A. Identitas Job Sheet

IDENTITAS JOB SHEET

Perguruan Tinggi	: Politeknik Negeri Bengkalis	Pertemuan ke	: 11
Jurusan/Program Studi	: Teknik Informatika / D4-KSI	Job ke	: 9
Nama Mata Kuliah	: Aplikasi Mobile Framework	Mulai Berlaku	: 2025

B. Komponen Job Sheet

1. Judul Job

JOB IX FORM DAN VALIDASI

2. Tujuan

Setelah mempelajari dan melakukan praktikum form dan validasi input, diharapkan mahasiswa :

a. Mampu merancang dan membuat form sebagai inputan aplikasi.
b. Mampu melakukan validasi input terhadap form.

3. Dasar Teori

Form adalah container opsional yang digunakan untuk mengelompokan komponen-komponen masukan dan pilihan. Di Flutter form di implementasikan dengan membuat objek dari class Form, biasanya digunakan jika data yang diinputkan membutuhkan proses validasi. Validasi inputan menjadi suatu hal yang sangat penting dalam proses penyimpanan data, agar data yang dimasukkan konsisten dan sesuai dengan yang diharapkan.

4. Referensi

a. Budi Raharjo, 2019, Pemrograman Android dengan Flutter, Informatika Bandung, Bandung
b. Dicoding, Belajar Membuat Aplikasi Flutter untuk Pemula, 2021
c. Button, https://docs.flutter.dev/, 2021

5. Alat dan Bahan

Alat yang digunakan pada praktikum ini adalah :

a. Perangkat keras atau *hardware* berupa 1 set PC Komputer atau Laptop dengan spesifikasi yang dianjurkan.
    f) CPU Core i3 atau lebih tinggi
    g) RAM 8 GB atau lebih tinggi
b. Perangkat lunak atau *software* yang digunakan antara lain :
    h) OS Windows 7 atau lebih tinggi
    i) Git Windows
    j) Visual Studio Code (VSCode)
    k) Dart Plugin VSCode
    l) DartPad (Editor Pemrograman Online)

Bahan pengujian pada praktikum ini ialah daftar potongan kode program yang harus di jalankan pada Dart Editor baik VS Code atau DartPad.

6. Keselamatan Kerja

a. Pastikan posisi PC atau laptop sudah benar
b. Hidupkan PC atau laptop mengikuti prosedur yang ada.
c. Jika terjadi sesuatu yang meragukan atau mencurigakan pada PC silahkan bertanya kepada Staf Laboran.
d. Jika praktikum sudah selesai, matikan PC atau laptop sesuai dengan prosedur yang ada.

7. Prosedur Kerja A.

Membuat Form

Pada bagian ini kita akan mencoba membuat form yang terdiri dari 4 isian data. Implementasinya kita akan membuat empat buah object dari class TextFormField. Class ini mirip dengan class TextField, namun disisi lain memiliki perbedaan seperti TextFormField hanya dapat digunakan pada class Form dan memiliki properti validator untuk melakukan validasi data yang diinputkan.

Pada saat menggunakan class Form kita membutuhkan objek dari kelas GlobalKey. Objek ini berfungsi untk membuat id unik dari form bersangkutan dan juga untuk melakukan validasi data pada form tersebut melalui objek dari kelas FormState. Objek dari kelass GlobalKey dibuat menggukanan kode berikut:

```
GlobalKey<FormState> formKey = GlobalKey<FormState>();
```

Dan contoh kode pembuatan form adalah seperti berikut:

```
body: Form(
      key: formKey,
      child: Column(children: <Widget>[
        TextFormField(
          decoration: const InputDecoration(
            hintText: 'Nim',
          ),
          keyboardType: TextInputType.text,
        ),
        TextFormField(
          decoration: const InputDecoration(
            hintText: 'Nama',
          ),
          keyboardType: TextInputType.text,
        ),
        TextFormField(
          decoration: const InputDecoration(
            hintText: 'Program Studi',
          ),
          keyboardType: TextInputType.text,
        ),
        TextFormField(
          decoration: const InputDecoration(
            hintText: 'Semester',
          ),
          keyboardType: TextInputType.number,
        ),
        ElevatedButton(child: const Text('Submit'), onPressed: () {}),
      ]),
    ),
```

Pada form di atas dapat kita lihat ada empat komponen TextFormField, kode dasar untuk membuat TextFormFielad adalah sebagai berikut:

```
        TextFormField(
          decoration: const InputDecoration(
            hintText: 'Semester',
          ),
          keyboardType: TextInputType.number,
        ),
        ElevatedButton(child: const Text('Submit'), onPressed: () {}),
    ]
```

Nilai property pada TextFormField dapat berbeda-beda antara satu field dengan field yang lainya, tergantung kebutuhan. Selain TextFormField kita juga menambahkan ElevatedButton, dengan kode seperti berikut:

```
ElevatedButton(
        child: const Text('Submit'),
        onPressed: validateInput,
),
```

Pada property onPressed kita isi dengan metode validateInput, yang artinya pada saat tombol ini di tekan oleh user maka ElevatedButton akan melakukan aksi yang terdapat pada metode validateInput.

Langkah-langkah pembuatan form adalah sebagai berikut:

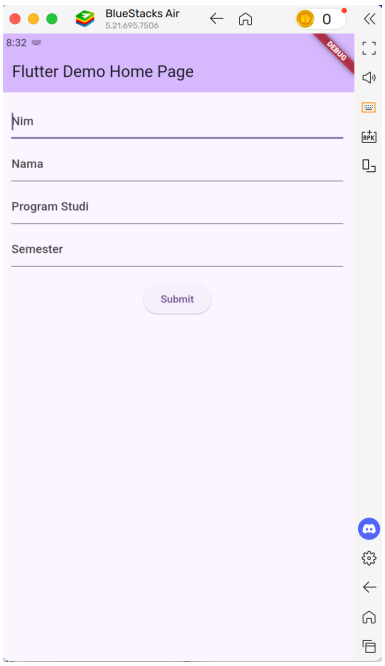
a. Buatlah project flutter baru dengan nama **form_validasi** di dalam folder P08_Form_Validasi. Caranya pilih menu View → Command Pallet atau tekan tombol Ctrl+Shif+P pada keyboard. Ketikan flutter dan pilih **Flutter: New Project**
b. Selanjutnya tentukan lokasi penyimpanan project Anda, pilih folder P08_Form_Validasi. Setelah memilih folder, kemudian pilih tombol "**Select a folder to create the project in**" untuk melanjutkan pembuatan project.
c. Selanjutnya akan muncul tampilan untuk memasukkan nama project. Masukkan nama project **form_validasi**. Setelah nama project diketik lalu tekan Enter di keyboard. Tunggu proses pembuatan project selesai, setelah proses selesai project Anda berhasil dibuat.

d. Editlah kode sumber main.dart yang berada pada folder lib, sehingga menjadi seperti berikut:

```
 import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor:
Colors.deepPurple),
        useMaterial3: true,
      ),
      home: const MyHomePage(title: 'Flutter Demo Home Page'),
    );
  }
}

class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key, required this.title});

  final String title;

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  GlobalKey<FormState> formKey = GlobalKey<FormState>();
  GlobalKey<ScaffoldState> scaffoldKey =
GlobalKey<ScaffoldState>();
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      key: scaffoldKey,
      appBar: AppBar(
        backgroundColor:
Theme.of(context).colorScheme.inversePrimary,
        title: Text(widget.title),
      ),
      body: Container(
        padding: const EdgeInsets.all(15.0),
        child: Form(
```

```
          key: formKey,
          child: Column(children: <Widget>[
            TextFormField(
              decoration: const InputDecoration(
                hintText: 'Nim',
              ),
              keyboardType: TextInputType.text,
            ),
            const SizedBox(
              height: 10.0,
            ),
            TextFormField(
              decoration: const InputDecoration(
                hintText: 'Nama',
              ),
              keyboardType: TextInputType.text,
            ),
            const SizedBox(
              height: 10.0,
            ),
            TextFormField(
              decoration: const InputDecoration(
                hintText: 'Program Studi',
              ),
              keyboardType: TextInputType.text,
            ),
            const SizedBox(
              height: 10.0,
            ),
            TextFormField(
              decoration: const InputDecoration(
                hintText: 'Semester',
              ),
              keyboardType: TextInputType.number,
            ),
            const SizedBox(
              height: 20.0,
            ),
            ElevatedButton(
              child: const Text('Submit'),
              onPressed: () {},
            ),
          ]),
        ),
      ),
    );
  }
}
```

e. Jalankan program di atas, tampilan program terlihat seperti di bawah ini:



B. Validasi Data

Proses validasi data di dalam form dilakukan dalam dua tahap, yaitu: a. Mengisi property validator pada widget TextFormField Langkah ini bertujuan untuk menentukan pesan kesalahan yang akan ditampilkan ketika text yang dimasukan ke dalam field tersebut salah. Berikut contoh untuk mengisi property validator:

```
…
    child: Form(
        key: formKey,
        child: Column(children: <Widget>[
          TextFormField(
            …
            keyboardType: TextInputType.text,
            validator: (String? value) {
              if (value!.isEmpty) {
                return 'Nim tidak boleh kosong';
              }
              return null;
            },
          ),
          const SizedBox(
            height: 10.0,
```

```
          ),
          TextFormField(
            …
            keyboardType: TextInputType.text,
            validator: (String? value) {
              if (value!.isEmpty) {
                return 'Nama tidak boleh kosong';
              }
              return null;
            },
          ),
          const SizedBox(
            height: 10.0,
          ),
          TextFormField(
            …
            keyboardType: TextInputType.text,
            validator: (String? value) {
              if (value!.isEmpty) {
                return 'Prodi tidak boleh kosong';
              }
              return null;
            },
          ),
          const SizedBox(
            height: 10.0,
          ),
          TextFormField(
            …
            keyboardType: TextInputType.number,
            validator: (String? value) {
              if (value!.isEmpty) {
                return 'Semester tidak boleh kosong';
              }
              return null;
            },
          ),
          const SizedBox(
            height: 20.0,
          ),
          ElevatedButton(
            child: const Text('Submit'),
            onPressed: validateInput,
          ),
        ]),
      ),
…
```

Pada kode di atas scenario validasi melakukan pengecekan field yang kosong, jika terdapat field yang kosong akan ditampilkan pesan kesalahan di bawah field tersebut.

b. Memanggil method validate() dari object FormState yang diambil melalui formKey.currentState()

Objek FormState adalah objek yang merepresentasikan keadaan form. Untuk mengambil keadaan aktif (saat ini) dari form, gunakan property currentState pada objek GlobalKey yang sudah dibuat sebelumnya, seperti berikut:

```
FormState? form = formKey.currentState;
```

Selanjutnya, proses validasi form dapat dilakukan dengan cara memanggil method validate() dari object FormState, seperti berikut:

```
if (form!.validate()) {
    ScaffoldMessenger.of(context).showSnackBar(snackBar);
}
```

Pada saat method validate() dipanggil, aplikasi secara otomatis akan mengeksekusi kode yang terdapat pada property validator di dalam setiap widget TextFormField.

Kita dapat memanggil method validate() ketika widget RaisedButton dipilih. Misalnya pada property onPressed di RaisedButton dipilih akan memanggil method validateInput(), maka kita harus membuat method tersebut. Dan didalamnya ada pemanggilan method validate(). Seperti kode berikut:

```
void validateInput() {
   FormState? form = formKey.currentState;
   const snackBar = SnackBar(      content: Text('Semua data
sudah tervalidasi'),
   );
   if (form!.validate()) {
     ScaffoldMessenger.of(context).showSnackBar(snackBar);
   }
 }
```

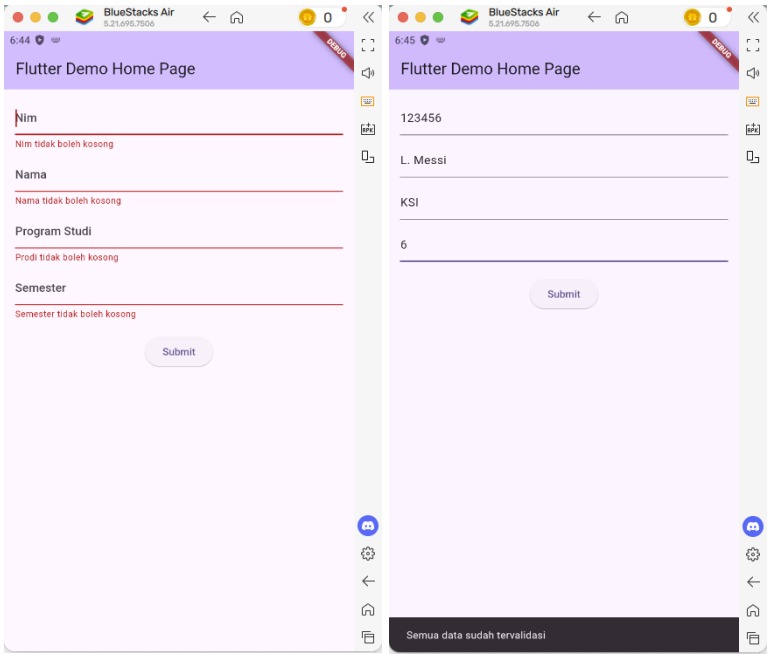
Sehingga kode lengkap proses validasinya menjadi seperti berikut ini:

```
class _MyHomePageState extends State<MyHomePage> {
  GlobalKey<FormState> formKey = GlobalKey<FormState>();
  GlobalKey<ScaffoldState> scaffoldKey = GlobalKey<ScaffoldState>();

  void validateInput() {
    FormState? form = formKey.currentState;

    const snackBar = SnackBar(
      content: Text('Semua data sudah tervalidasi'),
    );

    if (form!.validate()) {
      ScaffoldMessenger.of(context).showSnackBar(snackBar);
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      key: scaffoldKey,
      appBar: AppBar(
        backgroundColor: Theme.of(context).colorScheme.inversePrimary,
        title: Text(widget.title),
      ),
      body: Container(
        padding: const EdgeInsets.all(15.0),
        child: Form(
          key: formKey,
          child: Column(children: <Widget>[
            TextFormField(
              decoration: const InputDecoration(
                hintText: 'Nim',
              ),
              keyboardType: TextInputType.text,
              validator: (String? value) {
                if (value!.isEmpty) {
```

```
      return 'Nim tidak boleh kosong';
    }
    return null;
  },
),
const SizedBox(
  height: 10.0,
),
TextFormField(
  decoration: const InputDecoration(
    hintText: 'Nama',
  ),
  keyboardType: TextInputType.text,
  validator: (String? value) {
    if (value!.isEmpty) {
      return 'Nama tidak boleh kosong';
    }
    return null;
  },
),
const SizedBox(
  height: 10.0,
),
TextFormField(
  decoration: const InputDecoration(
    hintText: 'Program Studi',
  ),
  keyboardType: TextInputType.text,
  validator: (String? value) {
    if (value!.isEmpty) {
      return 'Prodi tidak boleh kosong';
    }
    return null;
  },
),
const SizedBox(
  height: 10.0,
```

```
            ),
            TextFormField(
              decoration: const InputDecoration(
                hintText: 'Semester',
              ),
              keyboardType: TextInputType.number,
              validator: (String? value) {
                if (value!.isEmpty) {
                  return 'Semester tidak boleh kosong';
                }
                return null;
              },
            ),
            const SizedBox(
              height: 20.0,
            ),
            ElevatedButton(
              child: const Text('Submit'),
              onPressed: validateInput,
            ),
          ]),
        ),
      ),
    );
  }
}
```

Simpan kode sumber di atas, dan jalankan aplikasinya. Lakukan uji coba dengan menginputkan data dan sebagian isian atau semuanya kosong.



Gambar 9. 2 Form dan Validasi Input
Sumber : Data Olahan, 2021

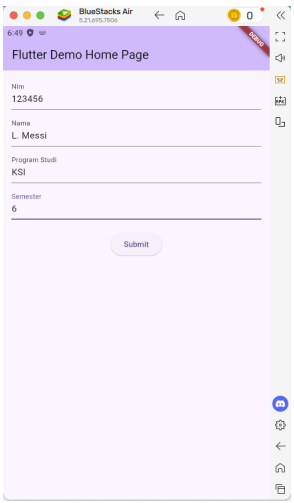
C. Mengatur Gaya Tampilan Pada TextFormField

Secara default widget TextFormField akan ditampilkan dengan gaya garis bawah. Seperti pada widget TextField, pada TextFormField juga dapat mengatur gaya tampilkan dengan menambahkan property decoration. Property ini diisi dengan nilai berupa objek dari kelas InputDecoration. Mari kita coba menambahkan label pada TextFormField.

```
…
TextFormField(
    decoration: const InputDecoration(
      hintText: 'Nim',
      labelText: 'Nim',
    ),
    …
  ),
  …
  TextFormField(
    decoration: const InputDecoration(
      hintText: 'Nama',
      labelText: 'Nama',
```

```
        ),
        …
      ),
      …
      TextFormField(
        decoration: const InputDecoration(
          hintText: 'Program Studi',
          labelText: ‘Program Studi’,
        ),
        …
      ),
      …
      TextFormField(
        Decoration: const InputDecoration(
          hintText: 'Semester',
          labelText: 'Semester',
        ),
        …
      ),
    …
```

Simpan kode sumber di atas, dan jalankan aplikasinya. Hasilnya terlihat seperti berikut:

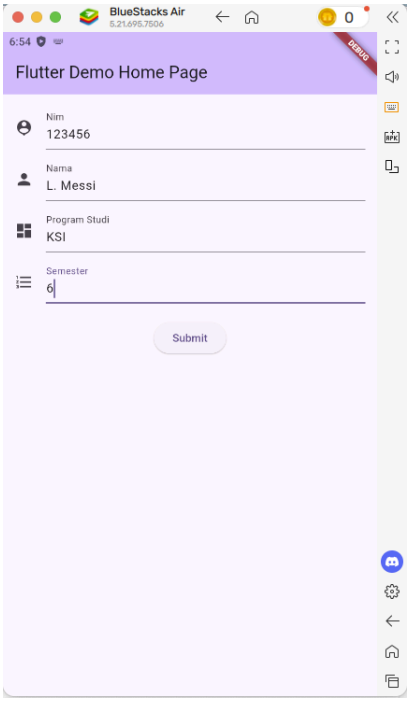


Gambar 9. 3 Menambahkan Label Text
Sumber : Data Olahan, 2021

Selanjutnya kita tambahkan icon pada field.

```
        …
        TextFormField(
          decoration: const InputDecoration(
            hintText: 'Nim',
            labelText: 'Nim',
            icon: Icon(Icons.person_pin),
          ),
          …
        ),
        …
        TextFormField(
          decoration: const InputDecoration(
            hintText: 'Nama',
            labelText: 'Nama',
            icon: Icon(Icons.person),
          ),
          …
        ),
        …
        TextFormField(
          decoration: const InputDecoration(
            hintText: 'Program Studi',
            labelText: 'Program Studi',
            icon: Icon(Icons.dashboard),
          ),
          …
        ),
        …
        TextFormField(
          decoration: const InputDecoration(
           hintText: 'Semester',
           labelText: 'Semester',
           icon: Icon(Icons.format_list_numbered),
          ),
          …
        ),
…
```

Simpan kembali kode sumber di atas, dan jalankan aplikasinya. Hasilnya terlihat seperti berikut:

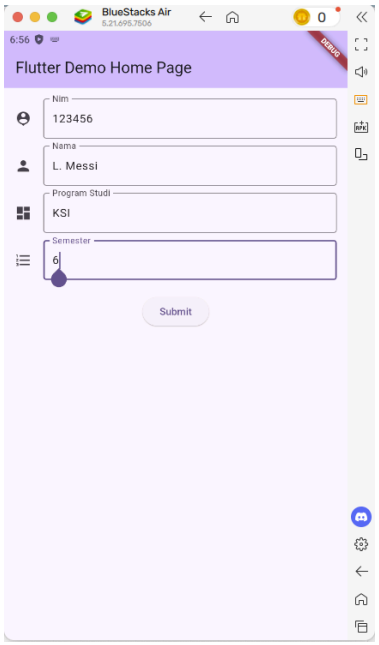


Selanjutnya kita menambahkan border pada field dengan menambahkan property border di object InputDecoration. Seperti terlihat pada kode berikut:

```
…
TextFormField(
   decoration: const InputDecoration(
     …
   border: OutlineInputBorder(),
   ),
   …
 ),
 …
 TextFormField(
   decoration: const InputDecoration(
     …
   border: OutlineInputBorder(),
   ),
```

```
      …
    ),
    …
    TextFormField(
      decoration: const InputDecoration(
        …
      border: OutlineInputBorder(),
      ),
      …
    ),
    …
    TextFormField(
      decoration: const InputDecoration(
        …
      border: OutlineInputBorder(),
      ),
      …
    ),
  …
```

Simpan kembali kode sumber di atas, Hasilnya terlihat seperti berikut ini:



Gambar 9. 5 Menambahkan Outline Input Border

Sumber : Data Olahan, 2021

Hasil akhir kode program adalah seperti berikut:

```
class _MyHomePageState extends State<MyHomePage> {
  GlobalKey<FormState> formKey = GlobalKey<FormState>();
  GlobalKey<ScaffoldState> scaffoldKey = GlobalKey<ScaffoldState>();

  void validateInput() {
    FormState? form = formKey.currentState;

    const snackBar = SnackBar(
      content: Text('Semua data sudah tervalidasi'),
    );

    if (form!.validate()) {
      ScaffoldMessenger.of(context).showSnackBar(snackBar);
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      key: scaffoldKey,
      appBar: AppBar(
        title: Text(widget.title),
        backgroundColor: Theme.of(context).colorScheme.inversePrimary,
      ),
      body: Container(
        padding: const EdgeInsets.all(15.0),
        child: Form(
          key: formKey,
          child: Column(children: <Widget>[
            TextFormField(
              decoration: const InputDecoration(
                hintText: 'Nim',
                labelText: 'Nim',
                icon: Icon(Icons.person_pin),
                border: OutlineInputBorder(),
              ),
              keyboardType: TextInputType.text,
              validator: (String? value) {
                if (value!.isEmpty) {
                  return 'Nim tidak boleh kosong';
                }
                return null;
              },
            ),
            const SizedBox(
              height: 10.0,
            ),
            TextFormField(
              decoration: const InputDecoration(
```

```
    hintText: 'Nama',
    labelText: 'Nama',
    icon: Icon(Icons.person),
    border: OutlineInputBorder(),
  ),
  keyboardType: TextInputType.text,
  validator: (String? value) {
    if (value!.isEmpty) {
      return 'Nama tidak boleh kosong';
    }
    return null;
  },
),
const SizedBox(
  height: 10.0,
),
TextFormField(
  decoration: const InputDecoration(
    hintText: 'Program Studi',
    labelText: 'Program Studi',
    icon: Icon(Icons.dashboard),
    border: OutlineInputBorder(),
  ),
  keyboardType: TextInputType.text,
  validator: (String? value) {
    if (value!.isEmpty) {
      return 'Prodi tidak boleh kosong';
    }
    return null;
  },
),
const SizedBox(
  height: 10.0,
),
TextFormField(
  decoration: const InputDecoration(
    hintText: 'Semester',
    labelText: 'Semester',
    icon: Icon(Icons.format_list_numbered),
    border: OutlineInputBorder(),
  ),
  keyboardType: TextInputType.number,
  validator: (String? value) {
    if (value!.isEmpty) {
      return 'Semester tidak boleh kosong';
    }
    return null;
  },
),
const SizedBox(
  height: 20.0,
```

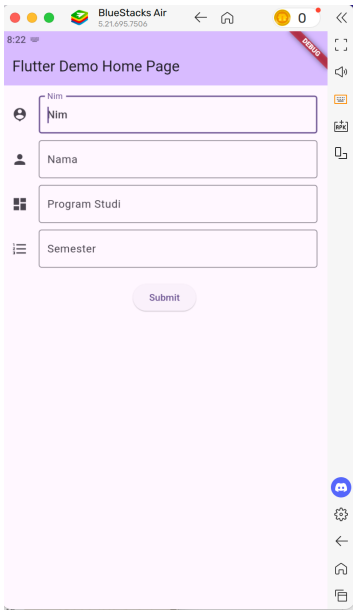
```
            ),
            ElevatedButton(
              child: const Text('Submit'),
              onPressed: validateInput,
            ),
          ]),
        ),
      ),
    );
  }
}
```

D. Mengatur Fokus pada TextFormField Ketika pada sebuah TextFormField siap menerima inputan, kondisi seperti ini dinamakan fokus. Biasanya focus terjadi pada saat user melakukan tap atau menekan pada TextFormField. a. Menggunakan property autofocus

Untuk memberikan focus pada TextFormField cukup mudah dengan menambahkan property autofocus: true, seperti yang terlihat pada code berikut:

```
TextFormField(
  //...
  autofocus: true,
),
```

Silahkan tambahkan autofocus pada field Nim. Dan restart aplikasinya, kita akan melihat field NIM secara otomatis menjadi focus.



Gambar 9. 6 Auto Focus di TextFormField NIM
Sumber : Data Olahan, 2021

b. Menggunakan property focusNode

Cara ini digunakan untuk mengatur focus berdasarkan suatu kejadian (event). Cara menggunakanya dengan cara menambahkan property focusNode, isi property tersebut berupa object dari class FocusNode. Object ini bisanya dibuat pada method iniState() dan di hapus dari memori di dalam method dispose(). Untuk menggunakanya kita dapat mendeklrasikan FocusNode

```
FocusNode myFocusNode;
```

Lalu membuat object di dalam method initState(), tambahkan kode berikut:

```
@override
void iniState(){
  Super.initState();
  myFocusNode = FocusNode();
}
```

Lalu menghapusnya kembali didalam method dispose(), tambahkan kode berikut:

```
@override
void
dispose(){
  myFocusNode.dispose();
  super.dispose();
}
```

Selanjutnya kaitkan objek myFocusNode pada TextFormField yang diinginkan untuk merima focus ketika event terjadi.

Contohnya seperti dibawah:

```
TextFormField(    focusNode: myFocusNode,
  //...
),
```

Untuk mengaktifkan focus pada widget TextFormField, gunakan kode seperti berikut:

```
ElevatedButton(
onPressed: () {
      FocusScope.of(context).requestFocus(myFocusNode);
   },
   //...
}
```

Sekarang silahkan coba pada project sebelumnya dengan menambahkan satu ElevatedButton lagi. Apabila tombol ini dipilih maka focus berada di field Nama. Sehingga potongan code dalam class _HomeState menjadi seperti berikut, khususnya pada bagian code yang bercetak tebal:

```
class _MyHomePageState extends State<MyHomePage> {
  GlobalKey<FormState> formKey = GlobalKey<FormState>();
  GlobalKey<ScaffoldState> scaffoldKey = GlobalKey<ScaffoldState>();
  late FocusNode myFocusNode;

  @override
  void initState() {
    super.initState();
    myFocusNode = FocusNode();
  }

  @override
```

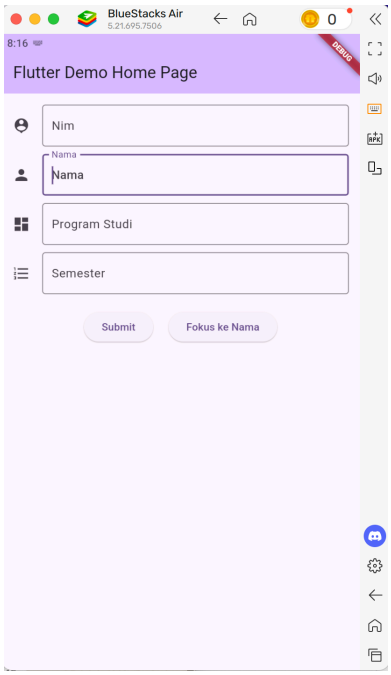
```
void dispose() {
  myFocusNode.dispose();
  super.dispose();
}

void validateInput() {
  FormState? form = formKey.currentState;

  const snackBar = SnackBar(
    content: Text('Semua data sudah tervalidasi'),
  );

  if (form!.validate()) {
    ScaffoldMessenger.of(context).showSnackBar(snackBar);
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    key: scaffoldKey,
    appBar: AppBar(
      backgroundColor: Theme.of(context).colorScheme.inversePrimary,
      title: Text(widget.title),
    ),
    body: Container(
      padding: const EdgeInsets.all(15.0),
      child: Form(
        key: formKey,
        child: Column(children: <Widget>[
          TextFormField(
            decoration: const InputDecoration(
              hintText: 'Nim',
              labelText: 'Nim',
              icon: Icon(Icons.person_pin),
              border: OutlineInputBorder(),
            ),
```

```
  keyboardType: TextInputType.text,
  validator: (String? value) {
    if (value!.isEmpty) {
      return 'Nim tidak boleh kosong';
    }
    return null;
  },
  autofocus: true,
),
const SizedBox(
  height: 10.0,
),
TextFormField(
  decoration: const InputDecoration(
    hintText: 'Nama',
    labelText: 'Nama',
    icon: Icon(Icons.person),
    border: OutlineInputBorder(),
  ),
  keyboardType: TextInputType.text,
  validator: (String? value) {
    if (value!.isEmpty) {
      return 'Nama tidak boleh kosong';
    }
    return null;
  },
  focusNode: myFocusNode,
),
const SizedBox(
  height: 10.0,
),
TextFormField(
  decoration: const InputDecoration(
    hintText: 'Program Studi',
    labelText: 'Program Studi',
    icon: Icon(Icons.dashboard),
    border: OutlineInputBorder(),
```

```
  ),
  keyboardType: TextInputType.text,
  validator: (String? value) {
    if (value!.isEmpty) {
      return 'Prodi tidak boleh kosong';
    }
    return null;
  },
),
const SizedBox(
  height: 10.0,
),
TextFormField(
  decoration: const InputDecoration(
    hintText: 'Semester',
    labelText: 'Semester',
    icon: Icon(Icons.format_list_numbered),
    border: OutlineInputBorder(),
  ),
  keyboardType: TextInputType.number,
  validator: (String? value) {
    if (value!.isEmpty) {
      return 'Semester tidak boleh kosong';
    }
    return null;
  },
),
const SizedBox(
  height: 20.0,
),
Row(
  mainAxisAlignment: MainAxisAlignment.center,
  children: <Widget>[
    ElevatedButton(
      child: const Text('Submit'),
      onPressed: validateInput,
    ),
```

```
            Container(
              width: 20,
            ),
            ElevatedButton(
              child: const Text('Fokus ke Nama'),
              onPressed: () {
                FocusScope.of(context).requestFocus(myFocusNode);
              },
            )
          ],
        ),
      ]),
    ),
  ),
 );
 }
}
```

Simpan perubahan pada kode sumber di atas. Hasilnya menjadi seperti berikut:



Gambar 9. 7 Menambahkan Focus Node di TextFormField Nama
Sumber : Data Olahan, 2021

E. Menangani Perubahan Nilai Pada TextFormField

Mendeteksi perubahan nilai pada TextFormField, terkadang dibutuhkan untuk kasus teretentu. Misalnya ketika kita ingin membuat fitur autocomplete, perlu dilakukan pembaharuan list berdasarkan text yang ketikan pada field. Nah jika di widget TextField terdapat properti onChanged untuk menangani masalah ini, namun fitur ini tidak tersedia di TextFormField. Oleh karena itu, kita harus menggunakan TextEditingController. Langkah-langkah yang diperlukan adalah sebagai berikut:

a. Membuat objek dari kelas TextEditingController.

```
final prodiController = TextEditingController();
```

b. Mengaitkan controller dengan TextFormField, dengan cara mengisi poperti controller dengan objek TextEditingController yang telah dibuat.

```
TextFormField(
  controller: prodiController,
  // ...
}
```

c. Membuat fungsi (method) yang akan digunakan untuk menangani kejadian perubahan teks.

```
void printValue() {
   print("Teks pada field Program Studi:
${prodiController.text}");
 }
```

d. Mengaitkan antara controller dan fungsi printValue() menggunakan addListener.

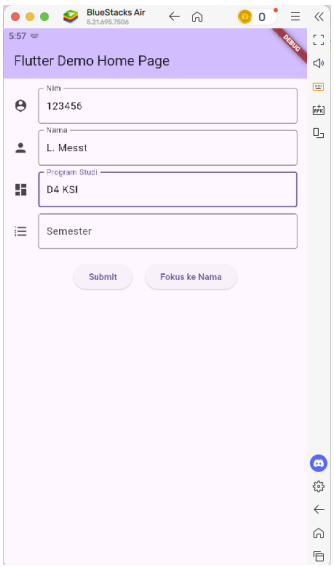
```
@override
 void initState() {
   super.initState();
   myFocusNode = FocusNode();
   prodiController.addListener(printValue);
 }
```

e. Berikut contoh potongan kode menangani perubahan nilai pada TextFormField program studi, fokus pada kode yang bercetak tebal:

```
class _MyHomePageState extends State<MyHomePage> {
  GlobalKey<FormState> formKey = GlobalKey<FormState>();
  GlobalKey<ScaffoldState> scaffoldKey = GlobalKey<ScaffoldState>();
late FocusNode myFocusNode;
  final prodiController = TextEditingController();

  void printValue() {
    print("Teks pada field Program Studi: ${prodiController.text}");
  }

  @override
  void initState() {
    super.initState();
    myFocusNode = FocusNode();
    prodiController.addListener(printValue);
  }

  …
 @override
    Widget build(BuildContext context) {
    return Scaffold(
      key: scaffoldKey,
      appBar: AppBar(
        backgroundColor: Theme.of(context).colorScheme.inversePrimary,
        title: Text(widget.title),
      ),
      body: Container(
        padding: const EdgeInsets.all(15.0),
        child: Form(
          key: formKey,
          child: Column(children: <Widget>[
            …
            TextFormField(
              controller: prodiController,
              decoration: const InputDecoration(
                hintText: 'Program Studi',
```

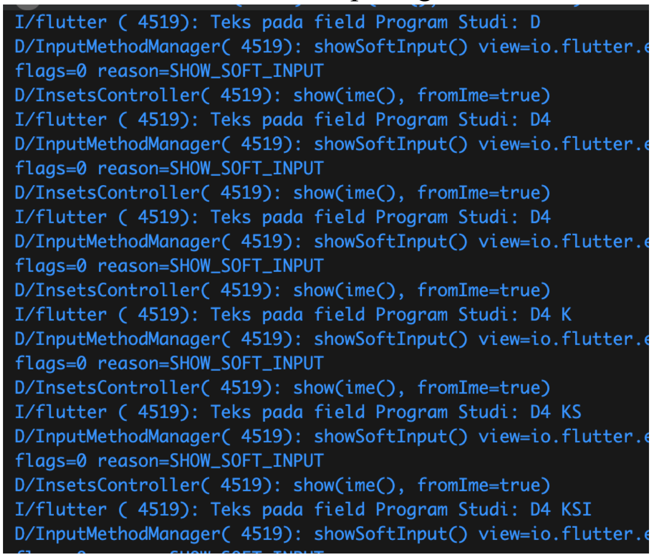
```
              labelText: 'Program Studi',
              icon: Icon(Icons.dashboard),
              border: OutlineInputBorder(),
            ),
            keyboardType: TextInputType.text,
            validator: (String? value) {
              if (value!.isEmpty) {
                return 'Prodi tidak boleh kosong';
              }
              return null;
            },
          ),
            …
          ],
        ),
      ]),
    ),
  ),
  );
 }
}
```

f. Simpan kode sumber di atas, lalu restart aplikasinya, dan ketikan teks pada field Program Studi:



Gambar 9. 8 Menambahkan Penanganan Nilai Pada TextFormField Prodi
Sumber : Data Olahan, 2021

g. Pada console akan terlihat seperti gambar berikut:



Gambar 9. 9 Hasil Perekaman Nilai Input di TextFormField Prodi
Sumber : Data Olahan, 2021

F. Mengambil Nilai Pada TextFormField

Text atau data yang diiputkan pada TextFormField dapat diambil menggunakan property text pada objek dari class TextEditingController. Pada kesempatan kali ini, kita akan mengambil text atau data dari semua field pada form. Masih di class _HomeState buatlah objek dari class TextEditingController untuk field nim, nama dan semester.

```
…
final prodiController = TextEditingController();
final nimController = TextEditingController();
final namaController = TextEditingController();
final semesterController = TextEditingController();
…
```

Selanjutnya kaitkan setiap controller tersebut ke TextFormField dengan mengisi properti controller dengan nama objek controller pada kode di atas. Kode nya terlihat pada text yang bercetak tebal di bawah ini:

```
Form(
key:
formKey,
  child: Column(children: <Widget>[
TextFormField(
    controller: nimController,
    // ...
  ),
  const
SizedBox(
height: 10.0,
  ),
  TextFormField(
    controller: namaController,
    // ...
  ),
  const
SizedBox(
height: 10.0,
  ),
  TextFormField(
    controller: prodiController,
    // ...
  ),
  const
SizedBox(
height: 10.0,
  ),
  TextFormField(
    controller: semesterController,
    // ...
  ),
 // ...
 ]),
)
```

Buatlah method showData() di dalam class _HomeState untuk menampilkan data ke dalam alert dialog, kode nya seperti berikut:

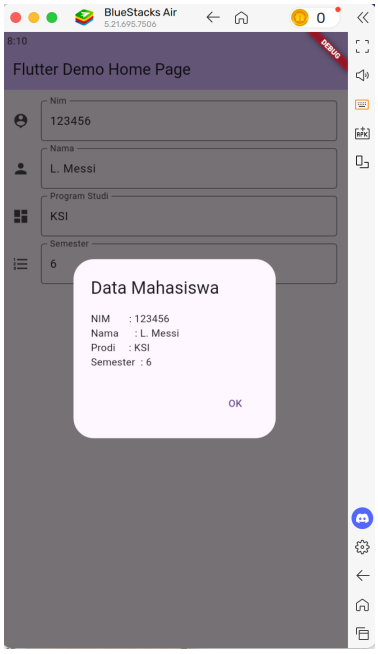
```
void showData() {
  showDialog(
    context: context,
    builder: (BuildContext context) {
      return AlertDialog(
        title: const Text('Data Mahasiswa'),
        content: Text(
          'NIM       : ${nimController.text}\n'
          'Nama      : ${namaController.text}\n'
          'Prodi     : ${prodiController.text}\n'
          'Semester  : ${semesterController.text}',
        ),
        actions: [
          TextButton(
            onPressed: () {
              Navigator.of(context).pop(); // Menutup dialog
            },
            child: const Text('OK'),
          ),
        ],
      );
    },
  );
}
```

Tambahkan kode bercetak tebal berikut pada method validateInput(), fungsinya untuk memanggil showData().

```
void validateInput() {
  FormState? form = formKey.currentState;

  const snackBar = SnackBar(
    content: Text('Semua data sudah tervalidasi'),
  );

  if (form!.validate()) {
    ScaffoldMessenger.of(context).showSnackBar(snackBar);
    showData();
```

```
  }
}
```

Simpan kode sumber di atas, jalankan ulang aplikasinya. Lalu inputkan data pada field lalu submit. Maka akan terlihat seperti gambar di bawah ini:



Gambar 9. 10 Menampilkan Data Input TextFormField
Sumber : Data Olahan, 2021

Hasil akhir kode adalah seperti berikut:

```
class _MyHomePageState extends State<MyHomePage> {
  GlobalKey<FormState> formKey = GlobalKey<FormState>();
  GlobalKey<ScaffoldState> scaffoldKey = GlobalKey<ScaffoldState>();
  late FocusNode myFocusNode;
  final prodiController = TextEditingController();
  final nimController = TextEditingController();
  final namaController = TextEditingController();
  final semesterController = TextEditingController();

  void printValue() {
```

```
  print("Teks pada field Program Studi: ${prodiController.text}");
}

void showData() {
  showDialog(
    context: context,
    builder: (BuildContext context) {
      return AlertDialog(
        title: const Text('Data Mahasiswa'),
        content: Text(
          'NIM       : ${nimController.text}\n'
          'Nama      : ${namaController.text}\n'
          'Prodi     : ${prodiController.text}\n'
          'Semester  : ${semesterController.text}',
        ),
        actions: [
          TextButton(
            onPressed: () {
              Navigator.of(context).pop(); // Menutup dialog
            },
            child: const Text('OK'),
          ),
        ],
      );
    },
  );
}

@override
void initState() {
```

```
  super.initState();
  myFocusNode = FocusNode();
  prodiController.addListener(printValue);
}

void validateInput() {
  FormState? form = formKey.currentState;

  const snackBar = SnackBar(
    content: Text('Semua data sudah tervalidasi'),
  );

  if (form!.validate()) {
    ScaffoldMessenger.of(context).showSnackBar(snackBar);
    showData();
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    key: scaffoldKey,
    appBar: AppBar(
      backgroundColor: Theme.of(context).colorScheme.inversePrimary,
      title: Text(widget.title),
    ),
    body: Container(
      padding: const EdgeInsets.all(15.0),
      child: Form(
        key: formKey,
```

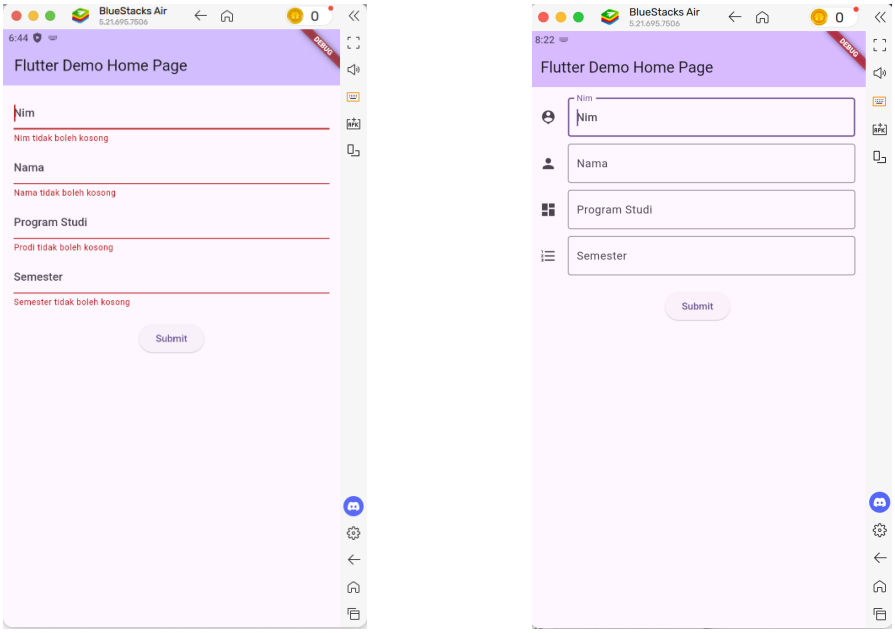
```
child: Column(children: <Widget>[
  TextFormField(
    controller: nimController,
    decoration: const InputDecoration(
      hintText: 'Nim',
      labelText: 'Nim',
      icon: Icon(Icons.person_pin),
      border: OutlineInputBorder(),
    ),
    keyboardType: TextInputType.text,
    validator: (String? value) {
      if (value!.isEmpty) {
        return 'Nim tidak boleh kosong';
      }
      return null;
    },
    autofocus: true,
  ),
  const SizedBox(
    height: 10.0,
  ),
  TextFormField(
    controller: namaController,
    decoration: const InputDecoration(
      hintText: 'Nama',
      labelText: 'Nama',
      icon: Icon(Icons.person),
      border: OutlineInputBorder(),
    ),
    keyboardType: TextInputType.text,
```

```
  validator: (String? value) {
    if (value!.isEmpty) {
      return 'Nama tidak boleh kosong';
    }
    return null;
  },
  focusNode: myFocusNode,
),
const SizedBox(
  height: 10.0,
),
TextFormField(
  controller: prodiController,
  decoration: const InputDecoration(
    hintText: 'Program Studi',
    labelText: 'Program Studi',
    icon: Icon(Icons.dashboard),
    border: OutlineInputBorder(),
  ),
  keyboardType: TextInputType.text,
  validator: (String? value) {
    if (value!.isEmpty) {
      return 'Prodi tidak boleh kosong';
    }
    return null;
  },
),
const SizedBox(
  height: 10.0,
),
```

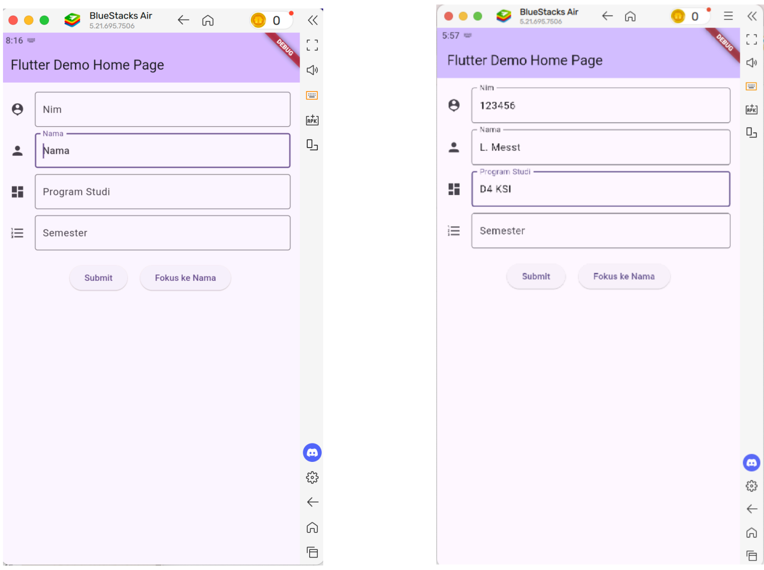
```
TextFormField(
  controller: semesterController,
  decoration: const InputDecoration(
    hintText: 'Semester',
    labelText: 'Semester',
    icon: Icon(Icons.format_list_numbered),
    border: OutlineInputBorder(),
  ),
  keyboardType: TextInputType.number,
  validator: (String? value) {
    if (value!.isEmpty) {
      return 'Semester tidak boleh kosong';
    }
    return null;
  },
),
const SizedBox(
  height: 20.0,
),
Row(
  mainAxisAlignment: MainAxisAlignment.center,
  children: <Widget>[
    ElevatedButton(
      child: const Text('Submit'),
      onPressed: validateInput,
    ),
    Container(
      width: 20,
    ),
    ElevatedButton(
```

```
                child: const Text('Fokus ke Nama'),
                onPressed: () {
                  FocusScope.of(context).requestFocus(myFocusNode);
                },
              )
            ],
          ),
        ]),
      ),
    ),
  );
 }
}
```

8. Gambar Kerja



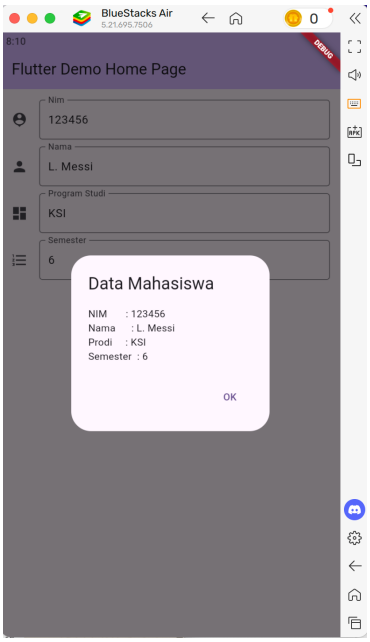
Gambar 9. 11 Penerapan Validasi Input (Kiri) dan Auto Focus (Kanan) Sumber : Data Olahan, 2021



Gambar 9. 12 Penerapan Node Focus (Kiri) dan Penangan Nilai Input (Kanan)
Sumber : Data Olahan, 2021

```
I/flutter ( 4519): Teks pada field Program Studi: D
D/InputMethodManager( 4519): showSoftInput() view=io.flutter.e
flags=0 reason=SHOW_SOFT_INPUT
D/InsetsController( 4519): show(ime(), fromIme=true)
I/flutter ( 4519): Teks pada field Program Studi: D4
D/InputMethodManager( 4519): showSoftInput() view=io.flutter.e
flags=0 reason=SHOW_SOFT_INPUT
D/InsetsController( 4519): show(ime(), fromIme=true)
I/flutter ( 4519): Teks pada field Program Studi: D4
D/InputMethodManager( 4519): showSoftInput() view=io.flutter.e
flags=0 reason=SHOW_SOFT_INPUT
D/InsetsController( 4519): show(ime(), fromIme=true)
I/flutter ( 4519): Teks pada field Program Studi: D4 K
D/InputMethodManager( 4519): showSoftInput() view=io.flutter.e
flags=0 reason=SHOW_SOFT_INPUT
D/InsetsController( 4519): show(ime(), fromIme=true)
I/flutter ( 4519): Teks pada field Program Studi: D4 KS
D/InputMethodManager( 4519): showSoftInput() view=io.flutter.e
flags=0 reason=SHOW_SOFT_INPUT
D/InsetsController( 4519): show(ime(), fromIme=true)
I/flutter ( 4519): Teks pada field Program Studi: D4 KSI
D/InputMethodManager( 4519): showSoftInput() view=io.flutter.e
```

Gambar 9. 13 Mengambil Nilai Input dari TextFormField Prodi
Sumber : Data Olahan, 2021



Gambar 9. 14 Menampilkan Data Input TextFormField
Sumber : Data Olahan, 2021

9. Data Pengujian

Data pengujian form pendaftaran akun pengguna adalah sebagai berikut:

Field	**Kriteria**	**Contoh Input**
Username	Tidak boleh kosong, Minimal 4 hingga 25 Karakter, Tidak Boleh Angka	Wikan
Email	Tidak boleh kosong, Minimal 4 hingga 25 Karakter, Harus ada simbol @	wikan@gmail.com
Nomor_HP	Tidak boleh kosong, Minimal 10 hingga 13 Karakter, Harus Angka	081234567890
Password	Tidak boleh kosong, Minimal 4 hingga 25 Karakter	xtrede

Confirm Password	Tidak boleh kosong, Minimal 4 hingga 25 Karakter	xtrede
Role	Berupa pilihan yang terdiri dari User atau Admin	0 → User 1 → Admin

10. Analisa dan Pembahasan

Berdasarkan data pengujian diatas apa saja kondisi yang harus diperiksa agar memenuhi kriteria validasi input.

11. Kesimpulan

Dari percobaan-percobaan yang telah dilakukan dapat diambil kesimpulan sebagai berikut:

a. Kita dapat menggunakan kelas Form untuk membuat form aplikasi.
b. Untuk menggunakan auto focus pada textformfield dapat dilakukan dengan mengatur nilai properti autofocus:true atau menggunakan node fokus.
c. Untuk menggunakan validasi kita dapat menambahkan property validator pada TextFormField.
d. Kita bisa menambahkan kriteria validasi tambahan pada properti validator.

C. Evaluasi

Kerjakan latihan berikut untuk menambah pemahaman anda mengenai materi praktikum yang telah dilakukan:

1. Berdasarkan data pengujian diatas rancang dan buatlah formnya.
2. Tambahkan juga validasi pada setiap Input form.
3. Lakukan uji coba input data yang sesuai kriteria dan tidak.