Nama : zaidan
Kelas : SIB-3D
NIM : 2241760130
Github : https://github.com/arabgg/P_Mobile

How to Build an API with Laravel Breeze in Laravel 11

A step-by-step guide on building a simple API with authentication using Laravel Breeze in Laravel 11.

Step 1: Install Laravel

Pertama, buat proyek Laravel baru menggunakan penginstal Laravel atau

Composer. laravel new api-breeze

Or via Composer composer create-project
laravel/laravel api-breeze cd apibreeze

Langkah	Deskripsi
1	PS C:\laragon\www\api-breeze> composer require laravel/breeze --dev Cannot use laravel/breeze's latest version v2.2.5 as it requires php ./composer.json has been updated Running composer update laravel/breeze Loading composer repositories with package information Updating dependencies Lock file operations: 1 install, 0 updates, 0 removals - Locking laravel/breeze (v1.29.1) Writing lock file Installing dependencies from lock file (including require-dev) Package operations: 1 install, 0 updates, 0 removals INFO: Could not find files for the given pattern(s).

Step 2: Install Laravel Breeze

Next, install Laravel Breeze and its dependencies.

composer require laravel/breeze --dev

php artisan breeze:install api

Perintah ini akan menginstal Breeze dan menyiapkan perancah yang diperlukan untuk autentikasi API.

Langkah	Deskripsi
1	PS C:\laragon\www\api-breeze> php artisan breeze:install api INFO Breeze scaffolding installed successfully. PS C:\laragon\www\api-breeze>

Step 3: Configure the Database and Run Migrations

1. Update your .env file with your database credentials:

DB_CONNECTION=mysql

DB_HOST=127.0.0.1

DB_PORT=3306

DB_DATABASE=laravel11_api

DB_USERNAME=root

DB_PASSWORD=

1. Run the migrations to set up your database tables: php artisan migrate

Langkah	Deskripsi
1	DB_CONNECTION=mysql DB_HOST=127.0.0.1 DB_PORT=3306 DB_DATABASE=laravel11_api DB_USERNAME=root DB_PASSWORD=
2	PS C:\laragon\www\api-breeze> php artisan migrate INFO Preparing database. Creating migration table ..... 131ms DONE INFO Running migrations. 2014_10_12_000000_create_users_table ..... 70ms DONE 2014_10_12_100000_create_password_reset_tokens_table ..... 6ms DONE 2019_08_19_000000_create_failed_jobs_table ..... 42ms DONE 2019_12_14_000001_create_personal_access_tokens_table ..... 27ms DONE

Step 4: Create Authentication Endpoints

Laravel Breeze provides the necessary endpoints for registration, login, and logout. The routes are defined in routes/api.php.

use App\Http\Controllers\Auth\AuthenticatedSessionController;

use App\Http\Controllers\Auth\RegisteredUserController; use

Illuminate\Support\Facades\Route;

Route::post('/register', [RegisteredUserController::class, 'store']); Route::post('/login',

[AuthenticatedSessionController::class, 'store']);

Route::post('/logout', [AuthenticatedSessionController::class, 'destroy'])->middleware('auth:sanctum');

Langkah	Deskripsi

1	routes > api.php
	`<?php`
	
	`use App\Http\Controllers\Auth\AuthenticatedSessionController;`
	`use App\Http\Controllers\Auth\RegisteredUserController;`
	`use Illuminate\Support\Facades\Route;`
	
	`Route::post('/register', [RegisteredUserController::class, 'store']);  // Rute untuk registrasi`
	`Route::post('/login', [AuthenticatedSessionController::class, 'store']);  // Rute untuk login`
	`Route::post('/logout', [AuthenticatedSessionController::class, 'destroy'])`
	`    ->middleware('auth:sanctum');  // Rute untuk logout yang dilindungi autentikasi`

Step 5: Update Controllers

Modify the RegisteredUserController and AuthenticatedSessionController to return JSON responses.

RegisteredUserController.php

namespace App\Http\Controllers\Auth;

use App\Models\User;

use Illuminate\Auth\Events\Registered;

use Illuminate\Http\Request; use

Illuminate\Support\Facades\Hash;

use Illuminate\Validation\Rules;

```
use App\Http\Controllers\Controller;

class RegisteredUserController extends Controller
{     public     function     store(Request
   $request)
   {
      $request->validate([
         'name' => ['required', 'string', 'max:255'],
         'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
         'password' => ['required', 'confirmed', Rules\Password::defaults()],
      ]);

      $user = User::create([
         'name' => $request->name,
         'email' => $request->email,
         'password' => Hash::make($request->password),
      ]);

      event(new Registered($user));

      $token = $user->createToken('auth_token')->plainTextToken;

      return response()->json([
         'access_token' => $token,
         'token_type' => 'Bearer', 'user'
         => $user
      ]);
```

```
    }
}
```

AuthenticatedSessionController.php

```
namespace App\Http\Controllers\Auth;

use Illuminate\Http\Request; use
Illuminate\Support\Facades\Auth; use
App\Http\Controllers\Controller;

class AuthenticatedSessionController extends Controller
{ public function store(Request
  $request)
  {
    $request->validate([
      'email' => ['required', 'string', 'email'], 'password' =>
      ['required', 'string'],
    ]);

if (!Auth::attempt($request->only('email', 'password'))) {
      return response()->json(['message' => 'Invalid login
                                credentials'], 401);
    }

    $user = Auth::user();
    $token = $user->createToken('auth_token')->plainTextToken;

    return response()->json([
```

```
        'access_token' => $token,
        'token_type' => 'Bearer', 'user'
        => $user,
        'status' => 'Login successful',
    ]);
  }

  public function destroy(Request $request)
  {
    $request->user()->currentAccessToken()->delete();

    return response()->json(['message' => 'Logout
  successful']); }
}
```

Langkah	Deskripsi

1	`<?php namespace App\Http\Controllers\Auth; use Illuminate\Http\Request; use Illuminate\Support\Facades\Auth; use App\Http\Controllers\Controller;`
	
	`class AuthenticatedSessionController extends Controller {`

```
    // Metode untuk menangani login pengguna
public function store(Request $request)
    {
        // Validasi input dari pengguna
        $request->validate([
            'email' => ['required', 'string', 'email'],
            'password' => ['required', 'string'],
        ]);

        // Mencoba autentikasi pengguna         if
(!Auth::attempt($request->only('email', 'password'))) {
// Jika kredensial tidak valid, kembalikan respon error
return response()->json(['message' => 'Invalid login credentials'],
401);
        }

        // Jika login berhasil, ambil data pengguna yang sedang login
        $user = Auth::user();

        // Buatkan token untuk pengguna yang berhasil login
        $token = $user->createToken('auth_token')->plainTextToken;
        // Kembalikan respons dalam format JSON dengan token dan data
pengguna         return response()->json([             'access_token'
=> $token,
            'token_type' => 'Bearer',
            'user' => $user,
            'status' => 'Login successful',
        ]);
    }

    // Metode untuk logout pengguna
    public function destroy(Request $request)
    {
        // Menghapus token yang digunakan saat ini
        $request->user()->currentAccessToken()->delete();
        // Kembalikan respons sukses logout        return response()-
>json(['message' => 'Logout successful']);     }
}




<?php namespace
App\Http\Controllers\Auth;
 use
App\Models\User;
use Illuminate\Auth\Events\Registered; use Illuminate\Http\Request; use
Illuminate\Support\Facades\Hash;
```

2

```php
use Illuminate\Validation\Rules; use
App\Http\Controllers\Controller;


class RegisteredUserController extends Controller
{ public function store(Request
$request)
{
$request->validate([
'name' => ['required', 'string', 'max:255'],
'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
'password' => ['required', 'confirmed', Rules\Password::defaults()],
]);


$user = User::create([
'name' => $request->name, 'email' => $request->email,
'password' => Hash::make($request->password),
]);

event(new Registered($user));

$token = $user->createToken('auth_token')->plainTextToken;


return response()->json([

'access_token' => $token, 'token_type' => 'Bearer', 'user' => $user
]);
}
}
```

Step 5: Run Laravel App

php artisan serve

Step 6: Check following API

Test Your API with Thunder Client

POST http://127.0.0.1:8000/api/register Send

Query Headers 2 Auth Body 1 Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
{
"name": "john maths9",
"email": "johnmaths9@example.com",
"password": "password123",
"password_confirmation": "password123"
}
```

Status: 200 OK Size: 0 Bytes Time: 1.17 s

Response Headers 9 Cookies 1 Results Docs


```
{
  "access_token": "2|qqL5BsuBQR0X9hWTTHij2qO92YydeogDFEikr18Z2f0ff604",
  "token_type": "Bearer",
  "user": {
    "name": "john maths9",
    "email": "johnmaths9@example.com",
    "updated_at": "2024-07-12T14:56:45.000000Z",
    "created_at": "2024-07-12T14:56:45.000000Z",
    "id": 2
  }
}
```


Response Chart

POST http://127.0.0.1:8000/api/login Send

Query Headers 2 Auth Body 1 Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

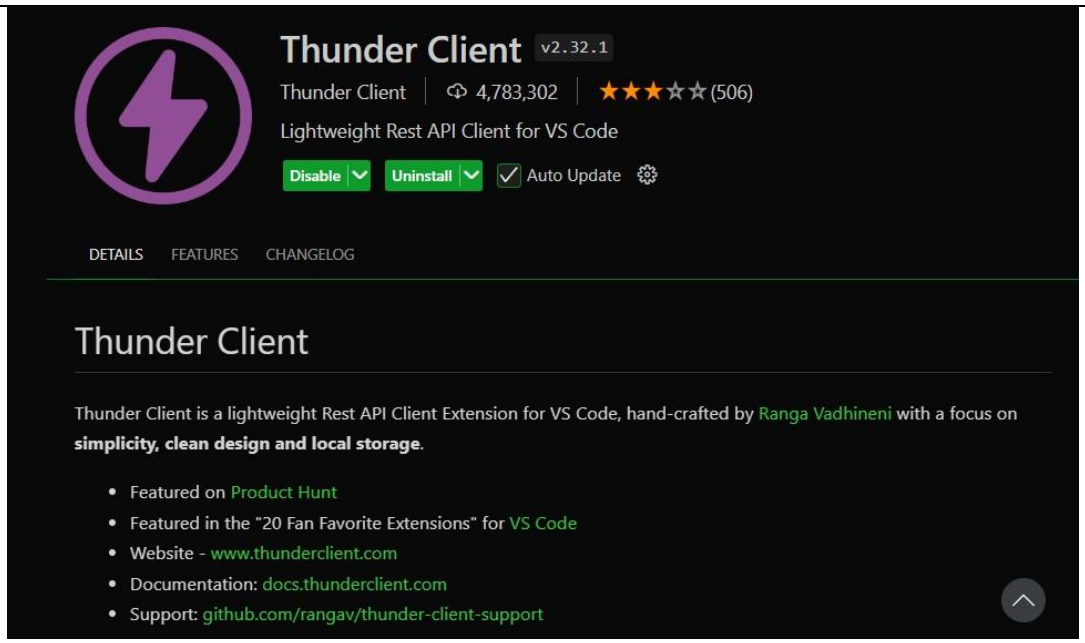
```
1 {
2 "email": "johnmaths9@example.com",
3 "password": "password123"
4 }
```

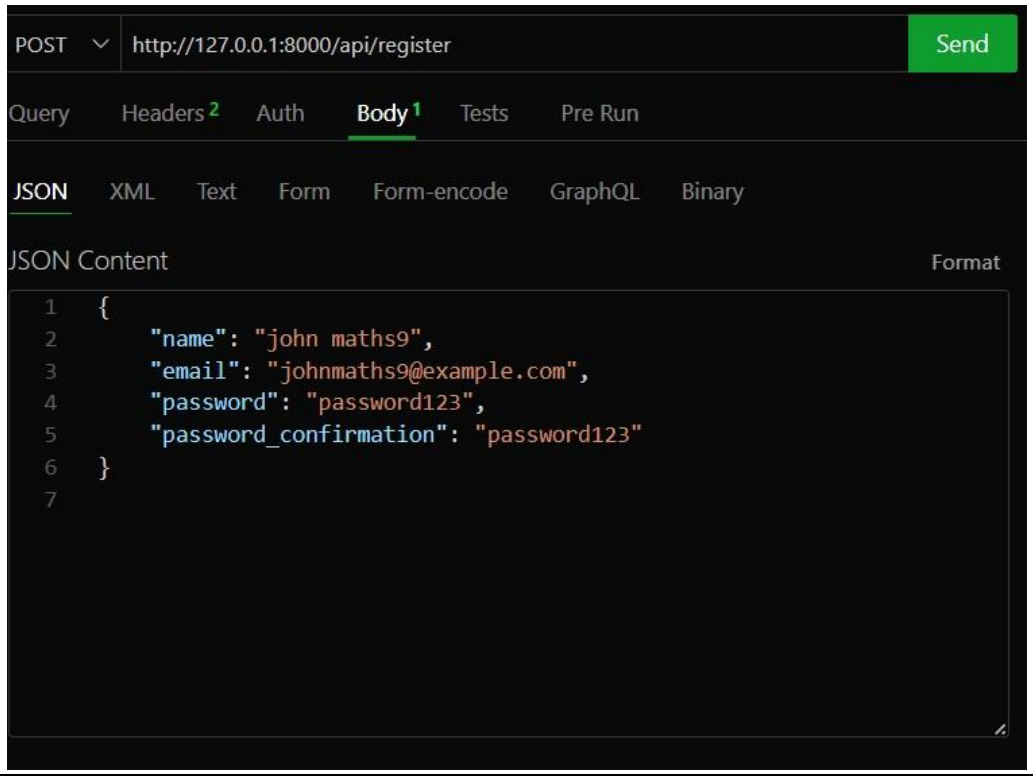
Status: 200 OK Size: 0 Bytes Time: 651 ms

Response Headers 9 Cookies 1 Results Docs {}


```
1 {
2   "access_token": "5|YPwsETaDLzgQdeQ6OPM64QTNrtC0LI5G10y5BAiea954de8e",
3   "token_type": "Bearer",
4   "user": {
5     "id": 2,
6     "name": "john maths9",
7     "email": "johnmaths9@example.com",
8     "email_verified_at": null,
9     "created_at": "2024-07-12T14:56:45.000000Z",
10    "updated_at": "2024-07-12T14:56:45.000000Z"
11  },
12  "status": "Login successful"
13 }
```


Langkah	Deskripsi

1	 Thunder Client v2.32.1 Thunder Client 4,783,302 ★★★☆☆ (506) Lightweight Rest API Client for VS Code Disable Uninstall Auto Update DETAILS FEATURES CHANGELOG Thunder Client Thunder Client is a lightweight Rest API Client Extension for VS Code, hand-crafted by Ranga Vadhineni with a focus on simplicity, clean design and local storage. Featured on Product Hunt Featured in the "20 Fan Favorite Extensions" for VS Code Website - www.thunderclient.com Documentation: docs.thunderclient.com Support: github.com/rangav/thunder-client-support

2	 POST http://127.0.0.1:8000/api/register Send
	Query Headers 2 Auth Body 1 Tests Pre Run
	JSON XML Text Form Form-encode GraphQL Binary
	JSON Content Format
	{
	"name": "john maths9",
	"email": "johnmaths9@example.com",
	"password": "password123",
	"password_confirmation": "password123"
	}

<table>
<tr>
<td></td>
<td>
Response Headers 10 Cookies Results Docs { }

```
{
  "access_token": "1|Se9pV4CC6iJ1b1b7qZ5ezHbGRuUY5WVUY9lpuOjL4c8ae1fa",
  "token_type": "Bearer",
  "user": {
    "name": "john maths9",
    "email": "johnmaths9@example.com",
    "updated_at": "2024-11-26T03:34:50.000000Z",
    "created_at": "2024-11-26T03:34:50.000000Z",
    "id": 1
  }
}
```

</td>
</tr>
<tr>
<td>3</td>
<td>
POST http://127.0.0.1:8000/api/login Send
Query Headers 2 Auth Body 1 Tests Pre Run
JSON XML Text Form Form-encode GraphQL Binary
JSON Content Format

```
{
"email": "johnmaths9@example.com",
"password": "password123"
}
```

</td>
</tr>
</table>

Status: 200 OK Size: 300 Bytes Time: 510 ms

Response Headers 10 Cookies Results Docs {}


```
{
  "access_token": "2|b1JAIwtNzAKE3u0gef0dt9HpqHBy2nmwqffPB1INd0603788",
  "token_type": "Bearer",
  "user": {
    "id": 1,
    "name": "john maths9",
    "email": "johnmaths9@example.com",
    "email_verified_at": null,
    "created_at": "2024-11-26T03:34:50.000000Z",
    "updated_at": "2024-11-26T03:34:50.000000Z"
  },
  "status": "Login successful"
}
```

Pembuatan Aplikasi Mobile Flutter,
Langkah 1: Persiapan Proyek Flutter

1. **Buat Proyek Flutter Baru:** bash flutter

create my_ flutter_app cd my_ _flutter_app

2. **Tambahkan Dependencies:**

Buka **pubspec.yaml** dan tambahkan beberapa dependencies yang

diperlukan: yaml dependencies: flutter:

sdk: flutter

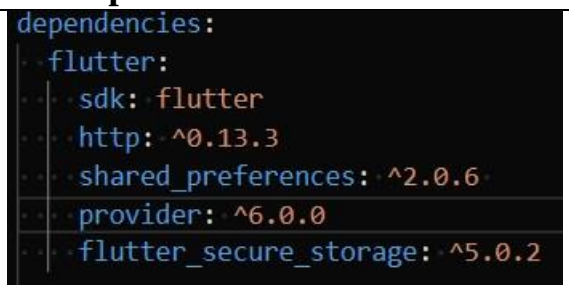
http: ^0.13.3 shared_preferences:

^2.0.6 provider:

^6.0.0

flutter_secure_storage: ^5.0.2

Jalankan **flutter pub get** untuk mengunduh dependencies.

Langkah	Deskripsi
1	
2	

Buat Splashscreen dengan animasi dari Lottie File Langkah

2: Mengatur Struktur Proyek

Buat folder berikut untuk mengatur kode Anda dengan lebih baik:

- **lib/screens/** untuk menyimpan file layar (UI).
- **lib/services/** untuk layanan HTTP dan manajemen API.
- **lib/models/** untuk model data.
- **lib/providers/** untuk manajemen state menggunakan Provider.

Langkah 3: Membuat Model Pengguna

Buat file **user_model.dart** di **lib/models/**: dart class User { final int id; final String name; final String email;

```
  User({required this.id, required this.name, required this.email});

  factory User.fromJson(Map<String, dynamic> json) { return
    User(
      id:          json['id'],
      name:
      json['name'],
      email:
      json['email'],
    );
  }
}
```

Langkah	Deskripsi

1	`class User { final int id; final String name; final String email;`
	
	`User({required this.id, required this.name, required this.email});`
	
	`factory User.fromJson(Map<String, dynamic> json) { return User( id: json['id'], name: json['name'], email: json['email'],`

	`);`
	`}`
	`}`

Langkah 4: Membuat Layanan API Buat file **auth_service.dart** di **lib/services/**: dart

```
import 'dart:convert'; import 'package:http/http.dart' as http; import
'package:flutter_secure_storage/flutter_secure_storage.dart';    import
'../models/user_model.dart';

class AuthService {
  final String apiUrl = 'http://your-laravel-api-url.com/api'; final
  storage = FlutterSecureStorage();

  Future<bool> login(String email, String password) async {
   final response = await http.post( Uri.parse('$apiUrl/login'),
   headers:     {'Content-Type':     'application/json'},     body:
   jsonEncode({'email': email, 'password': password}),
   );

   if (response.statusCode == 200) {
    final      data      =      jsonDecode(response.body);      await
    storage.write(key: 'token', value: data['token']); return true;
   }    else    {    return
    false;
   }
  }
  Future<User?> getProfile() async { final token
   =    await    storage.read(key:    'token');    final
   response = await http.get(
    Uri.parse('$apiUrl/profile'), headers:
    {
  'Content-Type': 'application/json', 'Authorization':
```

```
    'Bearer $token',
  },
);

if (response.statusCode == 200) {
  final data = jsonDecode(response.body); return
  User.fromJson(data['user']);
} else { return
  null;
}
}

Future<void> logout() async {
  await storage.delete(key:
  'token');
}
}
```

Langkah	Deskripsi

1	`import 'dart:convert'; import 'package:http/http.dart' as http;`
	`import`
	`'package:flutter_secure_storage/flutter_secure_storage.dart';`
	`import '../models/user_model.dart';`

```
class AuthService { final String apiUrl = 'http://your-laravel-
api-url.com/api'; final storage = const FlutterSecureStorage();

Future<bool> login(String email, String password) async { final
response = await http.post( Uri.parse('$apiUrl/login'),
headers: {'Content-Type': 'application/json'}, body:
jsonEncode({'email': email, 'password': password}), );

if (response.statusCode == 200) { final data =
jsonDecode(response.body); await storage.write(key: 'token',
value: data['token']); return true;
} else { return false;
}
}

Future<User?> getProfile() async { final token = await
storage.read(key: 'token'); final response = await http.get(
Uri.parse('$apiUrl/profile'), headers: {
'Content-Type': 'application/json', 'Authorization': 'Bearer
$token',
},
);

if (response.statusCode == 200) {
final data = jsonDecode(response.body); return
User.fromJson(data['user']);
} else { return null;
}
}


Future<void> logout() async { await
storage.delete(key: 'token');
}
}
```

Langkah 5: Menyusun State Management dengan Provider

Buat file **auth_provider.dart** di **lib/providers/**: dart

```
import 'package:flutter/material.dart'; import
'../models/user_model.dart'; import
'../services/auth_service.dart';

class AuthProvider with ChangeNotifier {
  final AuthService _authService = AuthService(); User?
  _user;

  User? get user => _user;

  Future<bool> login(String email, String password) async { bool
   success = await _authService.login(email, password); if (success)
   {
    _user = await _authService.getProfile();
    notifyListeners();
   } return
   success;
  }

  Future<void> logout() async { await
   _authService.logout();
   _user = null;
   notifyListeners();
  }

  Future<void> loadUser() async {
```

_user = await _authService.getProfile();

notifyListeners(); }

}

Langkah	Deskripsi
1	

```
import 'package:flutter/material.dart';
import '../models/user_model.dart';
import '../services/auth_service.dart';
 class AuthProvider with ChangeNotifier { final AuthService
_authService = AuthService(); User? _user;
User? get user => _user;


Future<bool> login(String email, String password) async { bool
success = await _authService.login(email, password); if (success) {
_user = await _authService.getProfile(); notifyListeners();
} return
success;
}


Future<void> logout() async { await _authService.logout();
_user = null;

notifyListeners();
}


Future<void> loadUser() async {
_user = await _authService.getProfile(); notifyListeners();
}
}
```

Langkah 6: Membuat Halaman Login

Buat file **login_screen.dart** di **lib/screens/**: dart import 'package:flutter/material.dart'; import 'package:provider/provider.dart'; import

```
'../providers/auth_provider.dart';

class LoginScreen extends StatelessWidget {
  final TextEditingController emailController = TextEditingController(); final
  TextEditingController passwordController = TextEditingController();

  @override
  Widget build(BuildContext context) {
   final authProvider = Provider.of<AuthProvider>(context);

   return Scaffold(
    appBar: AppBar(title: Text('Login')), body:
    Padding(                    padding:
     EdgeInsets.all(16.0),   child:
     Column( children: [
       TextField(      controller:      emailController,
        decoration:         InputDecoration(labelText:
        'Email'),
       ),
       TextField(      controller:      passwordController,
        decoration:                 InputDecoration(labelText:
        'Password'), obscureText: true,
       ),
       SizedBox(height:  20),  ElevatedButton(
       onPressed:  ()  async  {  bool  success  =
       await authProvider.login(
```

```
          emailController.text,
          passwordController.text,
         );
         if (success) {
       Navigator.of(context).pushReplacementNamed('/profile');
         } else { ScaffoldMessenger.of(context).showSnackBar(SnackBar(
         content: Text('Login failed!'),
          ));
         }
        },                child:
        Text('Login'),
       ),
      ],
     ),
    ),
   );
  }
}
```

Langkah	Deskripsi

1	

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import '../providers/auth_provider.dart';
 class LoginScreen extends StatelessWidget { final
TextEditingController emailController =
TextEditingController(); final
TextEditingController passwordController =
TextEditingController();

  LoginScreen({super.key});

@override
Widget build(BuildContext context) { final authProvider
= Provider.of<AuthProvider>(context);


return Scaffold(
  appBar: AppBar(title: const Text('Login')), body:
Padding( padding: const EdgeInsets.all(16.0), child:
Column( children: [ TextField( controller:
emailController, decoration: const
InputDecoration(labelText: 'Email'),
),
```

```
TextField( controller: passwordController, decoration:
const InputDecoration(labelText: 'Password'),
obscureText: true,
), const SizedBox(height: 20), ElevatedButton( onPressed: ()
async { bool success = await authProvider.login(
emailController.text, passwordController.text,
);
if (success) {
Navigator.of(context).pushReplacementNamed('/profile');
} else {

ScaffoldMessenger.of(context).showSnackBar(const SnackBar( content:
Text('Login failed!'),
));
} }, child: const
Text('Login'),
),
],
),
),
);
}
}
```

Langkah 7: Membuat Halaman Profil

Buat file **profile_screen.dart** di **lib/screens/**: dart

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

'package:provider/provider.dart'; import

'../providers/auth_provider.dart';

class ProfileScreen extends StatelessWidget { @override

Widget build(BuildContext context) { final

authProvider = Provider.of<AuthProvider>(context);

final user = authProvider.user;

```
    return Scaffold(
      appBar: AppBar( title:
      Text('Profile'), actions: [
        IconButton(           icon:
          Icon(Icons.logout),
          onPressed: () {
            authProvider.logout();
        Navigator.of(context).pushReplacementNamed('/login');
          },
        ),
      ],
    ),
    body:         Center(
      child: user != null
          ?              Column(              mainAxisAlignment:
              MainAxisAlignment.center, children: [
                Text('Welcome, ${user.name}!'),
                Text('Email: ${user.email}'),
              ],
            )
        : CircularProgressIndicator(),
    ),
  );
 }
}
```

Langkah	Deskripsi

1	

```
import 'package:flutter/material.dart'; import
'package:provider/provider.dart'; import
'../providers/auth_provider.dart';
 class ProfileScreen extends
StatelessWidget {   const
ProfileScreen({super.key});
 @override
Widget build(BuildContext context) { final
authProvider = Provider.of<AuthProvider>(context);
  final user =
authProvider.user;


return Scaffold( appBar: AppBar( title: const
Text('Profile'), actions: [ IconButton( icon: const
Icon(Icons.logout), onPressed: () { authProvider.logout();
Navigator.of(context).pushReplacementNamed('/login');
},
),
], ), body:
Center( child:
user != null
? Column(
mainAxisAlignment: MainAxisAlignment.center, children: [
Text('Welcome, ${user.name}!'),
Text('Email: ${user.email}'),
],
)

: const CircularProgressIndicator(),
),
);
}
}
```

Langkah 8: Mengatur Routing dan Provider

Buka **main.dart** dan atur routing serta Provider:

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

'package:provider/provider.dart'; import

```
'screens/splash_screen.dart'; import
'screens/login_screen.dart'; import
'screens/profile_screen.dart'; import
'providers/auth_provider.dart';

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) { return
    MultiProvider(
      providers: [
      ChangeNotifierProvide
      r(create: (_) =>
      AuthProvider()),

      ],
      child: MaterialApp( title:
        'Flutter App',
        theme: ThemeData(
          primarySwatch: Colors.blue,
        ),
        initialRoute:
        '/', routes: {
          '/': (context) => SplashScreen(), '/login':
          (context) => LoginScreen(), '/profile':
```

```
(context) => ProfileScreen(),
},
),
);
}
}
```

Langkah	Deskripsi

1	`import 'package:flutter/material.dart';`
	`import 'package:provider/provider.dart';`
	`import 'screens/splash_screen.dart';`
	`import 'screens/login_screen.dart';`
	`import 'screens/profile_screen.dart';`
	`import 'providers/auth_provider.dart';`
	
	`void main() { runApp(MyApp());`
	`}`
	
	`class MyApp extends StatelessWidget {`
	`const MyApp({super.key});`
	` @override`
	`Widget build(BuildContext context) { return MultiProvider(`
	`providers: [`

	`ChangeNotifierProvider(create: (_) => AuthProvider()),`
	`], child: MaterialApp( title: 'Flutter App', theme:`
	`ThemeData( primarySwatch: Colors.blue,`
	`), initialRoute: '/', routes: {`
	`'/': (context) =>`
	`SplashScreen(),`
	`'/login': (context) => LoginScreen(),`
	`'/profile': (context) => ProfileScreen(),`
	`},`
	`),`
	`);`
	`}`
	`}`

Langkah G: Menyiapkan Splash Screen Buat file **splash_screen.dart** di **lib/screens/**: dart

```
import 'package:flutter/material.dart'; import
'package:provider/provider.dart'; import
'../providers/auth_provider.dart';
class SplashScreen extends StatefulWidget { @override
  _SplashScreenState createState() => _SplashScreenState();
}

class _SplashScreenState extends State<SplashScreen> {
  @override
  void initState() {
    super.initState();
    _checkLoginStatus();
  }

  void _checkLoginStatus() async {
    final authProvider = Provider.of<AuthProvider>(context, listen: false);
    await authProvider.loadUser(); if (authProvider.user != null) {
      Navigator.of(context).pushReplacementNamed('/profile');
    } else {
      Navigator.of(context).pushReplacementNamed('/login');
    }
  }

  @override
  Widget build(BuildContext context) {
```

```
    return Scaffold(
      body: Center(
      child: Text('My Flutter App', style: TextStyle(fontSize:
                                                          24)),
      ),
    );
  }
}
```

Langkah	Deskripsi
1	`import 'package:flutter/material.dart'; import 'package:provider/provider.dart'; import`

```
'../providers/auth_provider.dart';
  class SplashScreen extends
StatefulWidget {   const
SplashScreen({super.key});
 @override
_SplashScreenState createState() => _SplashScreenState();
}

class _SplashScreenState extends State<SplashScreen> { @override
void initState() { super.initState();
_checkLoginStatus();
}

void _checkLoginStatus() async { final authProvider =
Provider.of<AuthProvider>(context, listen:
false); await authProvider.loadUser();
if (authProvider.user != null) {
Navigator.of(context).pushReplacementNamed('/profile');
} else {
Navigator.of(context).pushReplacementNamed('/login');
}
}


@override
Widget build(BuildContext context) {
  return const Scaffold( body: Center( child: Text('My
Flutter App', style: TextStyle(fontSize: 24)),
),
);




}
}
```

Langkah 10: Menjalankan Aplikasi

Pastikan API Laravel Anda sudah berjalan dan endpoint login serta profil sudah tersedia. Jalankan aplikasi Flutter: