

LAPORAN UAS
PRAKTIKUM PEMROGRAMAN MOBILE
SEMESTER GANJIL 2023/2024
“FLUTTER NEWS APP WITH API”



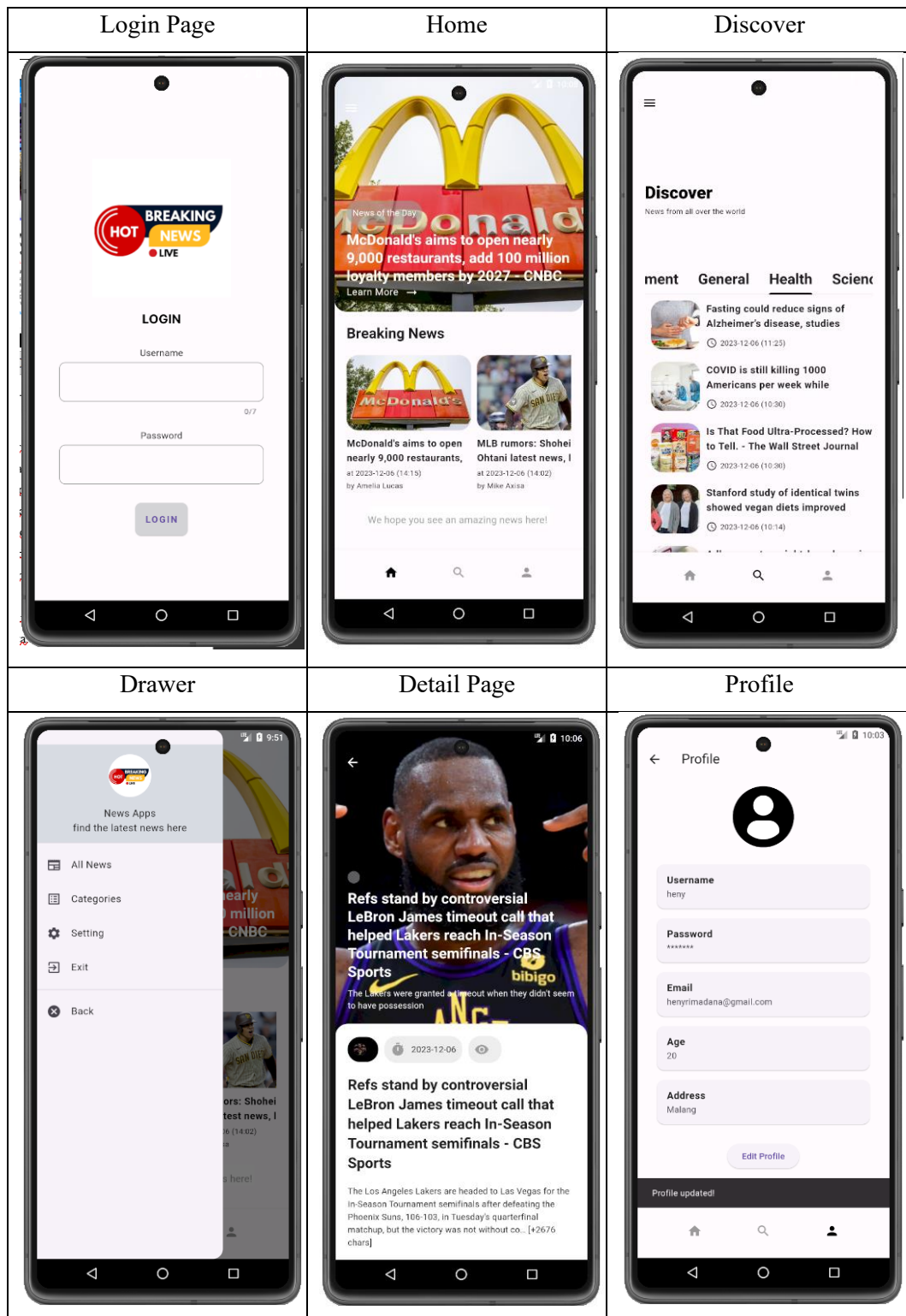
Disusun oleh:

Heny Rimadana	210605110009
Muhammad Reyhan Aditya H.	210605110060

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI
MAULANA MALIK IBRAHIM MALANG

2023

Flutter News App With API



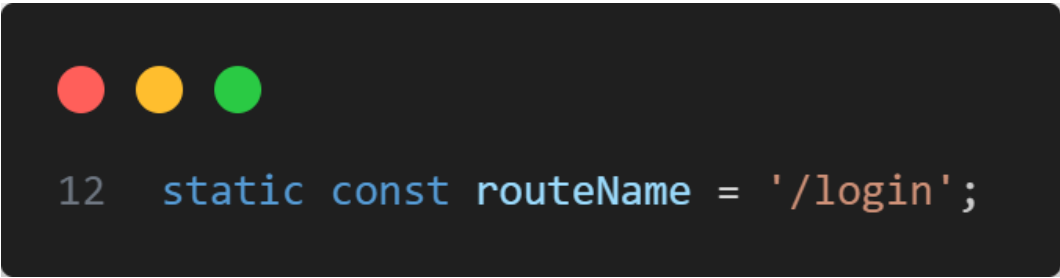
A. Login Page

Halaman Login adalah tampilan awal untuk proses login dalam aplikasi Flutter. Di sini, pengguna diminta untuk memasukkan username dan password mereka. Tampilan terdiri dari:

- **Gambar dan Judul:** Terdapat gambar logo atau ilustrasi (dalam kasus ini, menggunakan gambar 'news.png') di bagian atas halaman, diikuti dengan judul "LOGIN" yang ditampilkan dengan gaya huruf tebal.
- **Formulir Login:** Terdapat dua input teks untuk username dan password. Setiap input dilengkapi dengan label ("Username" dan "Password") di atasnya, memberikan petunjuk kepada pengguna tentang informasi apa yang harus dimasukkan ke dalam kotak input. Selain itu, ada juga batasan maksimum karakter pada input untuk username, yaitu 7 karakter.
- **Tombol Login:** Terdapat tombol "LOGIN" yang mengaktifkan proses login ketika ditekan. Tombol ini memiliki tata letak yang konsisten dengan desain secara keseluruhan dan memberikan umpan balik saat ditekan. Proses login akan memeriksa apakah username dan password yang dimasukkan oleh pengguna sesuai dengan data yang telah ditentukan sebelumnya dalam **users**. Jika sesuai, pengguna akan diarahkan ke halaman profil (**Profile.routeName**), jika tidak, akan muncul pesan kesalahan dengan Snackbar yang memberitahu bahwa username atau password yang dimasukkan salah.

Berikut beberapa code penting pada Login Page:

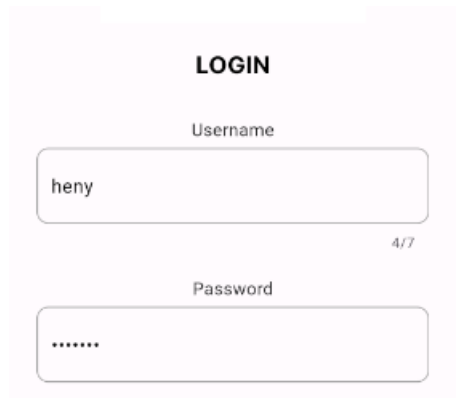
1. Routes



```
12 static const routeName = '/login';
```

Baris `static const routeName = '/login';` digunakan untuk menetapkan sebuah nilai konstan yang merepresentasikan rute atau lokasi dari halaman login dalam aplikasi. Dalam hal ini, nilai konstan tersebut adalah `'/login'`.

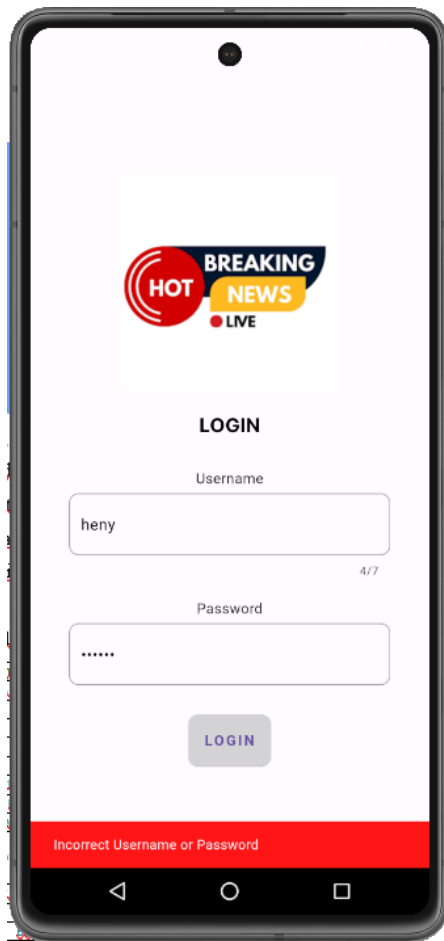
2. Text Field Login



Dua objek **TextEditingController** digunakan untuk mengelola input teks dari pengguna pada formulir login. Variabel **users** adalah sebuah map yang menyimpan pasangan username dan password. Dalam contoh ini, terdapat dua entri untuk pengguna "heny" dan "reyhan" beserta kata sandinya. Anda dapat menambahkan pasangan username dan password tambahan dalam map tersebut untuk memperluas fungsionalitas login dengan menambahkan entri baru menggunakan format `"username": "password"`. Adapun kode program penyusun bagian drawer ini sebagai berikut:

```
14 final TextEditingController _usernameController = TextEditingController();
15 final TextEditingController _passwordController = TextEditingController();
16
17 final Map<String, String> users = {
18     "heny": "heny123",
19     "reyhan": "reyhan123",
20     // Tambahkan username dan password tambahan di sini
21 };
22
```

3. Trigger Ketika Username/Password Salah



Di dalamnya terdapat elevated button yang mengatur tindakan saat ditekan. Ketika tombol tersebut ditekan, aplikasi akan mengambil nilai dari input username dan password yang dimasukkan oleh pengguna. Kemudian, akan dilakukan pengecekan apakah pasangan username dan password tersebut cocok dengan data yang tersimpan. Jika cocok, data username dan password akan disimpan menggunakan **SharedPreferences** dan pengguna akan diarahkan ke halaman profil. Jika tidak cocok, pesan kesalahan akan ditampilkan melalui **ScaffoldMessenger** sebagai Snackbar, memberitahukan bahwa username atau password yang dimasukkan salah. Tampilan tombol menampilkan teks "LOGIN" dengan gaya tertentu

```

106  onPressed: () async {
107      String enteredUsername = _usernameController.text;
108      String enteredPassword = _passwordController.text;
109
110      if (users.containsKey(enteredUsername) &&
111          users[enteredUsername] == enteredPassword) {
112          final prefs = await SharedPreferences.getInstance();
113          await prefs.setString('username', enteredUsername);
114          await prefs.setString('password', enteredPassword);
115
116          Navigator.pushReplacementNamed(
117              context,
118              Profile.routeName,
119          );
120      } else {
121          ScaffoldMessenger.of(context).showSnackBar(
122              SnackBar(
123                  backgroundColor: Color.fromARGB(255, 255, 15, 15),
124                  content: Text('Incorrect Username or Password'),
125              ),
126          );
127      }
128  },

```

B. Home

Kelas **HomeScreen** dalam kode Flutter ini merupakan tampilan utama aplikasi berita. Terdiri dari tiga bagian utama: app bar dengan tombol menu drawer, drawer untuk navigasi antar layar, dan konten utama yang menampilkan berita terkini. Saat dibuka, halaman ini akan memuat data dari layanan **fetchapi** untuk menampilkan berita terbaru. Terdapat bagian "Breaking News" yang menampilkan berita-berita terkini dalam bentuk horizontal scrollable list beserta informasi singkat. Juga terdapat bagian "News of the Day" yang menampilkan berita utama dalam bentuk gambar beserta judulnya. Drawer menyediakan navigasi ke layar-layar seperti semua berita, kategori, pengaturan, serta opsi keluar aplikasi.

1. Pengolahan Data Berita

Pengolahan Data Berita menggunakan **FutureBuilder** untuk mengambil data berita melalui API dan menampilkan loader saat data sedang diambil atau menampilkan pesan kesalahan jika terjadi kesalahan saat pengambilan data.

```

117 body: FutureBuilder(
118     future: fetchapi.getData(),
119     builder: (context, snapshot) {
120         if (snapshot.connectionState == ConnectionState.waiting) {
121             return Center(
122                 child: CircularProgressIndicator(),
123             );
124         } else if (snapshot.hasError) {
125             return Center(
126                 child: Text("Error: ${snapshot.error}"),
127             );
128         } else {
129             return ListView(padding: EdgeInsets.zero, children: [
130                 _NewsOfTheDay(article: snapshot.data!["articles"]),
131                 _BreakingNews(
132                     articles: snapshot.data!["articles"],
133                     username: username!,
134                 ),
135             ]);
136         }
137     },
138 ),

```

2. Function Ubah Date

Fungsi ubahdate memproses string tanggal menjadi format yang berbeda. Dengan menggunakan metode slicing pada string, ia membagi tanggal ke dalam tahun, bulan, hari, dan waktu (jam:menit). Menggunakan informasi tersebut, fungsi mengembalikan string baru dengan format "\$tahun-\$bulan-\$hari (\$jam:\$menit)". Sehingga, jika diberikan input tanggal, fungsi ini akan mengembalikan string dengan format yang diinginkan yang mencakup informasi tanggal dan waktu dalam format yang lebih terstruktur..

```

153 String ubahdate(String date) {
154     String dateTimeString = date;
155
156     // Ambil tahun, bulan, dan tanggal menggunakan slicing character
157     String year = dateTimeString.substring(0, 4);
158     String month = dateTimeString.substring(5, 7);
159     String day = dateTimeString.substring(8, 10);
160     String time = dateTimeString.substring(11, 16); // Ambil jam (HH:mm)
161
162     return "$year-$month-$day ($time)";
163 }
164

```

3. Function UbahAuthor

Fungsi **ubahAuthor** menerima string yang merupakan nama penulis (author). Dalam prosesnya, fungsi ini membagi string menjadi kata-kata, menyimpannya dalam list, lalu mengambil dua kata pertama dari string tersebut. Hasilnya adalah dua kata pertama dari nama penulis yang dipisahkan oleh spasi, atau nama penulis lengkap jika kata-kata tersebut tidak tersedia. Fungsi ini secara efektif mengembalikan potongan awal dari nama penulis untuk penggunaan yang lebih ringkas, misalnya, untuk menampilkan informasi penulis dalam format yang lebih singkat.

```

165 String ubahAuthor(String author) {
166     String originalString = author;
167
168     // Membagi string menjadi kata-kata
169     List<String> words = originalString.split(' ');
170
171     // Mengambil dua kata pertama
172     String firstTwoWords =
173         words.length >= 2 ? '${words[0]} ${words[1]}' : originalString;
174
175     // Tampilkan hasil
176     return firstTwoWords;
177 }

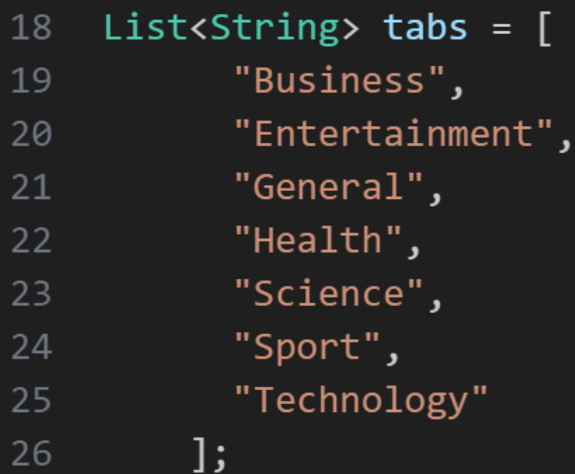
```


C. Discover Screen

Kelas ini membangun layar 'Discover' dalam aplikasi Flutter. Halaman ini menampilkan kategori berita yang dapat dipilih menggunakan TabBar dengan konten yang diambil dari API. Setiap kategori memiliki daftar berita yang ditampilkan dalam ListView yang dapat diklik untuk membaca lebih lanjut. Penggunaan FutureBuilder memungkinkan tampilan yang responsif, menampilkan loader saat data sedang diambil atau pesan kesalahan jika terjadi masalah. Setiap berita ditampilkan dengan judul, gambar, dan waktu publikasi. Secara keseluruhan, halaman ini memberikan pengalaman untuk menjelajahi berita dari berbagai kategori.

1. Daftar Kategori


Daftar **tabs** tersebut berisi kategori-kategori berita yang digunakan dalam layar "Discover" dalam aplikasi Flutter. Setiap elemen dalam list merepresentasikan satu kategori berita yang dapat dipilih oleh pengguna. Daftar ini memberikan opsi navigasi untuk melihat berita dalam kategori-kategori seperti Bisnis, Hiburan, Umum, Kesehatan, Sains, Olahraga, dan Teknologi, memungkinkan pengguna untuk menjelajahi berita berdasarkan topik-topik yang berbeda.



```
18 List<String> tabs = [  
19     "Business",  
20     "Entertainment",  
21     "General",  
22     "Health",  
23     "Science",  
24     "Sport",  
25     "Technology"  
26 ];
```

2. Variabel 'selectedIndex'

Dalam konteks ini, jika pengguna memilih kategori "Business", variabel **selectedIndex** akan diatur ke nilai 0 (indeks kategori bisnis). Penggunaan variabel ini memungkinkan aplikasi untuk mengetahui kategori mana yang sedang ditampilkan atau dipilih, memungkinkan tindakan atau perubahan tampilan yang sesuai dengan pilihan pengguna.



```
28  int selectedIndex = 0;
```

3. Fungsi setupData

Fungsi **setupData** merupakan bagian penting dalam aplikasi karena mengambil data dari API berdasarkan kategori yang dipilih. Prosesnya memanfaatkan objek **fetchapi** untuk mengambil data dan memperbarui variabel **data**. Jika respons dari API berhasil diperoleh, informasi tersebut disimpan dalam **data**. Pesan "Something error" dicetak jika respons kosong. Fungsi ini memastikan tampilan layar "Discover" selalu terkait dengan data terbaru dari API, memberikan pengalaman yang responsif dan mutakhir kepada pengguna.



```
70  Map data = {};  
71  ApiDatas fetchapi = new ApiDatas();  
72  
73  void setupData(String category) async {  
74    Map response = await fetchapi.getDataByCategory(category);  
75    if (response.isEmpty) {  
76      print("Something error");  
77    } else {  
78      print("Data Fetch Success");  
79      setState(() {  
80        data = response;  
81      });  
82    }  
83  }
```

D. Detail Artikel

Bagian dari layar tampilan artikel dalam aplikasi Flutter. **ArticleScreen** menampilkan gambar artikel sebagai latar belakang dan menampilkan judul serta deskripsi artikel di bagian atas. **_NewsBody** menampilkan informasi detail artikel seperti penulis, tanggal, dan konten artikel. Variabel **ubahdate** dan **ubahAuthor** digunakan untuk memformat tanggal dan nama penulis secara khusus. Keseluruhan kode ini bertujuan memberikan tampilan yang menarik dan informatif bagi pengguna yang membuka artikel tertentu dalam aplikasi

1. RouteName

Pada bagian ini, **static const routeName = '/article';** mendefinisikan rute yang digunakan untuk menavigasi ke layar artikel.



```
10 static const routeName = '/article';
```

2. _NewsBody Widget

Bagian ini merupakan widget untuk menampilkan informasi detail artikel seperti penulis, tanggal, judul, dan konten artikel.



```
37 class _NewsBody extends StatelessWidget {  
38   const _NewsBody({  
39     super.key,  
40     required this.article,  
41   });  
42  
43   final Map article;  
44
```

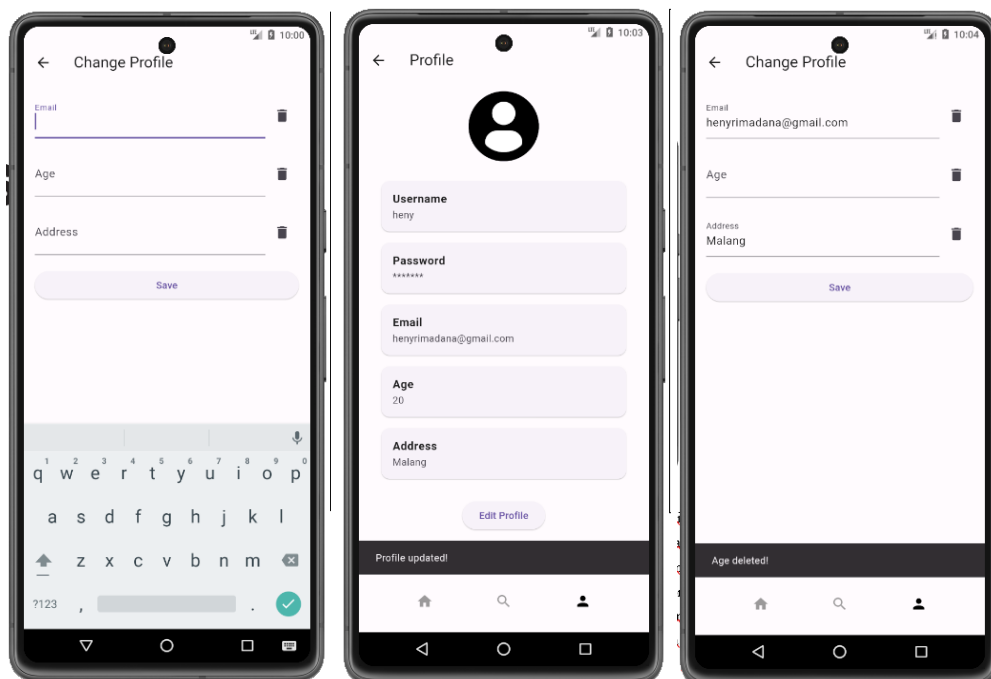
3. `_NewsHeadline` Widget

kode ini menampilkan judul dan deskripsi artikel di bagian atas tampilan artikel.

```
151 class _NewsHeadline extends StatelessWidget {  
152   const _NewsHeadline(  
153     super.key,  
154     required this.article,  
155   });  
156  
157   final Map article;  
158
```

E. Halaman Profile

Pada halaman ini memanfaatkan `SharedPreferences` untuk menyimpan dan memuat data profil pengguna, serta menampilkan informasi profil dalam sebuah layar yang mencakup informasi dasar pengguna dan memberikan opsi untuk mengedit profil melalui tombol "Edit Profile". Ini memungkinkan pengguna untuk melihat dan memperbarui informasi profil mereka dengan mudah melalui aplikasi.



1. SharedPreferences

Menggunakan SharedPreferences untuk menyimpan dan memuat data profil pengguna seperti email, usia, alamat, username, dan password saat aplikasi dimulai atau ada perubahan pada dependencies.

```
34 Future<void> _loadProfileData() async {
35     _prefs = await SharedPreferences.getInstance();
36     setState(() {
37         _email = _prefs.getString('email') ?? '';
38         _age = _prefs.getString('age') ?? '';
39         _address = _prefs.getString('address') ?? '';
40         _username = _prefs.getString('username') ?? '';
41         _password = _prefs.getString('password') ?? '';
42     });
43 }
```

2. Navigation

Terdapat tombol "Edit Profile" yang, saat ditekan, menggunakan **Navigator.push** untuk berpindah ke halaman **ChangeProfile**.

```
101 child: ElevatedButton(
102     onPressed: () {
103         Navigator.push(
104             context,
105             MaterialPageRoute(builder: (context) => ChangeProfile()),
106         );
107     },
108     child: Text('Edit Profile'),
109 ),
```

F. Halaman CRUD

Pada halaman ini memanfaatkan SharedPreferences untuk menyimpan dan memuat data profil pengguna, serta menampilkan informasi profil dalam sebuah layar yang mencakup informasi dasar pengguna dan memberikan opsi untuk mengedit profil melalui tombol "Edit Profile". Ini memungkinkan pengguna untuk melihat dan memperbarui informasi profil mereka dengan mudah melalui aplikasi.

1. RouteName & Stateful Widget:

ChangeProfile adalah **StatefulWidget** yang memiliki **routeName** untuk navigasi ke halaman perubahan profil.

```
5 class ChangeProfile extends StatefulWidget {
6   static const routeName = '/change_profile';
7
8   @override
9   _ChangeProfileState createState() => _ChangeProfileState();
10 }
```

2. State & Controllers:

Terdapat tiga controller yang digunakan untuk mengelola input pengguna: **_emailController**, **_ageController**, dan **_addressController**. Ini digunakan untuk menyimpan nilai yang diisi pengguna di dalam form perubahan profil.

```
12 class _ChangeProfileState extends State<ChangeProfile> {
13   late TextEditingController _emailController;
14   late TextEditingController _ageController;
15   late TextEditingController _addressController;
16   late SharedPreferences _prefs;
17
18   @override
19   void initState() {
20     super.initState();
21     _emailController = TextEditingController();
22     _ageController = TextEditingController();
23     _addressController = TextEditingController();
24     _loadProfileData();
25   }
```

3. Manipulasi Data

SharedPreferences digunakan untuk menyimpan data profil pengguna. Method **_loadProfileData** mengambil data profil yang disimpan dan memuatnya ke dalam controller.

```

18  @override
19  void initState() {
20    super.initState();
21    _emailController = TextEditingController();
22    _ageController = TextEditingController();
23    _addressController = TextEditingController();
24    _loadProfileData();
25  }
26
27  Future<void> _loadProfileData() async {
28    _prefs = await SharedPreferences.getInstance();
29    setState(() {
30      _emailController.text = _prefs.getString('email') ?? '';
31      _ageController.text = _prefs.getString('age') ?? '';
32      _addressController.text = _prefs.getString('address') ?? '';
33    });
34  }

```

4. CRUD

Fungsi **_saveProfile** digunakan untuk menyimpan perubahan yang dibuat pengguna pada profil. Fungsi **_deleteProfileData** digunakan untuk menghapus data profil berdasarkan jenisnya (email, usia, atau alamat).

```

36  Future<void> _saveProfile() async {
37    await _prefs.setString('email', _emailController.text);
38    await _prefs.setString('age', _ageController.text);
39    await _prefs.setString('address', _addressController.text);
40    ScaffoldMessenger.of(context).showSnackBar(
41      SnackBar(content: Text('Profile updated!')),
42    );
43    // Navigate back to Profile after saving
44    Navigator.pop(context);
45  }
46
47  Future<void> _deleteProfileData(String field) async {
48    switch (field) {
49      case 'email':
50        await _prefs.remove('email');
51        _emailController.clear(); // Clear email controller
52        ScaffoldMessenger.of(context).showSnackBar(
53          SnackBar(content: Text('Email deleted!')),
54        );
55        break;
56      case 'age':
57        await _prefs.remove('age');
58        _ageController.clear(); // Clear age controller
59        ScaffoldMessenger.of(context).showSnackBar(
60          SnackBar(content: Text('Age deleted!')),
61        );
62        break;
63      case 'address':
64        await _prefs.remove('address');
65        _addressController.clear(); // Clear address controller
66        ScaffoldMessenger.of(context).showSnackBar(
67          SnackBar(content: Text('Address deleted!')),
68        );
69        break;
70      default:
71        break;
72    }
73  }

```

G. Kode Program

Kode program selengkapnya desain UI aplikasi berita Jawa Pos Radar Malang ini dapat diakses melalui tautan berikut:

<https://github.com/henryrimadana/UASPRAKMOBILE>