
Flutter Integration Guide

Quick guide for using the Backend with Flutter

Required Dependencies

Add to `pubspec.yaml`:

```
YAML
dependencies:
  flutter:
    sdk: flutter
  http: ^1.1.0
  shared_preferences: ^2.2.2
  image_picker: ^1.0.7
  dio: ^5.4.0 # Optional - alternative to http
```

Setup - API Client Configuration

1. Create API Config

```
DART
// lib/config/api_config.dart
class ApiConfig {
  // Change IP for real device
  static const String baseUrl = 'http://10.0.2.2:8000'; // Android Emulator
  // static const String baseUrl = 'http://127.0.0.1:8000'; // iOS Simulator
  // static const String baseUrl = 'http://192.168.1.x:8000'; // Real Device

  static const String authRegister = '/auth/register/';
  static const String authLogin = '/auth/login/';
  static const String authLogout = '/auth/logout/';
  static const String authProfile = '/auth/profile/';
  static const String authProfileUpdate = '/auth/profile/update/';
  static const String authPasswordChange = '/auth/password/change/';
  static const String authTokenRefresh = '/auth/token/refresh/';

  static const String uploadNew = '/api/uploads/new/';
```

```
static const String uploadsList = '/api/uploads/';  
}
```

Auth Service

DART

```
// lib/services/auth_service.dart  
import 'dart:convert';  
import 'package:http/http.dart' as http;  
import 'package:shared_preferences/shared_preferences.dart';  
import '../config/api_config.dart';  
  
class AuthService {  
  // Save tokens  
  Future<void> saveTokens(String accessToken, String refreshToken) async {  
    final prefs = await SharedPreferences.getInstance();  
    await prefs.setString('access_token', accessToken);  
    await prefs.setString('refresh_token', refreshToken);  
  }  
  
  // Get tokens  
  Future<String?> getAccessToken() async {  
    final prefs = await SharedPreferences.getInstance();  
    return prefs.getString('access_token');  
  }  
  
  Future<String?> getRefreshToken() async {  
    final prefs = await SharedPreferences.getInstance();  
    return prefs.getString('refresh_token');  
  }  
  
  // Clear tokens  
  Future<void> clearTokens() async {  
    final prefs = await SharedPreferences.getInstance();  
    await prefs.remove('access_token');  
    await prefs.remove('refresh_token');  
  }  
  
  // Register new user  
  Future<Map<String, dynamic>> register({  
    required String username,  
    required String email,  
    required String password,  
    String? firstName,  
    String? lastName,  
  }) async {  
    final url = Uri.parse('${ApiConfig.baseUrl}${ApiConfig.authRegister}');  
  
    final response = await http.post(  
      url,
```

```
        headers: {'Content-Type': 'application/json'},
        body: jsonEncode({
            'username': username,
            'email': email,
            'password': password,
            'password2': password,
            if (firstName != null) 'first_name': firstName,
            if (lastName != null) 'last_name': lastName,
        }),
    );

    if (response.statusCode == 201) {
        final data = jsonDecode(response.body);
        await saveTokens(
            data['tokens']['access'],
            data['tokens']['refresh'],
        );
        return data;
    } else {
        throw Exception('Registration failed: ${response.body}');
    }
}

// Login
Future<Map<String, dynamic>> login({
    required String username,
    required String password,
}) async {
    final url = Uri.parse('${ApiConfig.baseUrl}${ApiConfig.authLogin}');

    final response = await http.post(
        url,
        headers: {'Content-Type': 'application/json'},
        body: jsonEncode({
            'username': username,
            'password': password,
        })),
    );

    if (response.statusCode == 200) {
        final data = jsonDecode(response.body);
        await saveTokens(data['access'], data['refresh']);
        return data;
    } else {
        throw Exception('Login failed: ${response.body}');
    }
}

// Logout
Future<void> logout() async {
    final refreshToken = await getRefreshToken();
    if (refreshToken == null) return;

    final url = Uri.parse('${ApiConfig.baseUrl}${ApiConfig.authLogout}');
    final accessToken = await getAccessToken();

    await http.post(
```

```
        url,
        headers: {
            'Content-Type': 'application/json',
            'Authorization': 'Bearer $accessToken',
        },
        body: jsonEncode({'refresh': refreshToken}),
    );

    await clearTokens();
}

// Get user profile
Future<Map<String, dynamic>> getProfile() async {
    final url = Uri.parse('${ApiConfig.baseUrl}${ApiConfig.authProfile}');
    final token = await getAccessToken();

    final response = await http.get(
        url,
        headers: {
            'Authorization': 'Bearer $token',
        },
    );

    if (response.statusCode == 200) {
        return jsonDecode(response.body);
    } else {
        throw Exception('Failed to get profile: ${response.body}');
    }
}

// Update profile
Future<Map<String, dynamic>> updateProfile({
    String? email,
    String? firstName,
    String? lastName,
}) async {
    final url = Uri.parse('${ApiConfig.baseUrl}${ApiConfig.authProfileUpdate}');
    final token = await getAccessToken();

    final response = await http.patch(
        url,
        headers: {
            'Content-Type': 'application/json',
            'Authorization': 'Bearer $token',
        },
        body: jsonEncode({
            if (email != null) 'email': email,
            if (firstName != null) 'first_name': firstName,
            if (lastName != null) 'last_name': lastName,
        })),
    );

    if (response.statusCode == 200) {
        return jsonDecode(response.body);
    } else {
        throw Exception('Failed to update profile: ${response.body}');
    }
}
```

```

    }

    // Change password
    Future<void> changePassword({
        required String oldPassword,
        required String newPassword,
    }) async {
        final url = Uri.parse('${ApiConfig.baseUrl}${ApiConfig.authPasswordChange}');
        final token = await getAccessToken();

        final response = await http.post(
            url,
            headers: {
                'Content-Type': 'application/json',
                'Authorization': 'Bearer $token',
            },
            body: jsonEncode({
                'old_password': oldPassword,
                'new_password': newPassword,
                'new_password2': newPassword,
            })),
        );

        if (response.statusCode != 200) {
            throw Exception('Failed to change password: ${response.body}');
        }
    }

    // Refresh access token
    Future<bool> refreshAccessToken() async {
        final refreshToken = await getRefreshToken();
        if (refreshToken == null) return false;

        final url = Uri.parse('${ApiConfig.baseUrl}${ApiConfig.authTokenRefresh}');

        final response = await http.post(
            url,
            headers: {'Content-Type': 'application/json'},
            body: jsonEncode({'refresh': refreshToken}),
        );

        if (response.statusCode == 200) {
            final data = jsonDecode(response.body);
            await saveTokens(data['access'], data['refresh']);
            return true;
        }
        return false;
    }
}

```

Upload Service

DART

```
// lib/services/upload_service.dart
import 'dart:convert';
import 'dart:io';
import 'package:http/http.dart' as http;
import '../config/api_config.dart';
import 'auth_service.dart';

class UploadService {
  final AuthService _authService = AuthService();

  // Upload medicine image
  Future<Map<String, dynamic>> uploadMedicineImage(File imageFile) async {
    final url = Uri.parse('${ApiConfig.baseUrl}${ApiConfig.uploadNew}');
    final token = await _authService.getAccessToken();

    var request = http.MultipartRequest('POST', url);
    request.headers['Authorization'] = 'Bearer $token';

    // Add image
    request.files.add(
      await http.MultipartFile.fromPath('image', imageFile.path),
    );

    final streamedResponse = await request.send();
    final response = await http.Response.fromStream(streamedResponse);

    if (response.statusCode == 201) {
      return jsonDecode(response.body);
    } else {
      throw Exception('Upload failed: ${response.body}');
    }
  }

  // Get user uploads
  Future<List<dynamic>> getUserUploads() async {
    final url = Uri.parse('${ApiConfig.baseUrl}${ApiConfig.uploadsList}');
    final token = await _authService.getAccessToken();

    final response = await http.get(
      url,
      headers: {
        'Authorization': 'Bearer $token',
      },
    );

    if (response.statusCode == 200) {
      return jsonDecode(response.body);
    } else {
      throw Exception('Failed to get uploads: ${response.body}');
    }
  }
}
```

Login Screen Example

DART

```
// lib/screens/login_screen.dart
import 'package:flutter/material.dart';
import '../services/auth_service.dart';

class LoginScreen extends StatefulWidget {
  @override
  _LoginScreenState createState() => _LoginScreenState();
}

class _LoginScreenState extends State<LoginScreen> {
  final _formKey = GlobalKey<FormState>();
  final _usernameController = TextEditingController();
  final _passwordController = TextEditingController();
  final _authService = AuthService();
  bool _isLoading = false;

  Future<void> _login() async {
    if (!_formKey.currentState!.validate()) return;

    setState(() => _isLoading = true);

    try {
      await _authService.login(
        username: _usernameController.text,
        password: _passwordController.text,
      );

      // Navigate to home
      Navigator.pushReplacementNamed(context, '/home');

      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Login successful!')),
      );
    } catch (e) {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Login failed: $e')),
      );
    } finally {
      setState(() => _isLoading = false);
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Login')),
      body: Padding(
        padding: EdgeInsets.all(16.0),
      ),
    );
  }
}
```

```

child: Form(
  key: _formKey,
  child: Column(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      TextFormField(
        controller: _usernameController,
        decoration: InputDecoration(
          labelText: 'Username',
          border: OutlineInputBorder(),
        ),
        validator: (value) {
          if (value == null || value.isEmpty) {
            return 'Please enter username';
          }
          return null;
        },
      ),
      SizedBox(height: 16),
      TextFormField(
        controller: _passwordController,
        decoration: InputDecoration(
          labelText: 'Password',
          border: OutlineInputBorder(),
        ),
        obscureText: true,
        validator: (value) {
          if (value == null || value.isEmpty) {
            return 'Please enter password';
          }
          return null;
        },
      ),
      SizedBox(height: 24),
      SizedBox(
        width: double.infinity,
        child: ElevatedButton(
          onPressed: _isLoading ? null : _login,
          child: _isLoading
            ? CircularProgressIndicator(color: Colors.white)
            : Text('Login'),
        ),
      ),
      TextButton(
        onPressed: () => Navigator.pushNamed(context, '/register'),
        child: Text('No account? Register now'),
      ),
    ],
  ),
),
);
}
}

```

Upload Medicine Image Example

DART

```
// lib/screens/upload_screen.dart
import 'dart:io';
import 'package:flutter/material.dart';
import 'package:image_picker/image_picker.dart';
import '../services/upload_service.dart';

class UploadScreen extends StatefulWidget {
  @override
  _UploadScreenState createState() => _UploadScreenState();
}

class _UploadScreenState extends State<UploadScreen> {
  final _uploadService = UploadService();
  final _picker = ImagePicker();
  File? _imageFile;
  bool _isUploading = false;
  Map<String, dynamic>? _result;

  Future<void> _pickImage(ImageSource source) async {
    final pickedFile = await _picker.pickImage(source: source);
    if (pickedFile != null) {
      setState(() {
        _imageFile = File(pickedFile.path);
        _result = null;
      });
    }
  }

  Future<void> _uploadImage() async {
    if (_imageFile == null) return;

    setState(() => _isUploading = true);

    try {
      final result = await _uploadService.uploadMedicineImage(_imageFile!);
      setState(() => _result = result);

      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Upload successful!')),
      );
    } catch (e) {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Upload failed: $e')),
      );
    } finally {
      setState(() => _isUploading = false);
    }
  }
}
```

```

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(title: Text('Upload Medicine Image')),
    body: SingleChildScrollView(
      padding: EdgeInsets.all(16),
      child: Column(
        children: [
          if (_imageFile != null)
            Container(
              height: 300,
              decoration: BoxDecoration(
                border: Border.all(color: Colors.grey),
                borderRadius: BorderRadius.circular(8),
              ),
              child: Image.file(_imageFile!, fit: BoxFit.cover),
            ),
          SizedBox(height: 16),
          Row(
            mainAxisAlignment: MainAxisAlignment.spaceEvenly,
            children: [
              ElevatedButton.icon(
                onPressed: () => _pickImage(ImageSource.camera),
                icon: Icon(Icons.camera_alt),
                label: Text('Camera'),
              ),
              ElevatedButton.icon(
                onPressed: () => _pickImage(ImageSource.gallery),
                icon: Icon(Icons.photo_library),
                label: Text('Gallery'),
              ),
            ],
          ),
          SizedBox(height: 16),
          if (_imageFile != null)
            SizedBox(
              width: double.infinity,
              child: ElevatedButton(
                onPressed: _isUploading ? null : _uploadImage,
                child: _isUploading
                  ? CircularProgressIndicator(color: Colors.white)
                  : Text('Analyze Image'),
              ),
            ),
          if (_result != null) ...[
            SizedBox(height: 24),
            Card(
              child: Padding(
                padding: EdgeInsets.all(16),
                child: Column(
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: [
                    Text(
                      'Analysis Result:',
                      style: TextStyle(
                        fontSize: 18,
                        fontWeight: FontWeight.bold,

```

```
        ),  
    ),  
    SizedBox(height: 8),  
    Text(_result!['result'] ?? 'No result'),  
  ],  
),  
),  
),  
),  
  1,  
  1,  
),  
),  
);  
}  
}
```

Important Tips

1. Run Server for Mobile Connection

```
POWERSHELL
python manage.py runserver 0.0.0.0:8000
```

2. Find Your PC IP

POWERSHELL
ipconfig

3. Update Django ALLOWED_HOSTS

```
PYTHON
# In medrec/settings.py:
```

4. For Android Emulator

Use `http://10.0.2.2:8000` instead of `localhost`

5. For iOS Simulator

Use `http://127.0.0.1:8000`

6. For Real Device

Use your PC IP like `http://192.168.1.5:8000`

Error Handling

DART

```
// lib/utils/api_error_handler.dart
class ApiErrorHandler {
  static String getErrorMessage(dynamic error) {
    if (error.toString().contains('SocketException')) {
      return 'No internet connection';
    } else if (error.toString().contains('TimeoutException')) {
      return 'Connection timeout';
    } else if (error.toString().contains('401')) {
      return 'Please login again';
    } else if (error.toString().contains('404')) {
      return 'Not found';
    } else if (error.toString().contains('500')) {
      return 'Server error';
    }
    return 'Unexpected error';
  }
}
```

Pre-Start Checklist

- Run server: `python manage.py runserver 0.0.0.0:8000`
- Change `baseUrl` in `api_config.dart` for your device
- Add dependencies in `pubspec.yaml`
- Add permissions in `AndroidManifest.xml`:

```
```xml
```

---

...

- Add permissions in Info.plist for iOS:

```xml

NSCameraUsageDescription

Camera needed to capture medicine

NSPhotoLibraryUsageDescription

Gallery access to select medicine image

...

Ready to start! ■