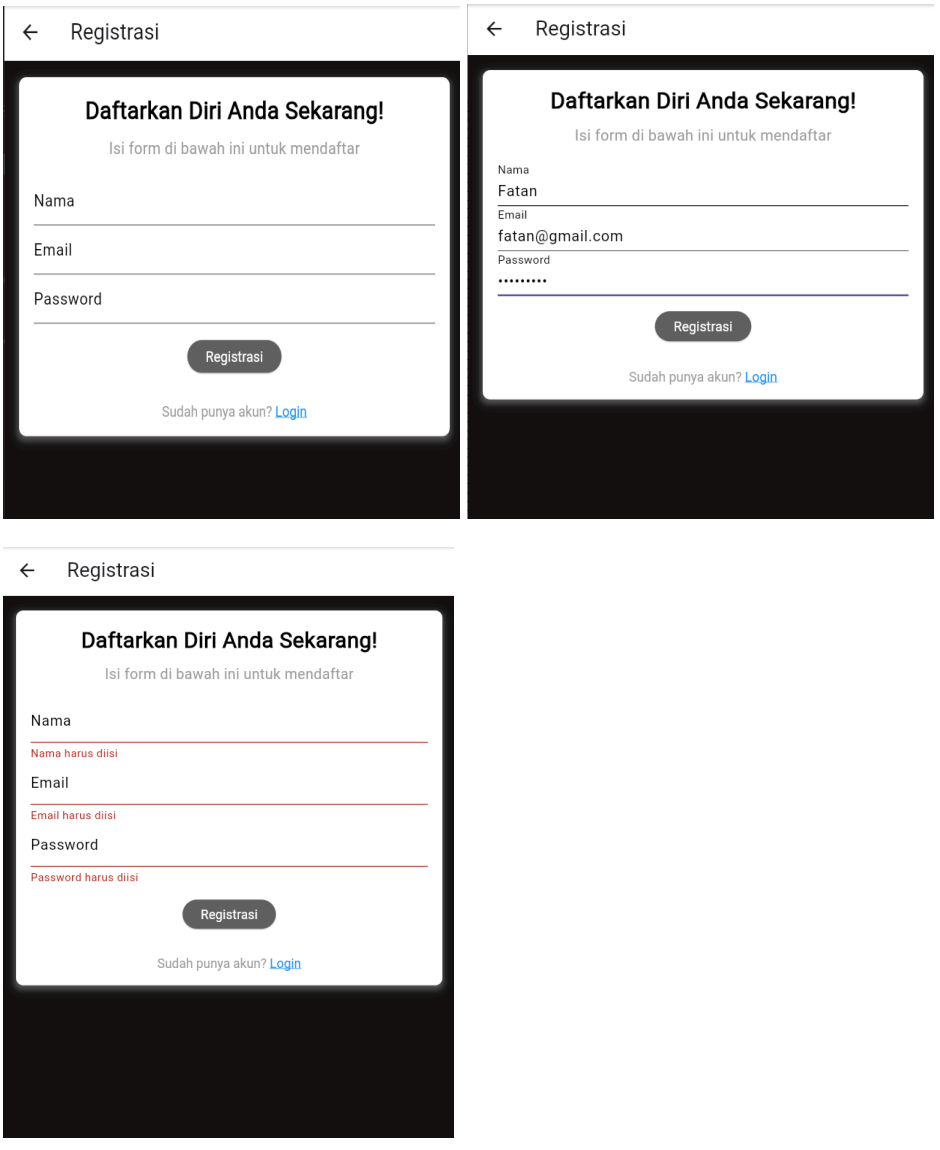
Destian Ardan Alfatanu

H1D022045

Shift E

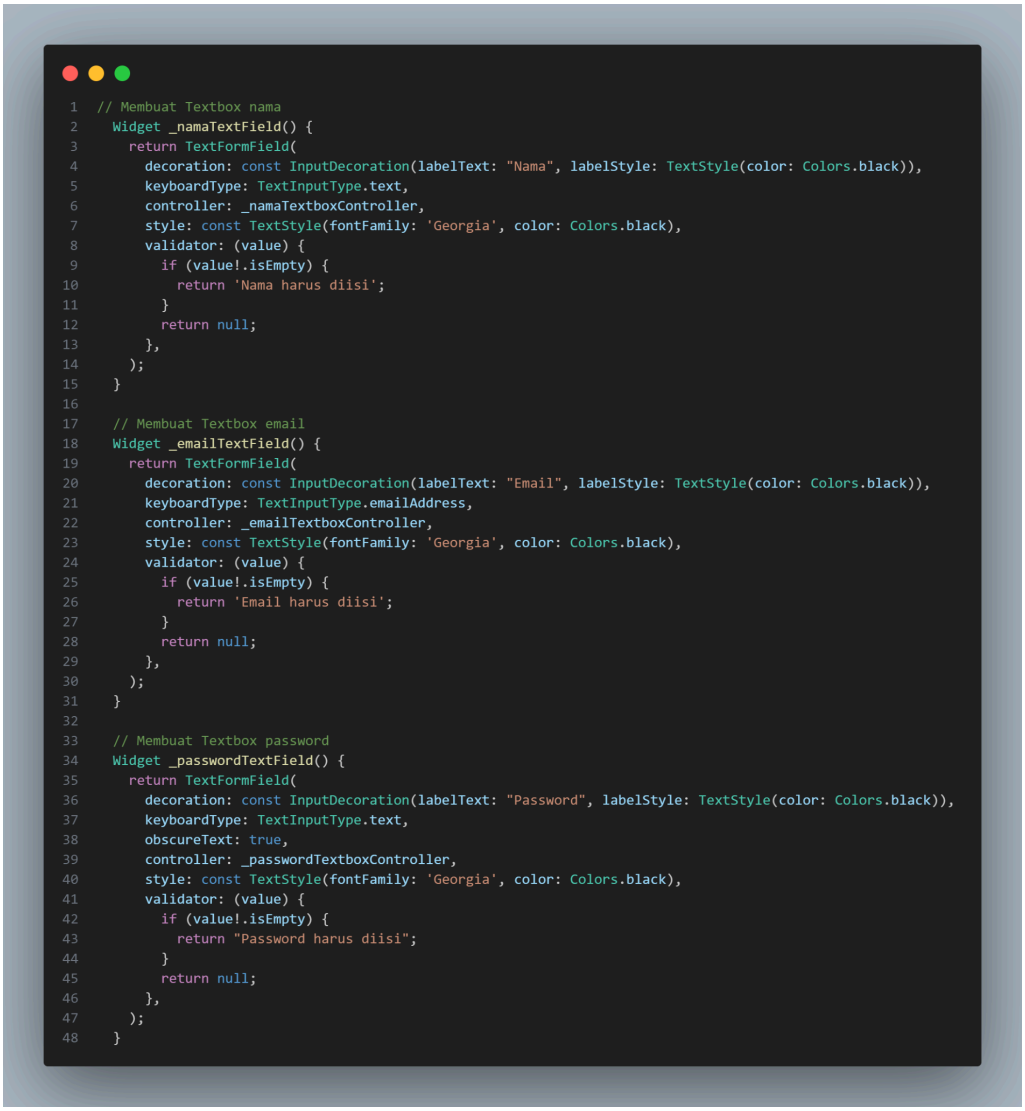
Laporan Responsi

1. Registrasi



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.



```dart
// Membuat Textbox nama
Widget _namaTextField() {
  return TextFormField(
    decoration: const InputDecoration(labelText: "Nama", labelStyle: TextStyle(color: Colors.black)),
    keyboardType: TextInputType.text,
    controller: _namaTextboxController,
    style: const TextStyle(fontFamily: 'Georgia', color: Colors.black),
    validator: (value) {
      if (value!.isEmpty) {
        return 'Nama harus diisi';
      }
      return null;
    },
  );
}

// Membuat Textbox email
Widget _emailTextField() {
  return TextFormField(
    decoration: const InputDecoration(labelText: "Email", labelStyle: TextStyle(color: Colors.black)),
    keyboardType: TextInputType.emailAddress,
    controller: _emailTextboxController,
    style: const TextStyle(fontFamily: 'Georgia', color: Colors.black),
    validator: (value) {
      if (value!.isEmpty) {
        return 'Email harus diisi';
      }
      return null;
    },
  );
}

// Membuat Textbox password
Widget _passwordTextField() {
  return TextFormField(
    decoration: const InputDecoration(labelText: "Password", labelStyle: TextStyle(color: Colors.black)),
    keyboardType: TextInputType.text,
    obscureText: true,
    controller: _passwordTextboxController,
    style: const TextStyle(fontFamily: 'Georgia', color: Colors.black),
    validator: (value) {
      if (value!.isEmpty) {
        return "Password harus diisi";
      }
      return null;
    },
  );
}
```

Kode diatas berfungsi untuk menampilkan kolom inputan di Halaman Registrasi

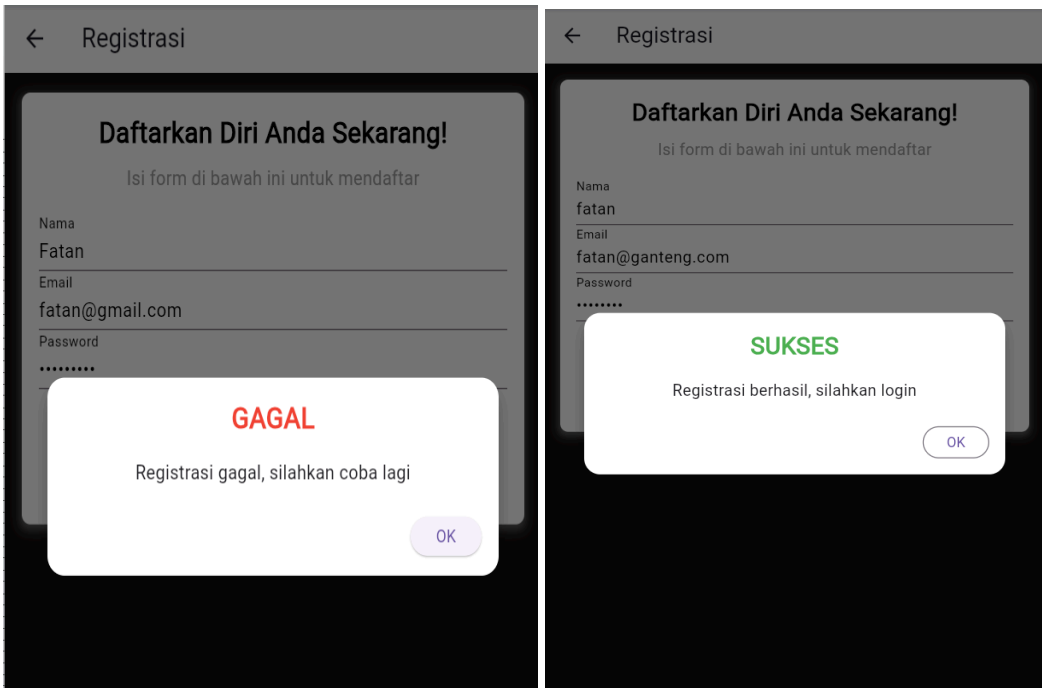
```dart
// Membuat Tombol Registrasi
  Widget _buttonRegistrasi() {
    return ElevatedButton(
      child: const Text("Registrasi", style: TextStyle(fontFamily: 'Georgia', color: Colors.white)), // Ubah warna tulisan tombol menjadi putih
      onPressed: () {
        var validate = _formKey.currentState!.validate();
        if (validate) {
          if (!_isLoading) _submit();
        }
      },
      style: ElevatedButton.styleFrom(
        backgroundColor: Colors.grey[700], // Warna tombol
        padding: const EdgeInsets.symmetric(vertical: 12.0, horizontal: 20.0), // Padding untuk tombol
      ),
    );
  }
```

Kode diatas berfungsi diatas membuat tombol Registrasi

```dart
void _submit() {
    _formKey.currentState!.save();
    setState(() {
      _isLoading = true;
    });
    RegistrasiBloc.registrasi(
      nama: _namaTextboxController.text,
      email: _emailTextboxController.text,
      password: _passwordTextboxController.text,
    ).then((value) async {
      if (value.code == 200) {
        // Tindakan setelah registrasi berhasil
        Navigator.pop(context); // Kembali ke halaman login
      } else {
        showDialog(
          context: context,
          barrierDismissible: false,
          builder: (BuildContext context) => const WarningDialog(
            description: "Registrasi gagal, silahkan coba lagi",
          ),
        );
      }
    }, onError: (error) {
      print(error);
      showDialog(
        context: context,
        barrierDismissible: false,
        builder: (BuildContext context) => const WarningDialog(
          description: "Registrasi gagal, silahkan coba lagi",
        ),
      );
    });
  }
```

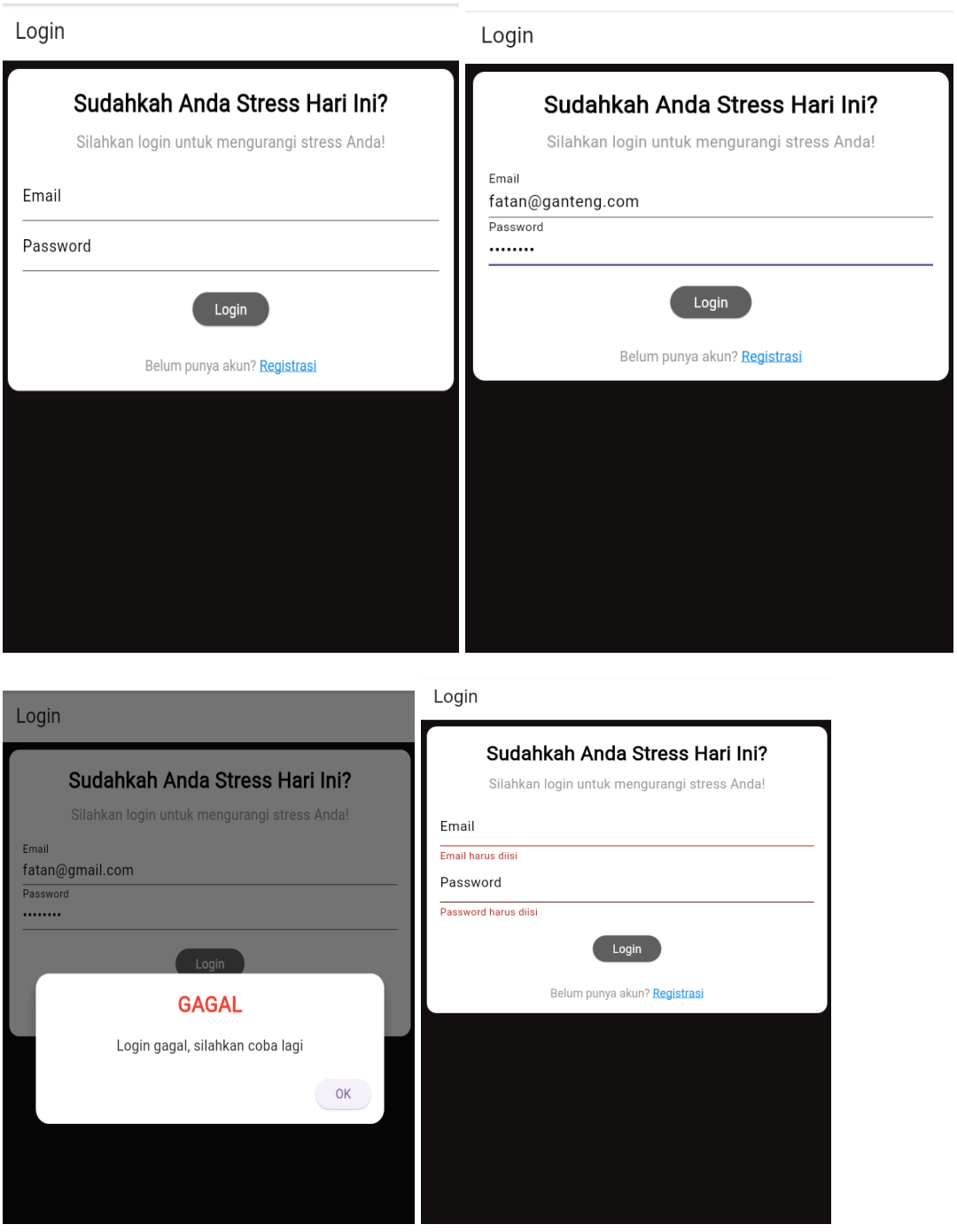
Kode diatas berfungsi ketika tombol Registrasi di klik maka akan memproses data yang telah diinputkan. Jika berhasil melakukan registrasi maka akan langsung

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



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

```dart
// Membuat Textbox email
  Widget _emailTextField() {
    return TextFormField(
      decoration: const InputDecoration(labelText: "Email", labelStyle: TextStyle(color: Colors.black)),
      keyboardType: TextInputType.emailAddress,
      controller: _emailTextboxController,
      style: const TextStyle(fontFamily: 'Georgia', color: Colors.black),
      validator: (value) {
        // Validasi harus diisi
        if (value!.isEmpty) {
          return 'Email harus diisi';
        }
        return null;
      },
    );
  }

  // Membuat Textbox password
  Widget _passwordTextField() {
    return TextFormField(
      decoration: const InputDecoration(labelText: "Password", labelStyle: TextStyle(color: Colors.black)),
      keyboardType: TextInputType.text,
      obscureText: true,
      controller: _passwordTextboxController,
      style: const TextStyle(fontFamily: 'Georgia', color: Colors.black),
      validator: (value) {
        // Validasi harus diisi
        if (value!.isEmpty) {
          return "Password harus diisi";
        }
        return null;
      },
    );
  }
```

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”.

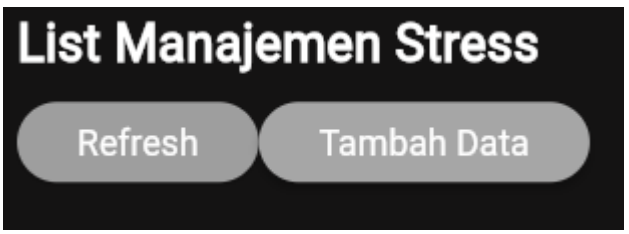
```dart
// Membuat Tombol Login
  Widget _buttonLogin() {
    return ElevatedButton(
      child: const Text("Login", style: TextStyle(fontFamily: 'Georgia', color: Colors.white)),
      onPressed: () {
        var validate = _formKey.currentState!.validate();
        if (validate) {
          if (!_isLoading) _submit();
        }
      },
      style: ElevatedButton.styleFrom(
        backgroundColor: Colors.grey[700], // Warna tombol
      ),
    );
  }
```

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

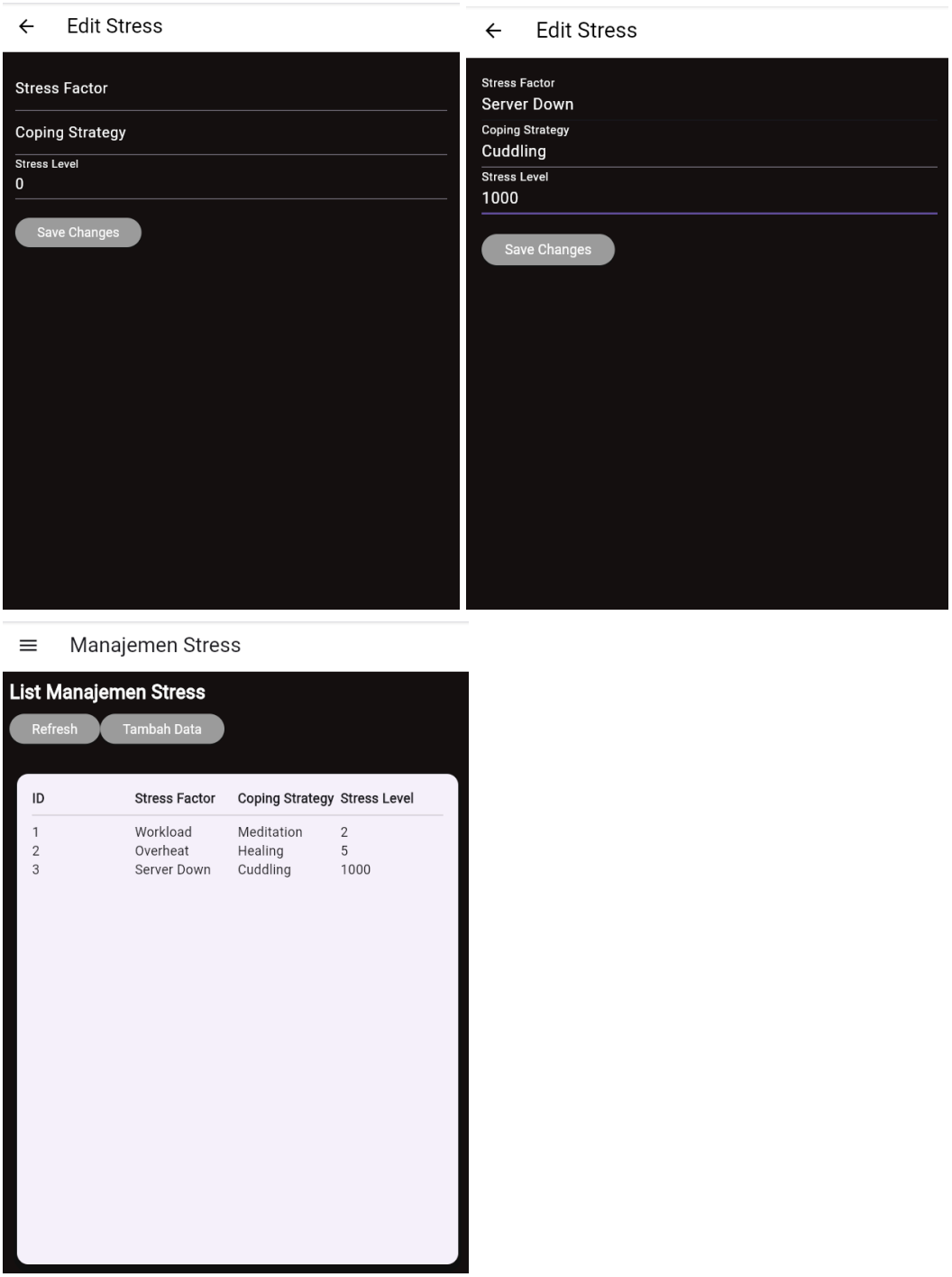
```dart
void _submit() {
    _formKey.currentState!.save();
    setState(() {
      _isLoading = true;
    });
    LoginBloc.login(
            email: _emailTextboxController.text,
            password: _passwordTextboxController.text)
        .then((value) async {
      if (value.code == 200) {
        await UserInfo().setToken(value.token.toString());
        await UserInfo().setUserID(int.parse(value.userID.toString()));
        Navigator.pushReplacement(context,
            MaterialPageRoute(builder: (context) => const StressPage()));
      } else {
        showDialog(
            context: context,
            barrierDismissible: false,
            builder: (BuildContext context) => const WarningDialog(
                  description: "Login gagal, silahkan coba lagi",
                ));
      }
    }, onError: (error) {
      print(error);
      showDialog(
          context: context,
          barrierDismissible: false,
          builder: (BuildContext context) => const WarningDialog(
                description: "Login gagal, silahkan coba lagi",
              ));
    });
  }
```

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



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.

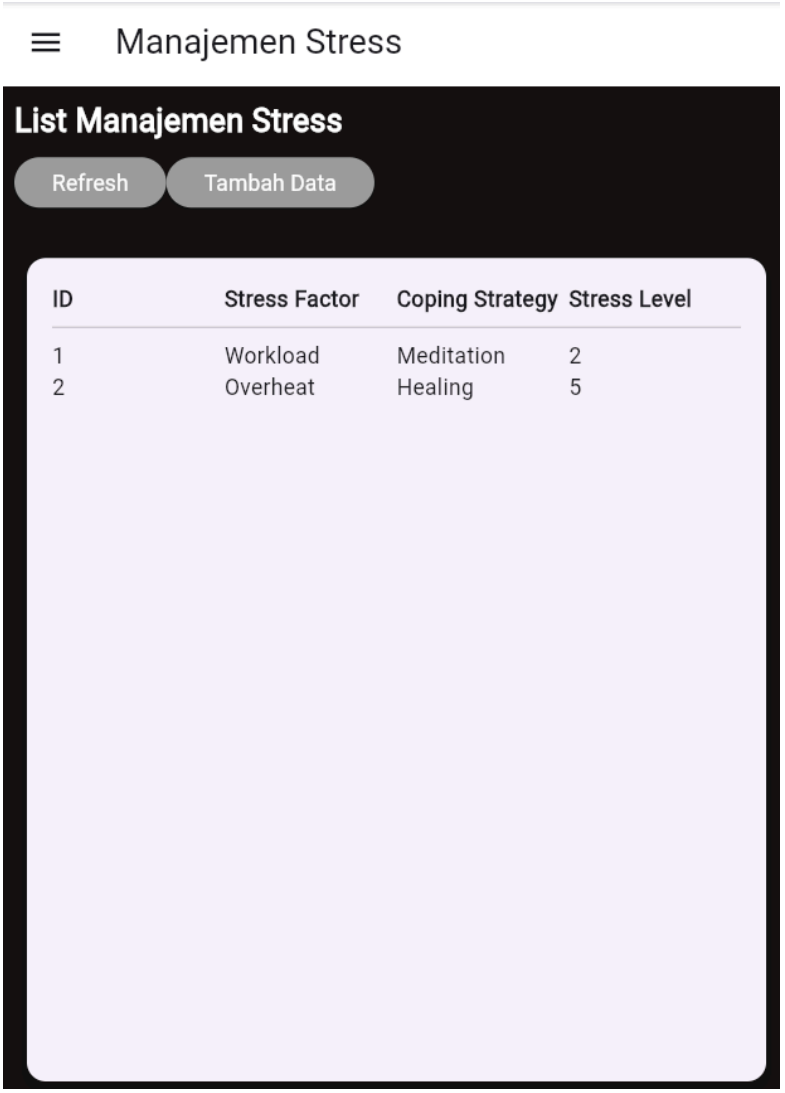
```
ElevatedButton(
                    onPressed: () {
                      // Arahkan ke halaman EditStressPage untuk menambah data baru
                      Navigator.push(
                        context,
                        MaterialPageRoute(
                          builder: (context) => EditStressPage(
                            stress: Stress(), // Buat objek Stress baru untuk ditambahkan
                          ),
                        ),
                      );
                    },
                    child: const Text('Tambah Data'),
                    style: ElevatedButton.styleFrom(
                      backgroundColor: Colors.grey,
                      foregroundColor: Colors.white,
                    ),
                  ),
```

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

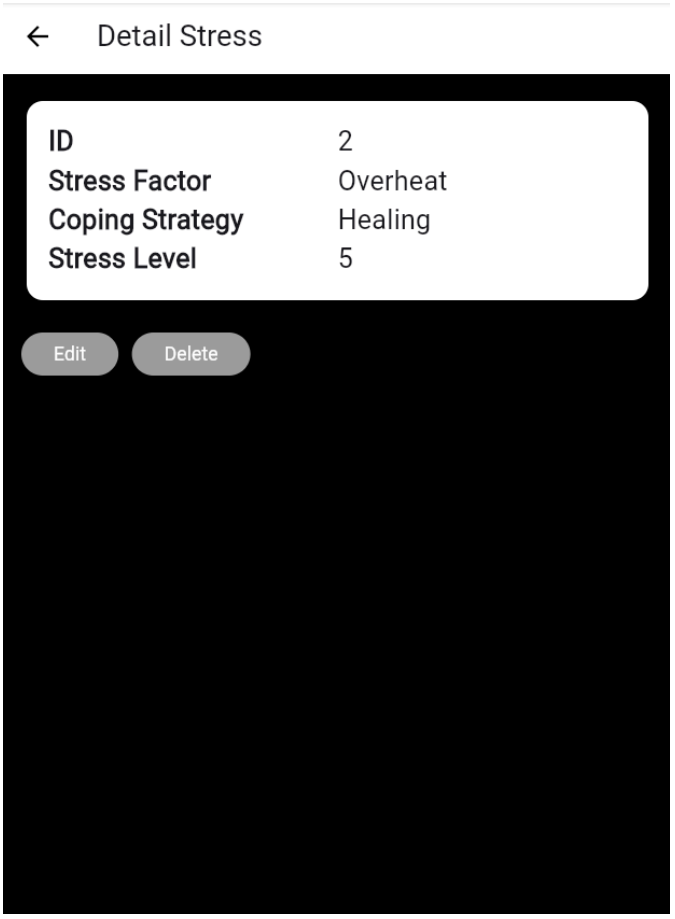
```
children: [
            TextField(
              controller: _stressFactorController,
              decoration: const InputDecoration(labelText: 'Stress Factor', labelStyle: TextStyle(color: Colors.white)),
              style: const TextStyle(color: Colors.white),
            ),
            TextField(
              controller: _copingStrategyController,
              decoration: const InputDecoration(labelText: 'Coping Strategy', labelStyle: TextStyle(color: Colors.white)),
              style: const TextStyle(color: Colors.white),
            ),
            TextField(
              controller: _stressLevelController,
              decoration: const InputDecoration(labelText: 'Stress Level', labelStyle: TextStyle(color: Colors.white)),
              keyboardType: TextInputType.number,
              style: const TextStyle(color: Colors.white),
            ),
            const SizedBox(height: 20),
            ElevatedButton(
              onPressed: _saveChanges,
              child: const Text('Save Changes'),
              style: ElevatedButton.styleFrom(
                backgroundColor: Colors.grey,
                foregroundColor: Colors.white,
              ),
            ),
          ],
```

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



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

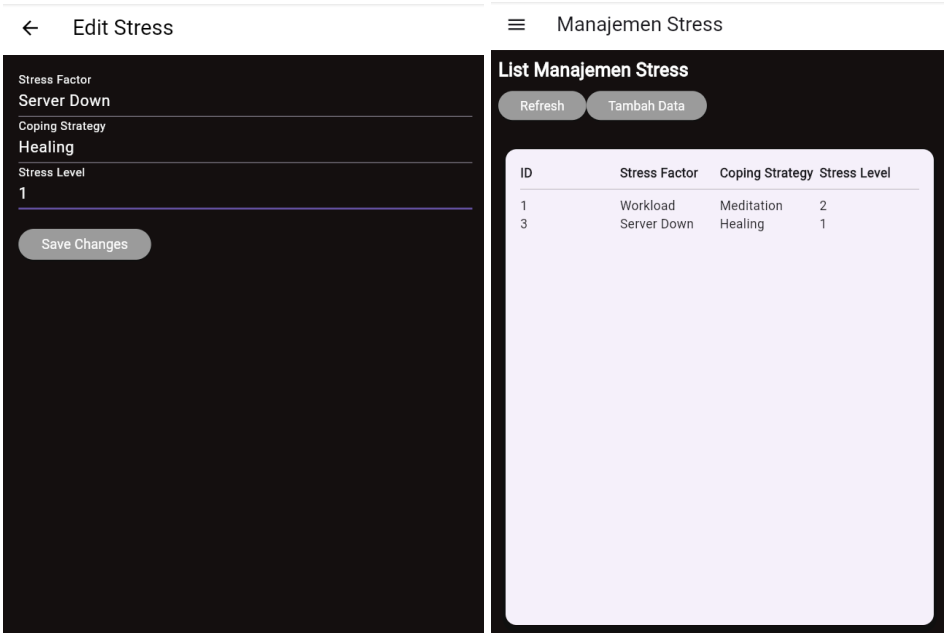


Ini adalah Halaman Detail yang menampilkan isi salah satu data dengan detail dan terdapat tombol edit untuk menuju ke halaman edit serta tombol delete untuk menghapus data tersebut

```
child: Table(
                    children: [
                      TableRow(
                        children: [
                          const Text('ID', style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold)),
                          Text('${stress.id}', style: const TextStyle(fontSize: 20)),
                        ],
                      ),
                      TableRow(
                        children: [
                          const Text('Stress Factor', style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold)),
                          Text('${stress.stressFactor}', style: const TextStyle(fontSize: 20)),
                        ],
                      ),
                      TableRow(
                        children: [
                          const Text('Coping Strategy', style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold)),
                          Text('${stress.copingStrategy}', style: const TextStyle(fontSize: 20)),
                        ],
                      ),
                      TableRow(
                        children: [
                          const Text('Stress Level', style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold)),
                          Text('${stress.stressLevel}', style: const TextStyle(fontSize: 20)),
                        ],
                      ),
                    ],
                  ),
```

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



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

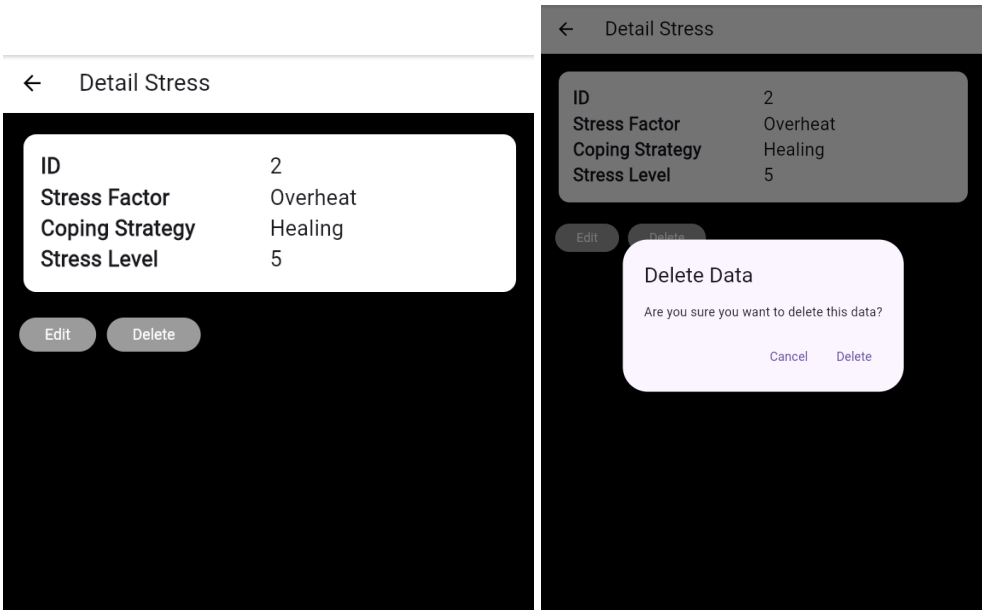
```
children: [
                ElevatedButton(
                  onPressed: () {
                    Navigator.push(
                      context,
                      MaterialPageRoute(
                        builder: (context) => EditStressPage(stress: stress),
                      ),
                    ).then((updatedStress) {
                      if (updatedStress != null) {
                        onUpdate(updatedStress); // Panggil fungsi untuk memperbarui data
                        Navigator.pop(context); // Kembali ke halaman sebelumnya
                      }
                    });
                  },
                  child: const Text('Edit'),
                  style: ElevatedButton.styleFrom(
                    backgroundColor: Colors.grey,
                    foregroundColor: Colors.white,
                  ),
                ),
              ],
```

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.

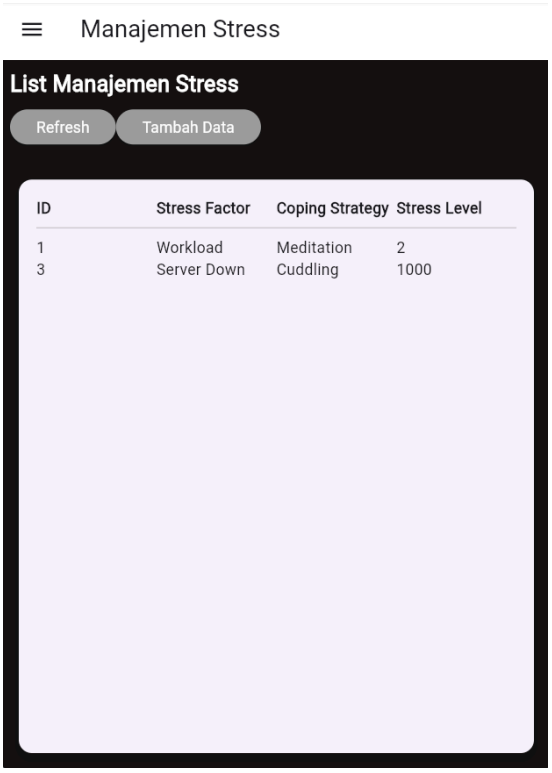
```dart
void _saveChanges() {
    // Mengirimkan data yang telah diperbarui kembali ke halaman List Stress
    Navigator.pop(
      context,
      Stress(
        id: widget.stress.id,
        stressFactor: _stressFactorController.text,
        copingStrategy: _copingStrategyController.text,
        stressLevel: int.tryParse(_stressLevelController.text),
      ),
    );
  }
```

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




```
ElevatedButton(
                    onPressed: () {
                      _deleteStress(context);
                    },
                    child: const Text('Delete'),
                    style: ElevatedButton.styleFrom(
                      backgroundColor: Colors.grey,
                      foregroundColor: Colors.white,
                    ),
                  ),
```

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

```dart
void _deleteStress(BuildContext context) {
    // Konfirmasi penghapusan data
    showDialog(
      context: context,
      builder: (BuildContext context) {
        return AlertDialog(
          title: const Text('Delete Data'),
          content: const Text('Are you sure you want to delete this data?'),
          actions: <Widget>[
            TextButton(
              onPressed: () {
                Navigator.of(context).pop();
              },
              child: const Text('Cancel'),
            ),
            TextButton(
              onPressed: () {
                onDelete(stress);
                Navigator.of(context).pop(); //
                Navigator.pop(context); // Kembali ke halaman list stress setelah menghapus data
              },
              child: const Text('Delete'),
            ),
          ],
        );
      },
    );
  }
```

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