BAB III
AUTHENTICATION & PROVIDER

Pembelajaran yang akan dilakukan pada bab ini merupakan pembelajaran yang berkaitan dengan keamanan sebuah aplikasi. Praktikum kali ini akan memanfaatkan *platform* pihak ke-3, yakni ***firebase*** untuk mengimplementasikan sistem keamanan terhadap sebuah aplikasi. Selain itu, para praktikan juga akan diajarkan tentang bagaimana caranya mengatur struktur *project* menjadi lebih rapi dengan menggunakan salah satu *state management* yang disediakan oleh Flutter, yakni Provider. *State management* sendiri merupakan sebuah cara yang dapat digunakan untuk mengatur, mengelola, dan menangani sebuah *state* / tindakan secara efisien.

3.1 Tujuan Praktikum

Tujuan	Penjelasan
Mengetahui dan memahami konsep *authentication* dalam keamanan sebuah aplikasi	Pada bab ini, praktikan diharapkan dapat mengetahui dan memahami konsep serta tujuan dari penggunaan *authentication* dalam sebuah aplikasi.
Mampu mengimplementasikan konsep *authentication* ke dalam aplikasi	Setelah mengetahui dan memahami konsep dan tujuan dari penggunaan *authentication*, praktikan diharapkan dapat menerapkan konsep tersebut ke dalam aplikasi yang telah dibangun pada pertemuan sebelumnya.
Mampu mengimplementasikan state management provider	Pada bab ini, praktikan akan dijelaskan terkait salah satu state management pada flutter, yakni provider. Praktikan akan dijelaskan mengenai provider, tujuan, hingga cara menggunakannya

3.2 Persyaratan Praktikum

Disarankan praktikan menggunakan *hardware* dan *software* sesuai pada dokumentasi ini. Apabila terdapat versi yang lumayan lampau dari versi yang direkomendasikan atau *hardware* yang lawas maka sebaiknya bertanya kepada Asisten Mengajar Shift.

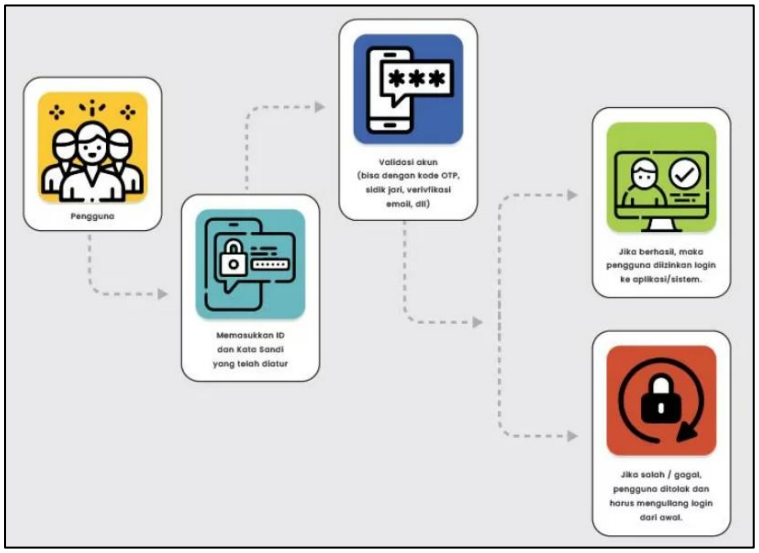
***HARDWARE* YANG DIBUTUHKAN PRAKTIKUM**	**JENIS**
PC / Laptop CPU	**≥ 4 Cores**
PC / Laptop RAM	**≥ 8 GB**
PC / Laptop *Storage*	**≥ 10 GB**

***SOFTWARE* YANG DIBUTUHKAN PRAKTIKUM**
Android *Studio* / *Visual Studio Code*

3.3 Materi Praktikum

3.3.1 *Authentication*

Authentication merupakan sebuah proses untuk memvalidasi atau memverifikasi identitas dari seseorang sebelum nantinya pengguna tersebut dapat menggunakan layanan dari sebuah aplikasi atau sistem. Authentication merupakan hal yang sangat penting dalam pengembangan sebuah aplikasi atau sistem, karena dengan adanya sistem authentication keamanan dari sebuah aplikasi atau sistem menjadi meningkat. Sistem ini akan memastikan bahwa hanya pengguna yang berwewenang saja yang dapat masuk dan menggunakan layanan dari sebuah aplikasi atau sistem. Pada aplikasi atau sistem dengan skala besar, authentication merupakan suatu layanan yang wajib digunakan untuk menjaga keamanan data, mencegah terjadinya penyalahgunaan akun, dan juga digunakan untuk melindungi informasi pribadi pengguna.



Gambar 3.1 Alur Kerja Sistem *Authentication*

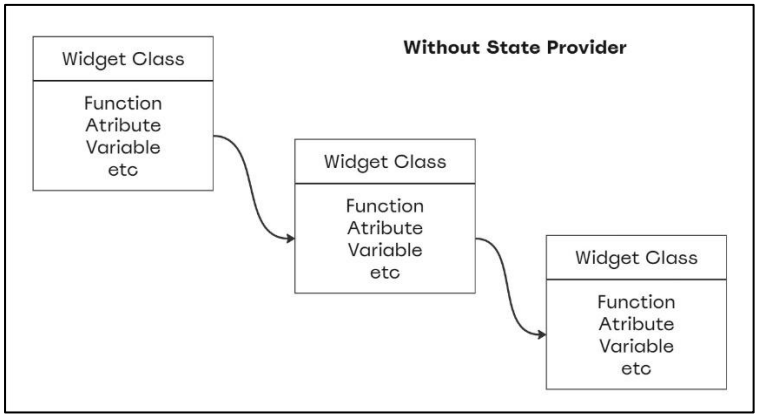
Cara kerja dari sistem *authentication* dapat dilihat pada ilustrasi gambar di atas. Pengguna akan memasukkan id (*email* atau *username*) dan *password* yang telah dibuat sebelumnya ke dalam *field* yang telah disediakan oleh aplikasi atau sistem. Selanjutnya sistem akan melakukan validasi atau verifikasi terhadap data tersebut menggunakan metode yang telah tersedia. Jika data yang di*input* oleh pengguna sudah sesuai, maka pengguna dapat masuk dan menikmati layanan yang disediakan oleh aplikasi atau sistem. Namun, jika terdapat salah satu data yang tidak sesuai, maka sistem akan menolak pengguna tersebut dan pengguna diarahkan kembali ke halaman *login.*

Saat ini, sistem *authentication* telah berkembang pesat dan memiliki berbagai metode yang digunakan, antara lain ***single factor authentication* (SFA), *double factor authentication* (2FA), *multi factor authentication* (MFA), *authentication token-based, authentication biometric-based, authentication* dengan OTP** dan lain-lain.

3.3.2 Provider

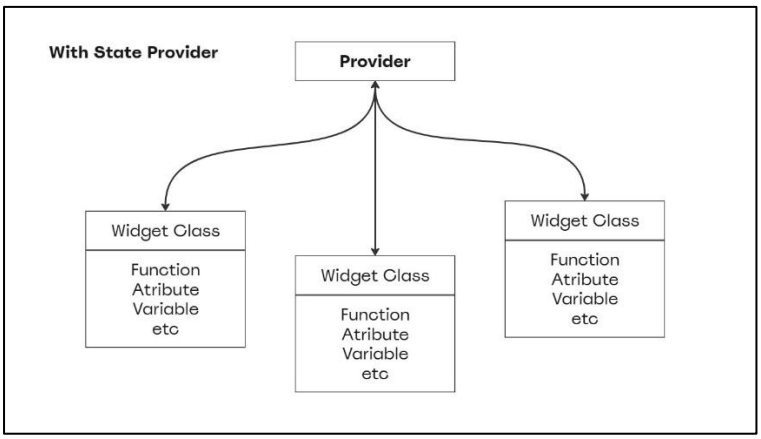
Flutter menyediakan banyak *package* yang dapat digunakan untuk mengimplementasikan *state management*, antara lain ***setstate***, ***provider***, ***bloc***, ***getX***, dan ***riverpod***. Provider merupakan salah satu *package state management* yang

menawarkan efisiensi, kesederhanaan dan fleksibilitas dalam hal mengelola *state* yang melibatkan beberapa widget tanpa harus melewati widget secara manual.



Gambar 3.2 Alur Kerja Aplikasi Tanpa Provider

Tanpa menggunakan provider proses *flow* atau alur *state* menjadi lebih panjang karena *bussines logic* tergabung dengan UI. Penggabungan tersebut akan membuat *state* harus diturunkan secara manual menggunakan *constructor* atau widget parameter dari widget induk ke widget anaknya (metode ini disebut *prop drilling*). Semakin kompleks dan dalam susunan hirarki widget pada suatu aplikasi maka semakin banyak level yang harus di lewati *state* tersebut meskipun diantaranya tidak membutuhkan *state* tersebut. Hal tersebut akan membuat kode menjadi lebih rumit dan sulit untuk di *maintenance*. *Flow* di dalamnya pun menjadi tidak efisien.



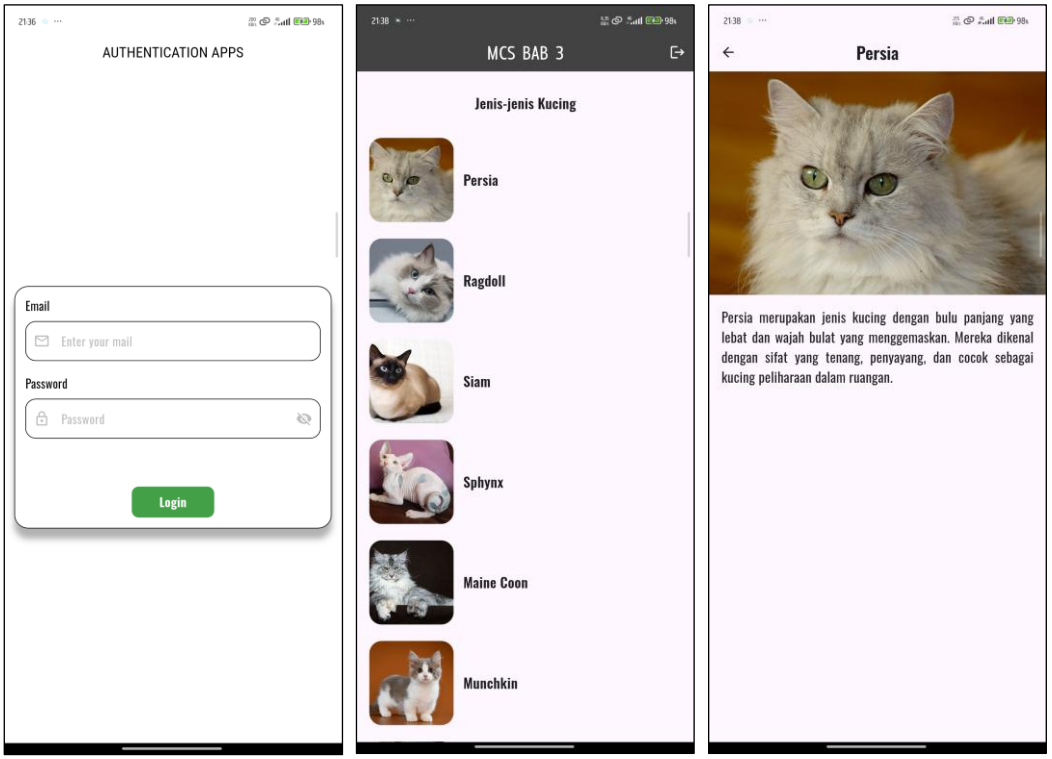
Gambar 3.3 Alur Kerja Aplikasi dengan Provider

Sedangkan jika menggunakan provider, maka *bussines logic* akan dibuat terpisah dari UI. *State* akan dikelola secara terpusat sehingga dapat diakses oleh widget manapun tanpa harus melewati widget induk terlebih dahulu. Provider memungkinkan widget untuk mendengar (*listen*) *state* yang ada tanpa harus meneruskan data melalui constructor atau parameter.

3.4 Prosedur Praktikum

3.4.1 Tampilan Aplikasi

Berikut merupakan tampilan dari aplikasi yang akan dibentuk pada praktikum bab 3.



Gambar 3.4 Tampilan Halaman Aplikasi yang Akan diimplementasikan

Penjelasan terkait bagaimana cara aplikasi bekerja akan diterangkan oleh asisten yang mengajar.

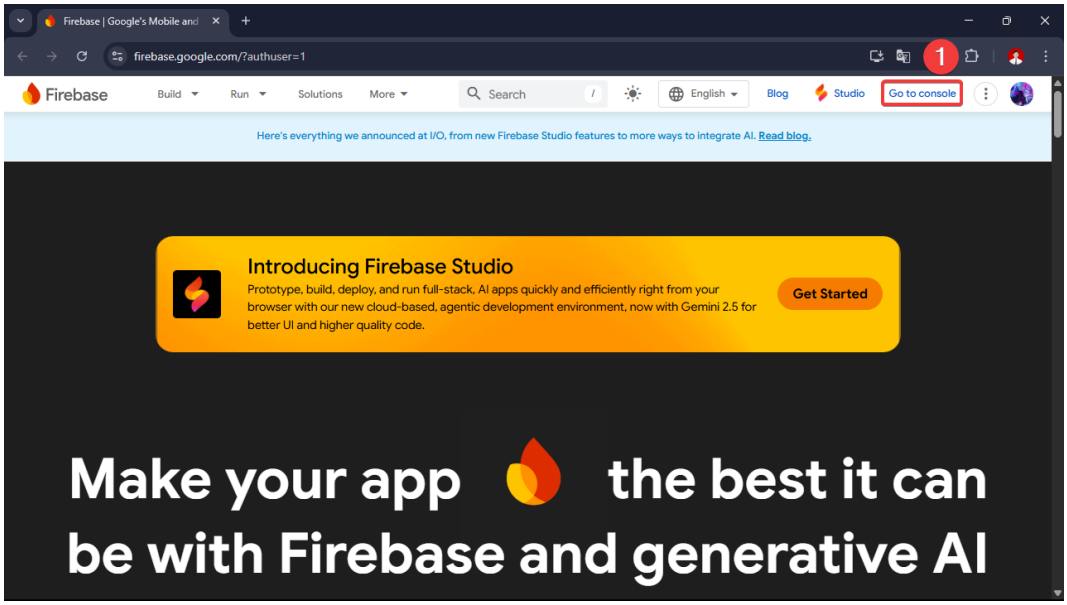
3.4.2 Implementasi Aplikasi

Praktikum yang akan dilakukan pada pertemuan ketiga ini merupakan lanjutan dari praktikum yang telah dilakukan pada pertemuan kedua. Pada

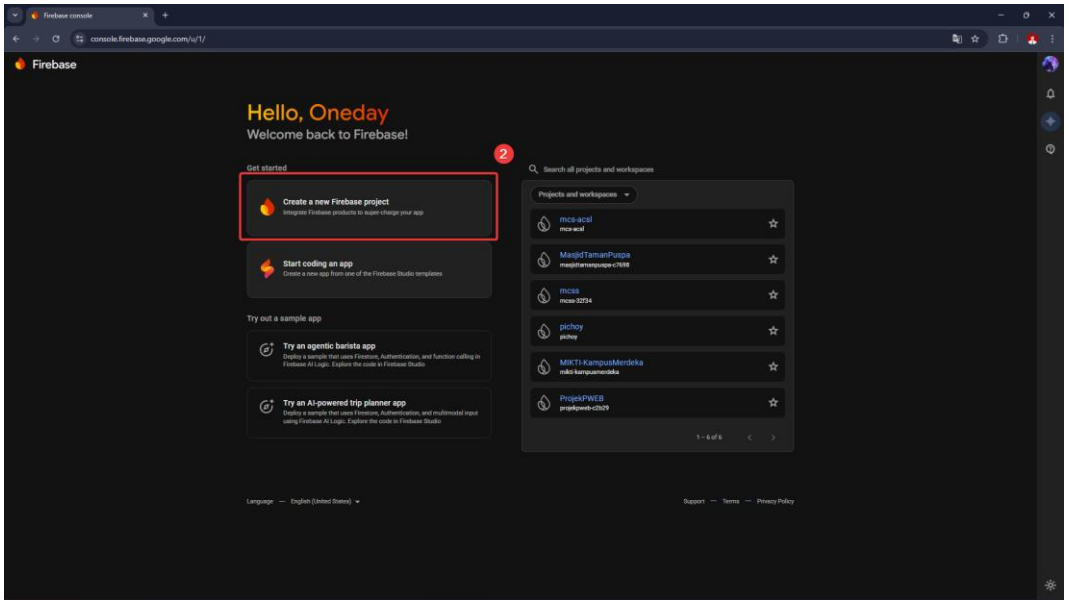
praktikum ini, aplikasi yang telah berisikan list kucing akan diberikan halaman tambahan, yakni halaman *login* dimana pada saat aplikasi dijalankan, pengguna tidak dapat langsung melihat halaman yang berisikan jenis-jenis kucing, melainkan pengguna akan diarahkan ke halaman *login* terlebih dahulu.

3.4.2.1 Konfigurasi Firebase

Dalam mengimplementasikan halaman *login* pada aplikasi, kita menggunakan *tools* tambahan yang disediakan oleh Google, yakni **firebase.** Pertama-tama bukalah website **https://firebase.google.com/** dan masuklah ke dalam bagian ***go to console*** yang dapat diakses pada appbar.

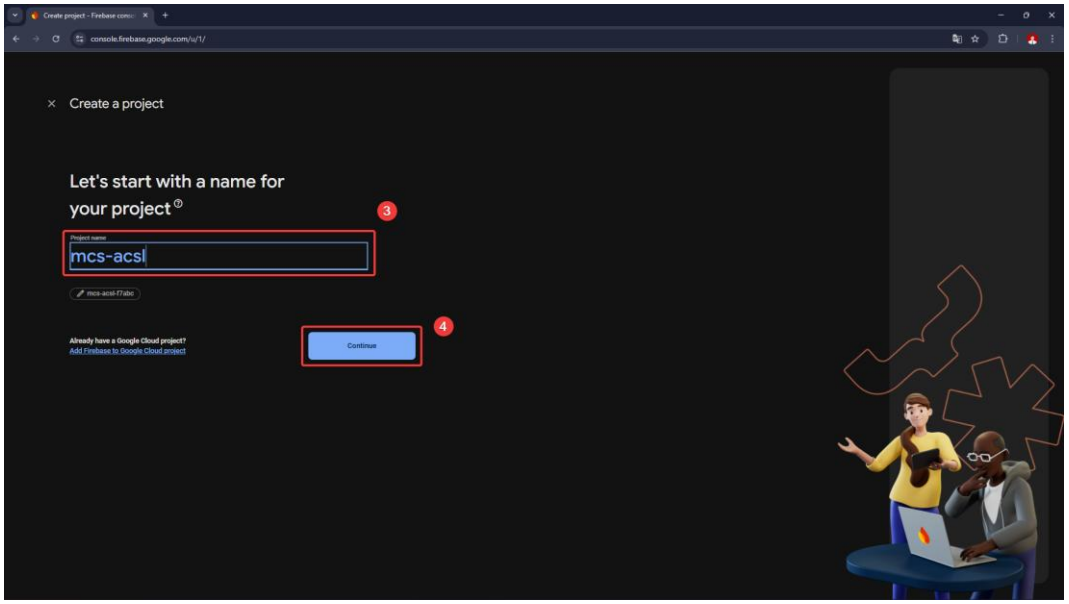


Gambar 3.5 Tampilan Halaman *Website Firebase*

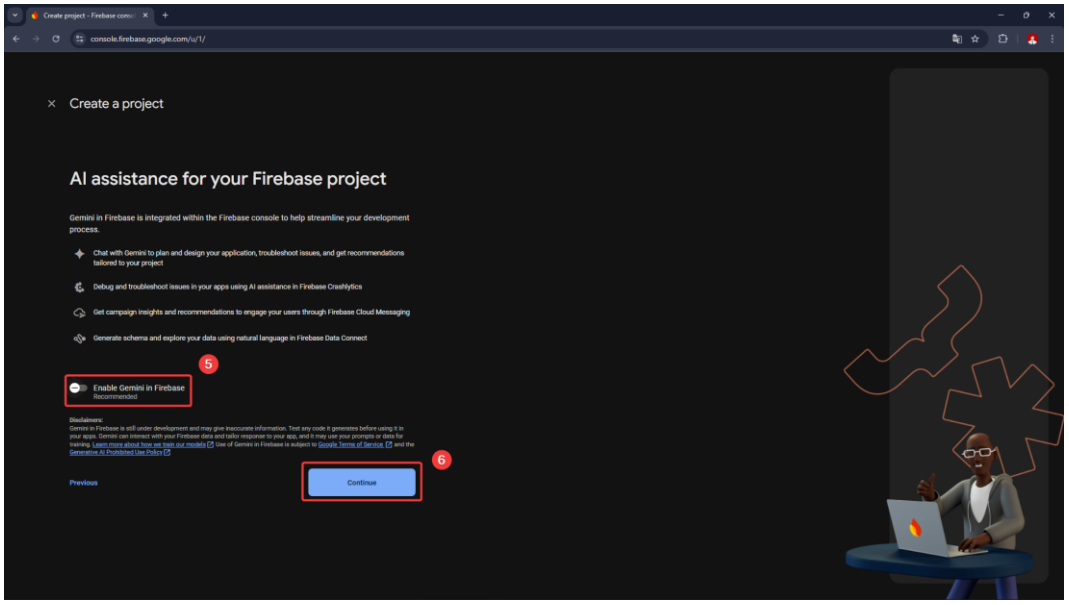


Gambar 3.6 Tampilan Halaman Menu *Console Firebase*

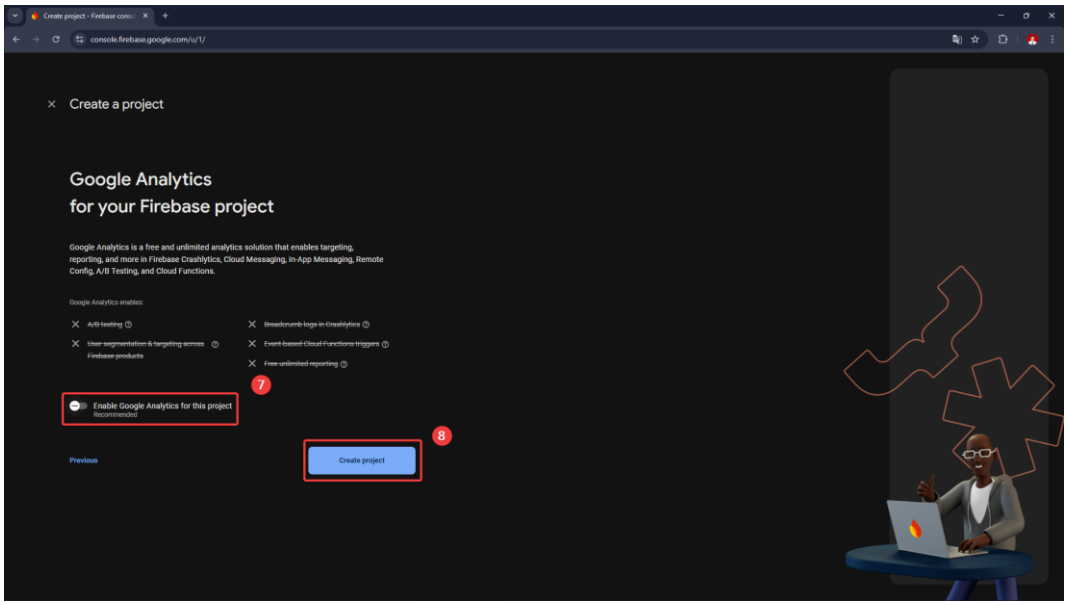
Setelah masuk ke dalam menu "*go to console*" tekanlah menu **"*create a new Firebase Projcet*"** untuk membuat *project* baru. Pada saat pembuatan *project* terdapat beberapa konfigurasi yang diminta, seperti nama *project*, AI *assistance*, dan *google analytics*.



Gambar 3.7 Proses Pembentukan Nama *Project*



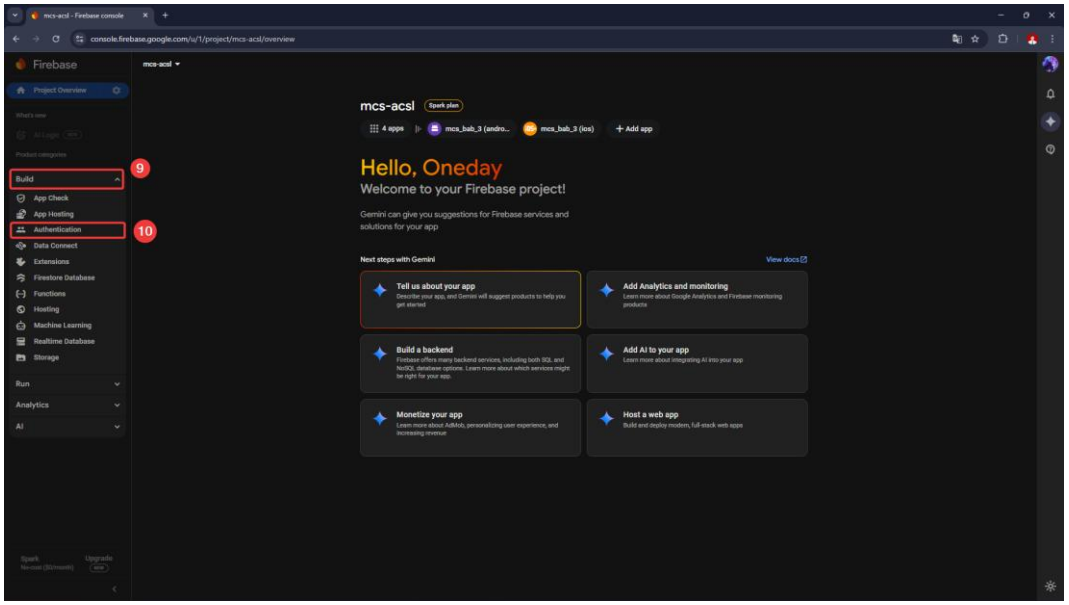
Gambar 3.8 Proses Konfigurasi AI *Assistance*



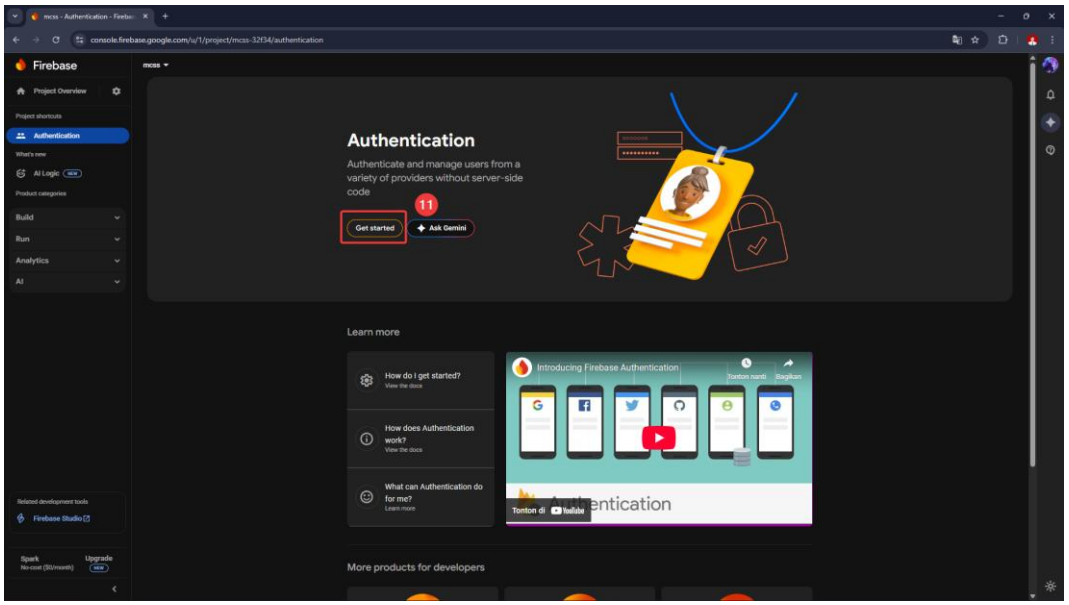
Gambar 3.9 Proses Konfiguasi Google *Analytics*

Setelah melewati beberapa konfigurasi dasar dari *firebase*, tampilan dari halaman *website* akan berubah, seperti yang terlihat pada gambar. Jika tampilan dari website sudah seperti pada gambar tersebut, tekanlah *navigation menu* ***build*** yang ada pada bagian sebelah kiri halaman *website* untuk menampilkan seluruh fitur yang disediakan oleh *firebase* yang dapat kita digunakan. Pilihlah fitur

authentication kemudian tekanlah menu **get started** untuk mulai menggunakan fitur tersebut.

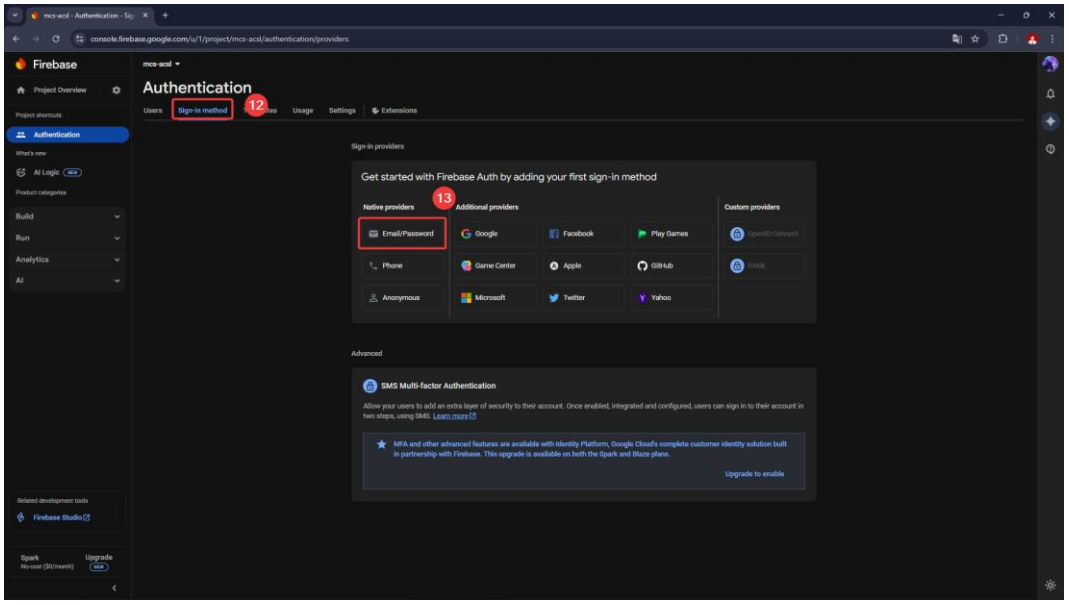


Gambar 3.10 Proses Pemilihan Fitur *Authentication*

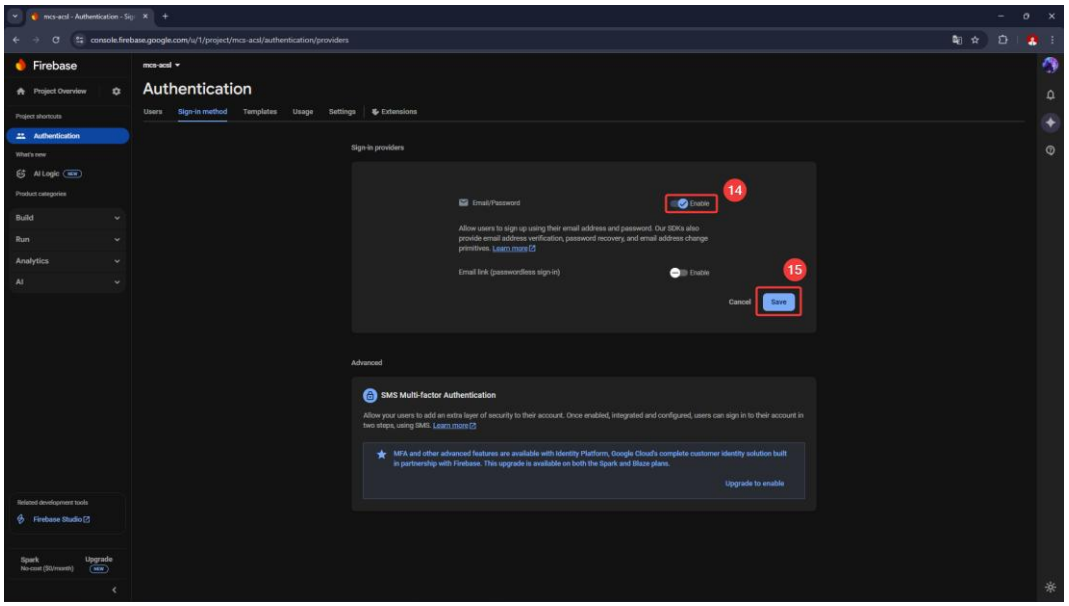


Gambar 3.11 Proses Penggunaan Pertama Fitur *Authentication*

Setelah proses pembuatan *project* berhasil, pilihlah provider **email / password** yang terdapat pada bagian native providers sebagai media *authentication* pada praktikum ini. Kemudian centang menu **enable** dan klik **save** untuk menyimpan hasil konfigurasi.

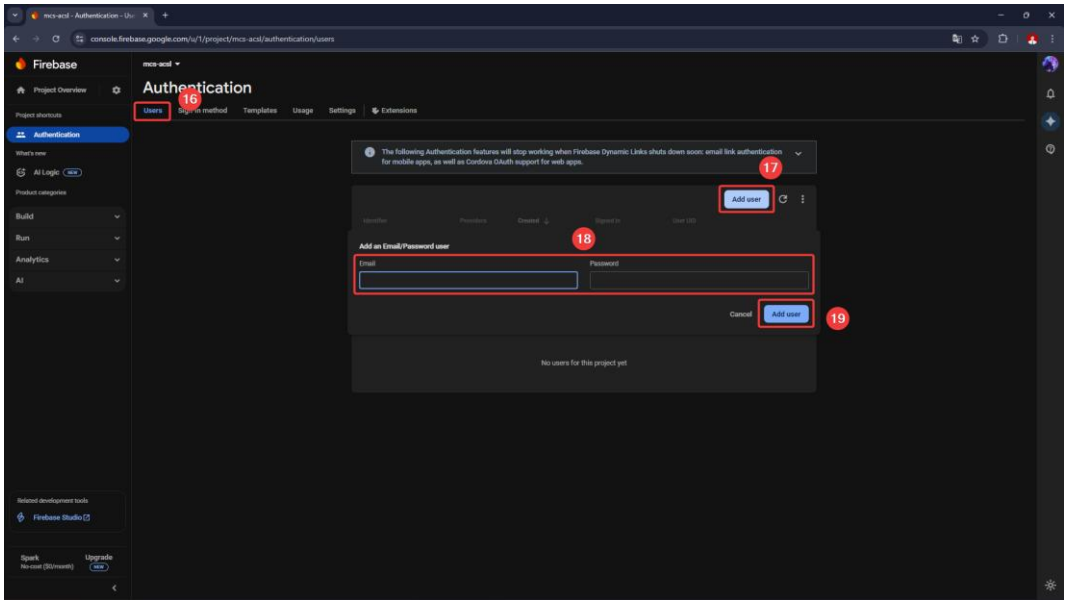


Gambar 3.12 Proses Pemilihan Metode *Authenticatoin*



Gambar 3.13 Proses Konfigurasi Metode *Authentication*

Berikutnya masuklah ke dalam menu **users** untuk menambahkan *user* baru untuk *authentication.* Pada menu tersebut klik bagian **add user** dan isikan ***email*** serta ***password***. Kemudian tekanlah *button* **add user** untuk menambahkan *user* baru ke dalam *firebase*.



Gambar 3.14 Proses Pembuatan Akun

Setelah *user* baru berhasil dibuat, langkah selanjutnya yang dilakukan adalah melakukan konfigurasi firebase dengan *project* flutter. Konfigurasi firebase dengan flutter *project* dilakukan dengan menggunakan ***Command Line Interface* (CLI)** yang prosesnya melalui beberapa tahapan, antara lain:

1. Pastikan laptop / PC sudah terinstall **node.js.** Jika laptop / PC belum terinstall node.js, maka harap mengunduhnya terlebih dahulu pada halaman website **https://nodejs.org/en.**
2. Pastikan path **C:\Users<nama user pada perangkat>\AppData\Local\Pub\Cache\bin** telah di*input* pada bagian **path** yang ada pada environtment variables.
3. Bukalah ***command prompt* (CMD)** atau ***windows powershell.***
4. Ketikan perintah **npm install -g firebase-tools** pada terminal yang telah dibuka dan tunggulah hingga proses installasi selesai. Perintah tersebut digunakan untuk menginstall firebase CLI pada perangkat secara global dengan menggunakan npm.

```
npm install -g firebase-tools
```

5. Bukalah *project* flutter yang telah dikerjakan pada pertemuan praktikum sebelumnya (praktikum bab 2) dan masuklah ke dalam terminal *project* tersebut.
6. Ketikan perintah **firebase login** untuk menghubungkan *project* dengan layanan firebase. Jika baru pertama kali menggunakan firebase dan belum pernah menghubungkan *project* flutter dengan firebase, maka nanti flutter akan mengarahkan ke sebuah halaman yang meminta akun untuk dihubungkan ke Firebase CLI.

```
Firebase login
```

7. Setelah berhasil *login* ke dalam firebase, ketikan perintah **firebase projects:list** untuk melihat seluruh *project* yang ada pada akun tersebut.

```
firebase projects:list
```

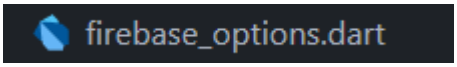
8. Ketikan perintah **dart pub global activate flutterfire_cli** untuk meng*install* flutterfire cli dan tunggu hingga proses installasi selesai.

```
dart pub global activate flutterfire_cli
```

9. Setelah proses installasi selesai, ketikan perintah **flutterfire configure** untuk mengonfigurasikan *project* flutter ke firebase. Dalam proses ini, terdapat beberapa langkah yang harus diikuti agar proses konfigurasi dapat berjalan dengan baik. **Praktikan diharapkan untuk memperhatikan dengan seksama penjelasan yang diberikan oleh PJ yang mengajar.**

```
flutterfire configure
```

10. Setelah proses konfigurasi ke firebase telah selesai, pada folder lib *project* flutter akan muncul sebuah file bernama **firebase_options.dart** secara otomatis. File tersebut berisikan konfigurasi untuk menghubungkan firebase dengan *project* Flutter.



Gambar 3.15 File Hasil Konfigurasi *Firebase*

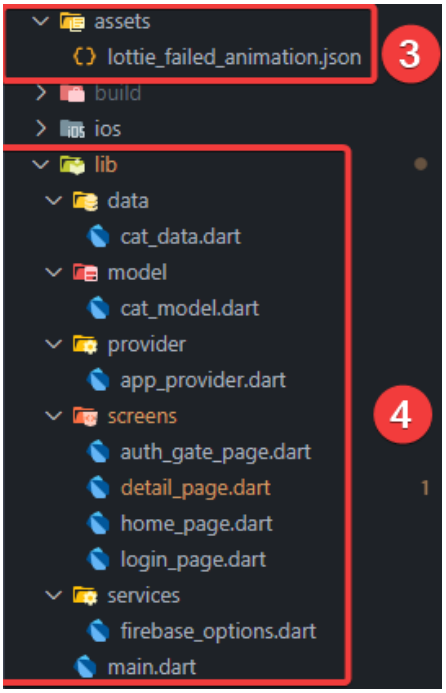
3.4.2.2 Pembuatan Aplikasi

Setelah file firebase_options.dart muncul, tambahkan 4 depedencies baru, yakni **firebase_core, firebase_auth, provider** dan **lottie** ke dalam file pubspec.yaml pada bagian *depedencies*. Selain itu, hilangkan komentar pada bagian **assets:** dan ubahlah *path* yang ada pada bagian **assets:** menjadi seperti pada Gambar 3.16.

```
30  dependencies:
31    flutter:
32      sdk: flutter
33
34    # The following adds the Cupertino Icons font to your application.
35    # Use with the CupertinoIcons class for iOS style icons.
36    cupertino_icons: ^1.0.8
37    google_fonts: ^6.2.1
38    firebase_core: ^3.14.0
39    firebase_auth: ^5.6.0
40    provider: ^6.1.5
41    lottie: ^3.3.1
42
43
44  dev_dependencies:
45    flutter_test:
46      sdk: flutter
47
48    # The "flutter_lints" package below contains a set of recommended lints to
49    # encourage good coding practices. The lint set provided by the package is
50    # activated in the `analysis_options.yaml` file located at the root of your
51    # package. See that file for information about deactivating specific lint
52    # rules and activating additional ones.
53    flutter_lints: ^5.0.0
54
55  # For information on the generic Dart part of this file, see the
56  # following page: https://dart.dev/tools/pub/pubspec
57
58  # The following section is specific to Flutter packages.
59  flutter:
60
61    # The following line ensures that the Material Icons font is
62    # included with your application, so that you can use the icons in
63    # the material Icons class.
64    uses-material-design: true
65
66    # To add assets to your application, add an assets section, like this:
67    assets:
68      - assets/
```

Gambar 3.16 *Package* yang digunakan

Berikutnya tambahkan beberapa file dan folder ke dalam folder lib, sehingga struktur *tree project* pada folder lib akan terlihat, seperti pada gambar berikut:



Gambar 3.17 Struktur *Tree Project*

Setelah menambahkan beberapa folder dan file ke dalam folder lib, masuklah ke dalam file **app_provider.dart** yang terdapat dalam folder **provider** dan masukkanlah kode berikut ke dalamnya.

```
class AppProvider extends ChangeNotifier {
  // INISIALISASI VARIABEL
  final GlobalKey<FormState> formKey = GlobalKey<FormState>();
  TextEditingController           usernameController           =
TextEditingController();
  TextEditingController passController = TextEditingController();
  final firebaseAuthentication = FirebaseAuth.instance;
  bool visibilityIcon = true;
  final String hintUsernameText = "Enter your mail";
  final String hintPassText = "Password";
  final String loginButtonText = "Login";
  final Color? backgroundColor = Colors.white;
  final Color? iconColor = Colors.grey[400];

  final titleFontstyle = GoogleFonts.robotoCondensed(
    fontSize: 17,
    color: Colors.black,
```

```
  );
  final hintTextFontStyle = GoogleFonts.oswald(
    fontSize: 14,
    color: Colors.grey[400],
  );
  final buttonFontStyle = GoogleFonts.oswald(
    fontSize: 14,
    color: Colors.white,
    fontWeight: FontWeight.bold,
  );
  final universalFontStyle = GoogleFonts.oswald(
    fontSize: 14,
    color: Colors.black,
  );

  // METHOD LOGIN FIREBASE
  Future loginToApp(
    String usernameController,
    String passwordController,
  ) async {
    try {
      UserCredential        userCredential        =        await
firebaseAuthentication
          .signInWithEmailAndPassword(
            email: usernameController,
            password: passwordController,
          );
      return userCredential;
    } catch (e) {
      throw Exception(e);
    }
  }

  // METHOD LOGOUT FIREBASE
  Future logoutAccount() async {
    await firebaseAuthentication.signOut();
  }
```

```
  // METHOD VALIDASI FORM
  Future validationForm(BuildContext context) async {
    if (formKey.currentState!.validate()) {
      notifyListeners();
      try {
        await                loginToApp(usernameController.text,
passController.text);
      } catch (e) {
        showDialog(
          context: context,
          builder: (builder) {
            return AlertDialog(
              title: Column(
                crossAxisAlignment: CrossAxisAlignment.center,
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  Text("Login Failed"),
                  SizedBox(width: 10),
                  LottieBuilder.asset(
                    height:  MediaQuery.of(context).size.width  /
6,

                    width: MediaQuery.of(context).size.width / 6,
                    "assets/lottie_failed_animation.json",
                    repeat: true,
                  ),
                ],
              ),
              content: Text(
                "Silahkan  Masukkan  Username  dan  Password  yang
Benar",

                textAlign: TextAlign.center,
              ),
            );
          },
        );
      }
```

```
    }
  }

  // METHOD RESET FIELD FORM
  void resetForm() {
    usernameController.clear();
    passController.clear();
    formKey.currentState!.reset();
    notifyListeners();
  }
}
```

Dalam file tersebut, terdapat *class* **AppProvider** yang meng*extends* **ChangeNotifier** yang berfungsi sebagai notifikasi bagi widget lain ketika adanya suatu perubahan terhadap data atau *state*, sehingga tampilan aplikasi dapat diperbarui secara otomatis. *Class* ini memiliki beberapa pendefinisian terhadap variabel dan *method* yang akan digunakan dalam membuat aplikasi *authentication.*

1. **formKey** merupakan variabel unik yang digunakan untuk mengontrol seluruh *field input* yang dibungkus oleh widget Form().
2. **usernameController** merupakan variabel yang menyimpan inisialisasi terhadap TextEditingController() yang berfungsi untuk mengontrol dan mengambil data yang di*input* oleh pengguna pada *field username*.
3. **passController** merupakan variabel yang menyimpan inisialisasi terhadap TextEditingController() yang berfungsi untuk mengontrol dan mengambil data yang di*input* oleh pengguna pada *field password*.
4. **firebaseAuthentication** berfungsi sebagai variabel yang menyimpan inisialisasi firebase *authentication* ke dalam *project*. Inisialisasi ini digunakan agar pada saat pembangunan aplikasi, kita dapat menggunakan beberapa *method* yang berkaitan dengan *authentication,* seperti **signInWithEmailAndPassword(), signInAnonymously(), signOut()** dan metode *authentication* lainnya.
5. **visibilityIcon** yang diberi nilai *default* ***true***. Variabel ini nantinya akan digunakan untuk *logic* pada icon mata yang ada pada *field password*.

6. **hintUsernameText** merupakan variabel yang berisikan inisialisasi text yang akan digunakan sebagai petunjuk untuk *field username*.
7. **hintPassText** merupakan variabel yang berisikan inisialisasi text yang digunakan sebagai petunjuk untuk *field password.*
8. **loginButtonText** merupakan variabel yang berisikan inisialisasi text yang digunakan pada tombol *login* aplikasi.
9. **backgroundColor** merupakan variabel yang berisikan pendefinisian terhadap warna *background* aplikasi.
10. **iconColor** merupakan variabel yang berisikan pendefinisian terhadap warna *icon* aplikasi.

Kemudian pada *class* tersebut juga terdapat pendefinisian terhadap beberapa method yang akan digunakan, antara lain:

1. *Method* **loginToApp()** merupakan *method* yang bersifat *asynchronous. Method* ini digunakan untuk menangani proses verifikasi *login* firebase dengan pemanggilan *method* **signInWithEmailAndPassword().** Data yang digunakan untuk *login* diambil dari *input* pengguna dan diambil dengan menggunakan parameter.
2. *Method* **logoutAccount()** merupakan *method* yang bersifat *asynchronous* yang akan menangani proses *logout* dari *firebase*.
3. *Method* **validationForm()** merupakan *method* yang digunakan untuk memenangani proses validasi data berdasarkan data yang telah di*input* oleh pengguna. *Method* ini akan memeriksa seluruh *field* yang ada dan mengirimkan data yang diterima dari masing-masing *field* ke *method* loginToApp() untuk diproses oleh firebase.
4. *Method* **resetForm()** merupakan *method* yang berisikan beberapa baris kode yang digunakan untuk menghapus dan mengosongkan nilai dari *field input* dan digunakan juga untuk mereset status validasi dari widget.

Berikutnya bukalah file **login_page.dart** dan masukkanlah kode program berikut:

```
class LoginPage extends StatefulWidget {
  const LoginPage({super.key});

  @override
  State<LoginPage> createState() => _LoginPageState();
}

class _LoginPageState extends State<LoginPage> {
  @override
  Widget build(BuildContext context) {
    return Consumer<AppProvider>(
      builder: (context, appProvider, child) {
        return Scaffold(
          backgroundColor: appProvider.backgroundColor,
          appBar: AppBar(
            backgroundColor: appProvider.backgroundColor,
            title: Text(
              "AUTHENTICATION APPS",
              style: appProvider.titleFontstyle,
            ),
            centerTitle: true,
          ),
          body: Padding(
            padding: const EdgeInsets.all(15),
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                Container(
                  decoration: BoxDecoration(
                    border: BoxBorder.all(color: Colors.black),
                    borderRadius: BorderRadius.circular(20),
                    color: Colors.white,
                    boxShadow: [
                      BoxShadow(
                        color: Colors.black.withOpacity(0.3),
```

```
                        blurRadius: 7,
                        offset: const Offset(0, 15),
                      ),
                    ],
                  ),
                  child: Padding(
                    padding: const EdgeInsets.all(15),
                    child: Column(
                      mainAxisAlignment:
MainAxisAlignment.center,
                      mainAxisSize: MainAxisSize.min,
                      children: [
                        Form(
                          key: appProvider.formKey,
                          child: Column(
                            mainAxisAlignment:
MainAxisAlignment.center,
                            crossAxisAlignment:
CrossAxisAlignment.start,
                            children: [
                              Text(
                                "Email",
                                style:
appProvider.universalFontStyle,
                              ),
                              SizedBox(height: 10),
                              TextFormField(
                                controller:
appProvider.usernameController,
                                keyboardType:
TextInputType.emailAddress,
                                style:
appProvider.universalFontStyle,
                                decoration: InputDecoration(
                                  border: OutlineInputBorder(
                                    borderRadius:
BorderRadius.circular(15),
```

```
                                    ),
                                    hintText:
appProvider.hintUsernameText,
                                    hintStyle:
appProvider.hintTextFontStyle,
                                    enabledBorder:
OutlineInputBorder(
                                      borderRadius:
BorderRadius.circular(15),
                                      borderSide:
BorderSide(color: Colors.black),
                                    ),
                                    focusedBorder:
OutlineInputBorder(
                                      borderRadius:
BorderRadius.circular(15),
                                      borderSide:
BorderSide(color: Colors.black),
                                    ),
                                    prefixIcon: Icon(
                                      Icons.mail_outline,
                                      color:
appProvider.iconColor,
                                    ),
                                  ),
                                  autovalidateMode:

AutovalidateMode.onUserInteraction,
                                  validator: (value) {
                                    if (value!.isEmpty) {
                                      return   "Harap   mengisikan
kolom username";
                                    }
                                    RegExp emailFormat = RegExp(
                                      r'^[\w-\.]+@([\w-]+\.)+[\w-
]{2,4}$',
                                    );
```

```
                                    if
(emailFormat.hasMatch(value)) {
                                      return null;
                                    }
                                    return 'Harap masukkan e-mail
sesuai format';
                                  },
                                ),
                                SizedBox(height: 20),
                                Text(
                                  "Password",
                                  style:
appProvider.universalFontStyle,
                                ),
                                SizedBox(height: 10),
                                TextFormField(
                                  controller:
appProvider.passController,
                                  obscureText:
appProvider.visibilityIcon == true
                                      ? true
                                      : false,
                                  style:
appProvider.universalFontStyle,
                                  decoration: InputDecoration(
                                    border: OutlineInputBorder(
                                      borderRadius:
BorderRadius.circular(15),
                                    ),
                                    hintText:
appProvider.hintPassText,
                                    hintStyle:
appProvider.hintTextFontStyle,
                                    enabledBorder:
OutlineInputBorder(
                                      borderRadius:
BorderRadius.circular(15),
```

```
                                    borderSide:
BorderSide(color: Colors.black),
                                  ),
                                  focusedBorder:
OutlineInputBorder(
                                    borderRadius:
BorderRadius.circular(15),
                                    borderSide:
BorderSide(color: Colors.black),
                                  ),
                                  prefixIcon:
Icon(Icons.lock_outline),
                                  prefixIconColor:
appProvider.iconColor,
                                  suffixIcon: GestureDetector(
                                    child:
appProvider.visibilityIcon == true
                                        ? Icon(

Icons.visibility_off,
                                            color:
appProvider.iconColor,
                                          )
                                        : Icon(
                                            Icons.visibility,
                                            color:
appProvider.iconColor,
                                          ),
                                    onTap: () {
                                      setState(() {

appProvider.visibilityIcon =

!appProvider.visibilityIcon;
                                      });
                                    },
                                  ),
```

```
                                    ),
                                    autovalidateMode:
AutovalidateMode.onUnfocus,
                                    validator: (value) {
                                      if (value!.isEmpty) {
                                        return   "Harap   mengisikan
kolom password";
                                      }
                                      return null;
                                    },
                                  ),
                                ],
                              ),
                            ),
                            SizedBox(height: 70),
                            GestureDetector(
                              child: Container(
                                decoration: BoxDecoration(
                                  color: Colors.green[600],
                                  borderRadius:
BorderRadius.circular(10),
                                ),
                                child: Padding(
                                  padding:                        const
EdgeInsets.symmetric(
                                    horizontal: 40,
                                    vertical: 10,
                                  ),
                                  child: Text(
                                    appProvider.loginButtonText,
                                    style:
appProvider.buttonFontStyle,
                                    textAlign: TextAlign.center,
                                  ),
                                ),
                              ),
                              onTap: () {
```

```
                                appProvider.validationForm(context);
                              },
                            ),
                          ],
                        ),
                      ),
                    ),
                  ],
                ),
              ),
            ),
          );
        },
      );
    }
}
```

Halaman login_page.dart berisikan baris kode program yang akan membangun tampilan dari halaman login aplikasi dan merupakan tampilan pertama yang akan ditampilkan oleh aplikasi. Halaman ini akan meng*extends StatefulWidget* dan mengembalikan widget **Consumer()** yang merupakan widget yang tersedia jika kita menggunakan Provider. Widget ini dapat membantu kita dalam mengakses *instance* dari provider yang di dalamnya terdapat beberapa variabel atau *method* yang telah didefinisikan. Widget ini juga akan melakukan *rebuild* UI yang ada di dalamnya ketika Consumer mendapati perubahan pada salah satu *state*, sehingga pengguna dapat langsung melihat perubahan data tersebut pada tampilan aplikasi.

Widget ini memiliki properti builder: yang didalamnya terdapat 3 parameter, antara lain **context, provider,** dan **child.** Parameter provider merupakan parameter yang merupakan *instance* dari *class* provider yang telah dibuat sebelumnya. Pada praktikum ini, parameter tersebut diberi nama sebagai **appProvider,** sehingga ketika ingin mengakses salah satu variabel atau *method* yang telah didefinisikan pada *class* AppProvider() kita hanya cukup memanggil appProvider yang diikuti dengan pemanggilan nama variabel atau *method* yang ingin digunakan.

Widget ini mengembalikan widget **Scaffold()** untuk membentuk halamannya dimana widget langsung menggunakan properti body: yang mengembalikan widget **Form().** Widget tesebut digunakan untuk membantu kita dalam proses validasi nilai *input* yang didapatkan dari pemanggilan terhadap widget TextFormField(). Implementasi dari provider pertama kali dapat dilihat pada properti **key:** dengan pemanggilan terhadap **appProvider** yang diikuti dengan pemanggilan terhadap variabel **formKey** yang telah terdefinisikan di dalam *class* AppProvider(). Penggunaan dari provider lainnya dapat dilihat pada properti **controller:** yang ada di setiap widget TextFormField, properti **suffixIcon** yang ada pada widget TextFormField *password*, dan ***button* untuk *login***.

Berikutnya masuklah ke dalam file **home_page.dart** dan ubahlah kode di dalamnya menggunakan kode program berikut:

```
class HomePage extends StatefulWidget {
  const HomePage({super.key});

  @override
  State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  @override
  Widget build(BuildContext context) {
    return Consumer<AppProvider>(
      builder: (context, appProvider, child) {
        return Scaffold(
          appBar: AppBar(
            title: const Text(
              "MCS BAB 3",
              style: TextStyle(color: Colors.white),
            ),
            centerTitle: true,
            backgroundColor: Colors.grey[800],
            actions: [
              IconButton(
                icon: Icon(Icons.logout, color: Colors.white),
```

```
                onPressed: () {
                  appProvider.logoutAccount();
                  appProvider.resetForm();
                },
              ),
            ],
          ),
          body: ListView(
            children: [
              const SizedBox(height: 32),
              Center(
                child: Text(
                  "Jenis-jenis Kucing",
                  style: GoogleFonts.oswald(
                    fontSize: 16,
                    fontWeight: FontWeight.w600,
                  ),
                ),
              ),

              const SizedBox(height: 24),

              ListView.builder(
                itemCount: cats.length,
                shrinkWrap: true,
                physics: const NeverScrollableScrollPhysics(),
                itemBuilder: (_, index) {
                  CatModel cat = cats[index];
                  return Container(
                    margin: const EdgeInsets.symmetric(
                      vertical: 12,
                      horizontal: 18,
                    ),
                    child: GestureDetector(
                      child: Row(
                        children: [
                          SizedBox(
```

```
                              height:
MediaQuery.of(context).size.width / 4,
                              width:
MediaQuery.of(context).size.width / 4,
                              child: ClipRRect(
                                borderRadius:
BorderRadius.circular(18),
                                child: Image.network(
                                  cat.urlImage,
                                  fit: BoxFit.cover,
                                ),
                              ),
                            ),
                            const SizedBox(width: 14),
                            Text(
                              cat.name,
                              style: GoogleFonts.oswald(
                                fontSize: 16,
                                fontWeight: FontWeight.w600,
                              ),
                            ),
                          ],
                        ),
                        onTap: () {
                          Navigator.push(
                            context,
                            MaterialPageRoute(
                              builder: (context) {
                                return DetailPage(catModel: cat);
                              },
                            ),
                          );
                        },
                      ),
                    );
                  },
                ),
```

```
              ],
            ),
          );
        },
      );
    }
}
```

Perubahan kode tersebut dilakukan pada 2 bagian, pertama pada bagian pengembalian widget yang awalnya langsung mengembalikan widget Scaffold() dirubah menjadi widget Consumer(), sehingga widget Scaffold() berada di dalam widget Consumer(). Kemudian perubahan kedua terjadi pada bagian widget AppBar() dimana terdapat penambahan properti **action:** yang berisikan *icon* berbentuk *logout*. *Icon* tersebut nantinya dapat ditekan dan pada saat *icon* ditekan, maka sistem akan memanggil *method* **logoutApp()** dan **resetForm()** yang tersimpan di dalam *class* AppProvider. Kedua *method* tersebut akan membuat pengguna keluar dari akun tersebut dan kembali ke halaman *login* serta mengosongkan *field input* yang ada pada halaman *login*.

Selanjutnya masuklah ke dalam file **auth_page_gate.dart** dan masukkanlah kode program berikut:

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

  @override
  Widget build(BuildContext context) {
    return Consumer<AppProvider>(
      builder: (context, appProvider, child) {
        return StreamBuilder(
          stream:
appProvider.firebaseAuthentication.authStateChanges(),
          builder: (context, snapshot) {
            if (snapshot.hasData) {
              return HomePage();
            } else {
```

```
                return LoginPage();
              }
            },
          );
        },
      );
    }
}
```

Class **AuthPageGate** merupakan *class* yang berisikan beberapa baris kode program yang akan mengatur halaman mana yang akan ditampilkan. *Class* ini mengembalikan widget Consumer() yang di dalamnya terdapat pemanggilan terhadap widget Scaffold(). Pada widget Scaffold() dipanggil properti body: yang mengembalikan widget **StreamBuilder()** yang di dalamnya terdapat properti **stream:** yang akan memantau data *stream* secara terus-menerus, sehingga tampilan UI dari halaman tersebut akan di*rebuild* secara *real time* jika terdapat data yang berubah. Terdapat 2 kondisi berbeda yang akan ditampilkan oleh aplikasi, yakni kondisi pertama jika pada saat proses *stream* terdapat data (pengguna telah melakukan *login*), maka aplikasi akan langsung menampilkan halaman *home* yang berisikan *list* dari jenis-jenis kucing. Namun, jika pada saat *stream* sistem tidak menemukan data (pengguna belum pernah *login* atau sudah *logout*), maka tampilan yang akan ditampilkan adalah halaman *login.*

Berikutnya, masuklah ke dalam file **main.dart** dan ubahlah kode di dalamnya dengan menggunakan kode program berikut:

```
void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await                          Firebase.initializeApp(options:
DefaultFirebaseOptions.currentPlatform);
  runApp(const MyApp());
}

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

```
  Widget build(BuildContext context) {
    return MultiProvider(
      providers: [ChangeNotifierProvider(create: (context) =>
AppProvider())],
      child: MaterialApp(
        title: 'MCS BAB 3',
        debugShowCheckedModeBanner: false,
        theme: ThemeData(useMaterial3: true),
        home: AuthGatePage(),
      ),
    );
  }
}
```

Dalam file **main.dart** terdapat penambahan terhadap *function* main() dan *class* MyApp. *Function* main() diubah menjadi bentuk *asynchronous* yang di dalamnya terdapat kode program **WidgetsFlutterBinding.ensureInitialized();** yang akan memastikan bahwa seluruh widget yang akan digunakan telah selesai diinisialisasi dengan baik. Kemudian di bawahnya terdapat kode program **await Firebase.initializeApp(options: DefaultFirebaseOptions.currentPlatform)** yang akan membuat flutter menunggu terlebih dahulu sampai firebase berhasil terinisialisasi.

Kemudian *class* **MyApp** akan mengembalikan widget MultiProvider yang memungkinkan penggunaan beberapa provider. Provider yang akan digunakan pada praktikum ini didefinisikan pada properti **providers:** yang tersimpan dalam bentuk list. Dalam praktikum ini didefinisikan **ChangeNotifierProvider** yang mengembalikan *class* AppProvider() yang meng*extends* ChangeNotifier. ChangeNotifierProvider menyediakan *instance* dari *class* AppProvider yang akan digunakan pada aplikasi sebagai *state management*.