

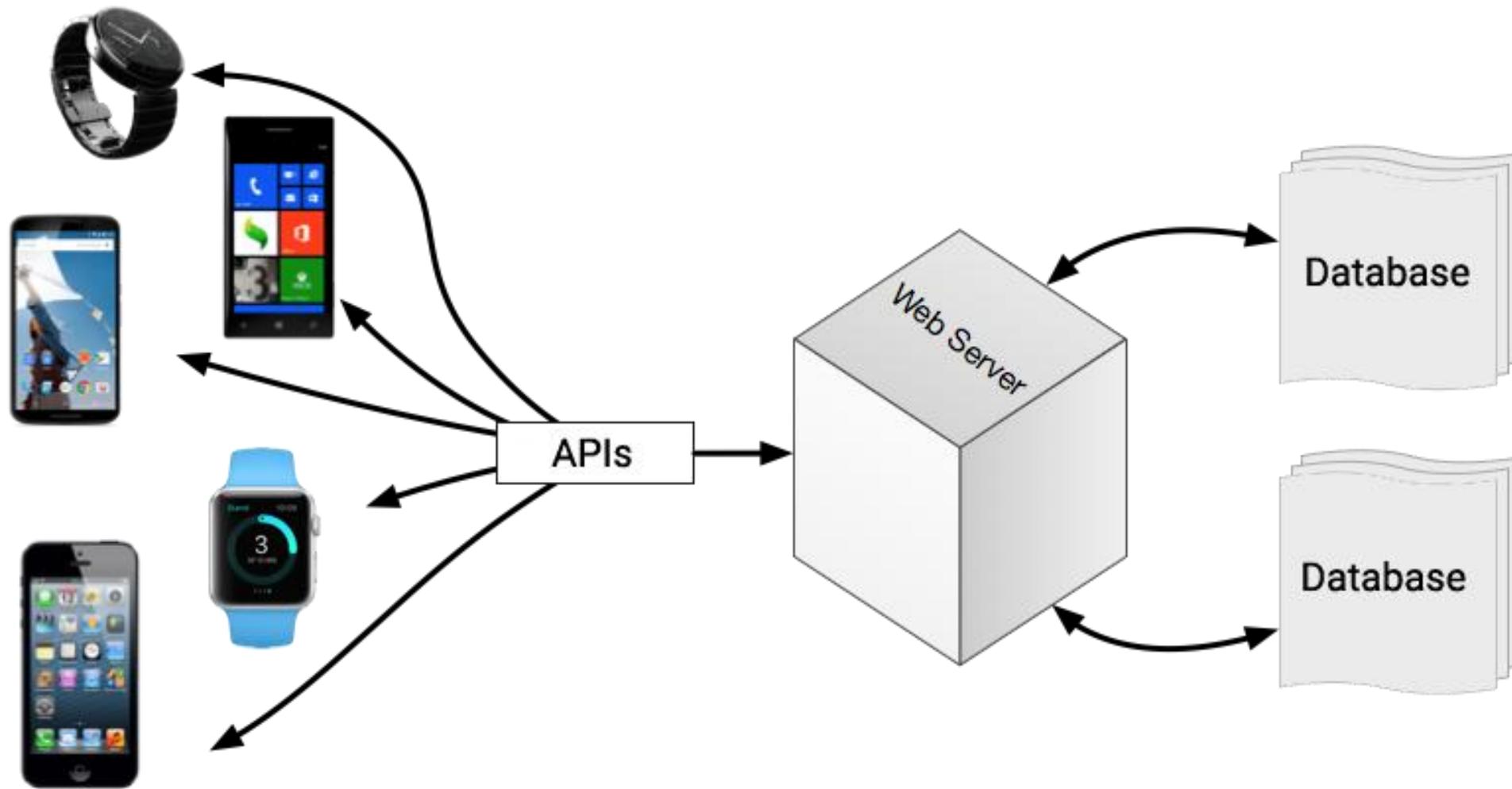
API  
(Application  
Programming Interface)

# What is an API?

API is the acronym for **Application Programming Interface**, which is a software intermediary that allows two applications to talk to each other.

Each time you use an app like Facebook, send an instant message, or check the weather on your phone, you're using an API.

# API Architecture



# Data Interchange format : JSON vs XML

JSON	XML
JavaScript Object Notation	Extensible Markup Language
Text based format	Markup Language
Light-weighted	Heavier
Does not support comments and namespaces	Supports comments and namespaces

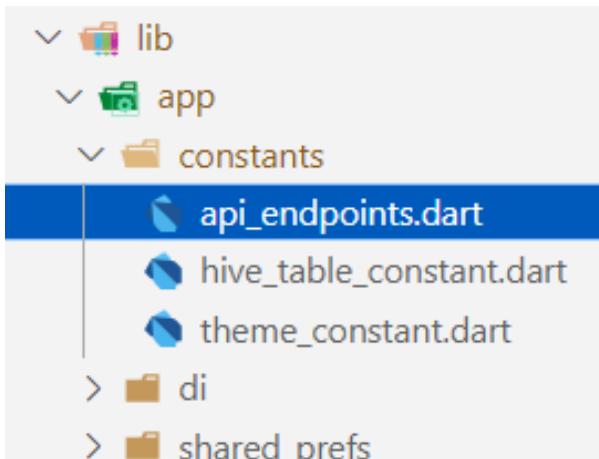
```
{  
    id: "tt1231",  
    title: "Pride and Prejudice"  
    year: 2093,  
    director: "Michael Bay",  
    genres: [  
        "Horror",  
        "Comedy"  
    ],  
    stars: [  
        {  
            name: "Kiera Knightley",  
            id: 9863  
        },  
        {  
            name: "Danny DeVito",  
            id: 2031  
        }  
    ]  
}
```

1/29/2025

```
<?xml version="1.0" encoding="UTF-8"?>  
<events>  
    <event>  
        <week>-mtwtf-</week>  
        <starttime>06:00</starttime>  
        <endtime>11:59</endtime>  
    </event>  
    <event>  
        <week>SMTWTF-</week>  
        <starttime>12:00</starttime>  
        <endtime>20:59</endtime>  
    </event>  
</events>
```

© Kiran Rana

# Step 1 : Constant



```
1 class ApiEndpoints {
2     ApiEndpoints._();
3
4     static const Duration connectionTimeout = Duration(seconds: 1000);
5     static const Duration receiveTimeout = Duration(seconds: 1000);
6     static const String baseUrl = "http://10.0.2.2:3000/api/v1/";
7     // For iPhone
8     //static const String baseUrl = "http://localhost:3000/api/v1/";
9
10    // ====== Auth Routes ======
11    static const String login = "auth/login";
12    static const String register = "auth/register";
13    static const String getAllStudent = "auth/getAllStudents";
14    static const String getStudentsByBatch = "auth/getStudentsByBatch/";
15    static const String getStudentsByCourse = "auth/getStudentsByCourse/";
16    static const String updateStudent = "auth/updateStudent/";
17    static const String deleteStudent = "auth/deleteStudent/";
18    static const String imageUrl = "http://10.0.2.2:3000/uploads/";
19    static const String uploadImage = "auth/uploadImage";
20
21    // ====== Batch Routes ======
22    static const String createBatch = "batch/createBatch";
23    static const String getAllBatch = "batch/getAllBatches";
24
25    // ====== Course Routes ======
26    static const String createCourse = "course/createCourse";
27    static const String deleteCourse = "course/";
28    static const String getAllCourse = "course/getAllCourse";
29 }
```



# dio 5.4.3+1

Published 58 days ago · [flutter.cn](#) Dart 3 compatible

SDK DART FLUTTER

PLATFORM ANDROID IOS LINUX MACOS WEB WINDOWS

1 7.0K

[Readme](#) [Changelog](#) [Example](#) [Installing](#) [Versions](#) [Scores](#)

## Use this package as a library

Depend on it

Run this command:

With Dart:

```
$ dart pub add dio
```

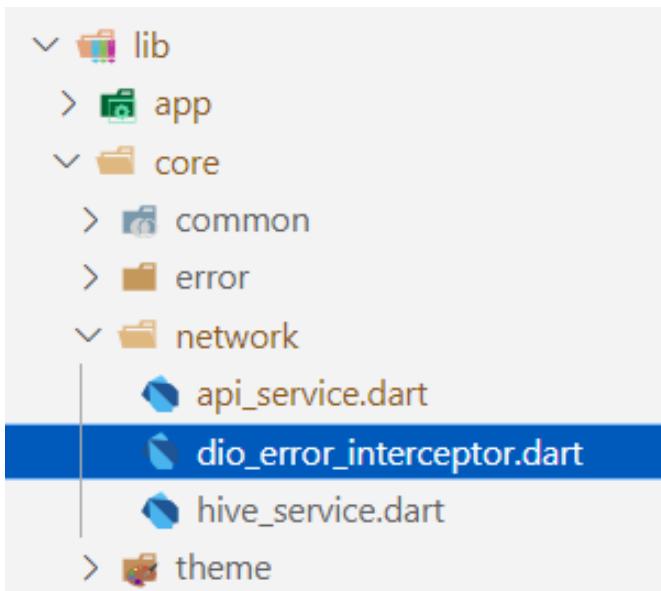
With Flutter:

```
$ flutter pub add dio
```

This will add a line like this to your package's pubspec.yaml (and run an implicit `dart pub get`):

```
dependencies:  
  dio: ^5.4.3+1
```

# Step 2 : Dio



```
1 import 'package:dio/dio.dart';
2
3 class DioErrorInterceptor extends Interceptor {
4   @override
5   void onError(DioException err, ErrorInterceptorHandler handler) {
6     if (err.response != null) {
7       // Handle server errors
8       if (err.response!.statusCode! >= 300) {
9         err = DioException(
10           requestOptions: err.requestOptions,
11           response: err.response,
12           error: err.response!.data['message'] ?? err.response!.statusMessage!,
13           type: err.type,
14         );
15       } else {
16         err = DioException(
17           requestOptions: err.requestOptions,
18           response: err.response,
19           error: 'Something went wrong',
20           type: err.type,
21         );
22       }
23     } else {
24       // Handle connection errors
25     err = DioException(
26       requestOptions: err.requestOptions,
27       error: 'Connection error',
28       type: err.type,
29     );
30   }
31   super.onError(err, handler);
32 }
33 }
```

## Step 2 : Dio

```
6  class ApiService {
7    final Dio _dio;
8
9    Dio get dio => _dio;
10
11   ApiService(this._dio) {
12     _dio
13       ..options.baseUrl = ApiEndpoints.baseUrl
14       ..options.connectTimeout = ApiEndpoints.connectionTimeout
15       ..options.receiveTimeout = ApiEndpoints.receiveTimeout
16       ..interceptors.add(DioErrorInterceptor())
17       ..interceptors.add(PrettyDioLogger(
18         requestHeader: true, requestBody: true, responseHeader: true))
19     ..options.headers = {
20       'Accept': 'application/json',
21       'Content-Type': 'application/json',
22     };
23   }
24 }
```

# Batch Data Layer

---

# Install JsonAnnotation

## json\_annotation 4.9.0

Published 48 days ago • ⚙️ google.dev Dart 3 compatible

SDK DART FLUTTER PLATFORM ANDROID IOS LINUX MACOS WEB WINDOWS

1.0

[Readme](#) [Changelog](#) [Installing](#) [Versions](#) [Scores](#)

### Use this package as a library

Depend on it

Run this command:

With Dart:

```
$ dart pub add json_annotation
```

With Flutter:

```
$ flutter pub add json_annotation
```

This will add a line like this to your package's pubspec.yaml (and run an implicit `dart pub get`):

```
dependencies:  
  json_annotation: ^4.9.0
```

# Step 3: Batch Model

batch
data
data_source
dto
model
batch_api_model.dart
batch_hive_model.dart
batch_hive_model.g.dart
repository

```
{  
  "success": true,  
  "count": 5,  
  "data": [  
    {  
      "_id": "6489a560571a9025bb2f5199",  
      "batchName": "30-B",  
      "__v": 0  
    },  
    {  
      "_id": "6489a566571a9025bb2f519b",  
      "batchName": "30-A",  
      "__v": 0  
    },  
    {  
      "_id": "6489a56b571a9025bb2f519d",  
      "batchName": "31-A",  
      "__v": 0  
    },  
    {  
      "_id": "6489a56e571a9025bb2f519f",  
      "batchName": "31-B",  
      "__v": 0  
    },  
  ]  
}
```

```
5  @JsonSerializable()  
6  class BatchApiModel extends Equatable {  
7    @JsonKey(name: '_id')  
8    final String? batchId;  
9    final String batchName;  
10  
11   const BatchApiModel({  
12     this.batchId,  
13     required this.batchName,  
14   });  
15  
16   const BatchApiModel.empty()  
17   |   : batchId = '',  
18   |   batchName = '';
```

# Step 3 : Add toJson and fromJson

```
24 // From Json , write full code without generator
25 factory BatchApiModel.fromJson(Map<String, dynamic> json) {
26     return BatchApiModel(
27         batchId: json['_id'],
28         batchName: json['batchName'],
29     );
30 }
31
32 // To Json , write full code without generator
33 Map<String, dynamic> toJson() {
34     return {
35         '_id': batchId,
36         'batchName': batchName,
37     };
38 }
```



```
{  
    "success": true,  
    "count": 5,  
    "data": [  
        {  
            "_id": "6489a560571a9025bb2f5199",  
            "batchName": "30-B",  
            "__v": 0  
        },  
        {  
            "_id": "6489a566571a9025bb2f519b",  
            "batchName": "30-A",  
            "__v": 0  
        },  
        {  
            "_id": "6489a56b571a9025bb2f519d",  
            "batchName": "31-A",  
            "__v": 0  
        },  
        {  
            "_id": "6489a56e571a9025bb2f519f",  
            "batchName": "31-B",  
            "__v": 0  
        }  
    ]  
}
```

# Step 3 : Add conversion code

```
40 // Convert API Object to Entity
41 BatchEntity toEntity() => BatchEntity(
42     batchId: batchId,
43     batchName: batchName,
44 );
45
46 // Convert Entity to API Object
47 BatchApiModel fromEntity(BatchEntity entity) => BatchApiModel(
48     batchId: entity.batchId ?? '',
49     batchName: entity.batchName,
50 );
51
52 // Convert API List to Entity List
53 List<BatchEntity> toEntityList(List<BatchApiModel> models) =>
54     models.map((model) => model.toEntity()).toList();
55
56 @override
57 List<Object?> get props => [batchId, batchName];
58 }
```

# Postman

POST ▼ localhost:3000/api/v1/batch/createBatch Send ▼

Params Auth Headers (9) **Body** ● Pre-req. Tests Settings Cookies

raw ▼ JSON ▼ Beautify

```
1 {  
2   "batchName": "32-B"  
3 }
```

Body ▼ 🌐 201 Created 31 ms 1.02 KB 💾 Save as Example ...

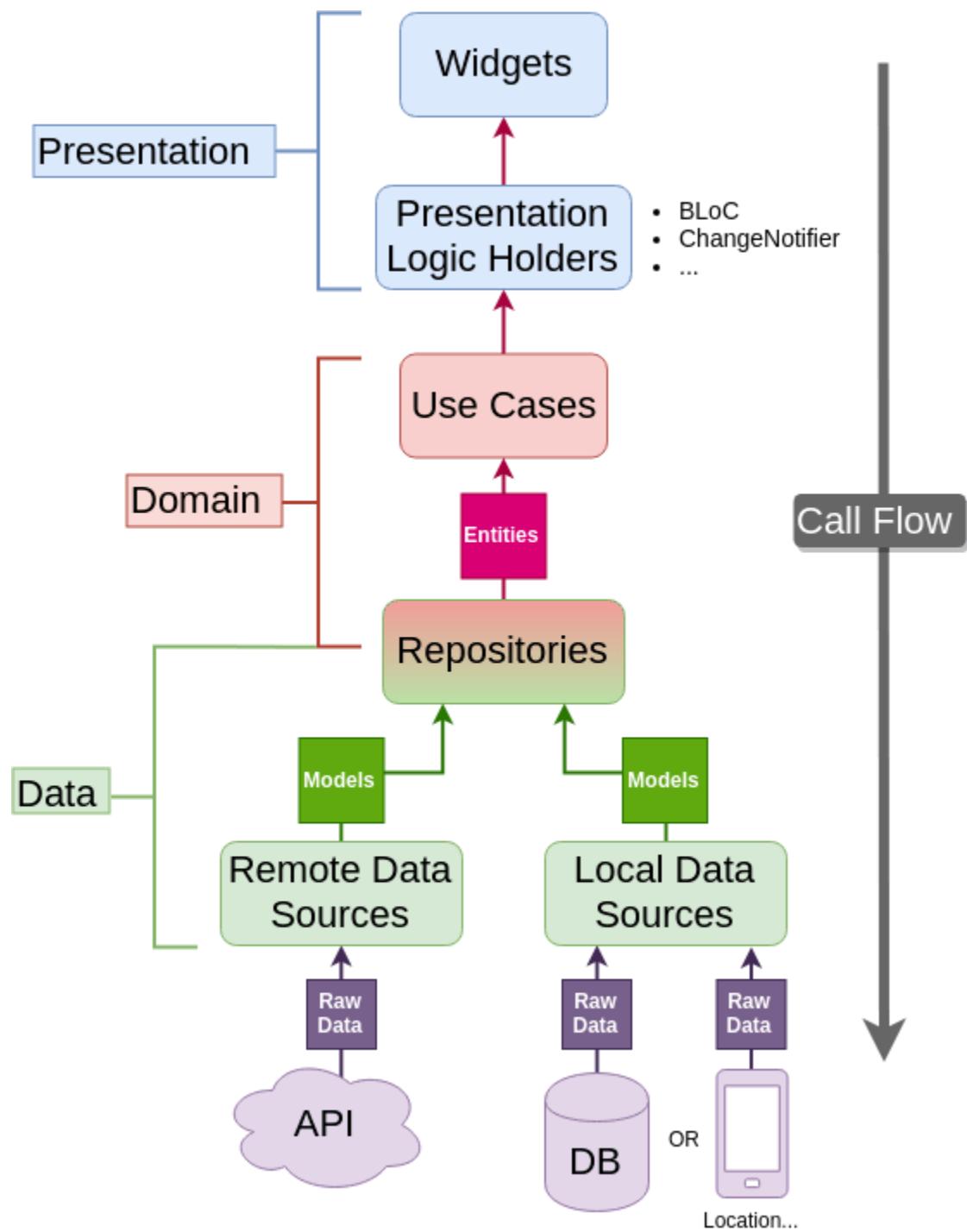
Pretty Raw Preview Visualize JSON 🔗 ◻ 🔍

```
1 {  
2   "success": true,  
3   "data": {  
4     "batchName": "32-B",  
5     "_id": "648ae46e0377283545881fdf",  
6     "__v": 0  
7   }  
8 }
```

# Remote Data Source

```
35 @override  
36 Future<void> deleteBatch(String id) {  
37     // TODO: implement deleteBatch  
38     throw UnimplementedError();  
39 }  
40  
41 @override  
42 Future<List<BatchEntity>> getBatches() {  
43     // TODO: implement deleteBatch  
44     throw UnimplementedError();  
45 }  
46 }
```

```
7 class BatchRemoteDataSource implements IDataSource {  
8     final Dio _dio;  
9  
10    BatchRemoteDataSource({  
11        required Dio dio,  
12    }) : _dio = dio;  
13  
14    @override  
15    Future<void> createBatch(BatchEntity batch) async {  
16        try {  
17            // Convert entity to model  
18            var batchApiModel = BatchApiModel.fromEntity(batch);  
19            var response = await _dio.post(  
20                ApiEndpoints.createBatch,  
21                data: batchApiModel.toJson(),  
22            );  
23            if (response.statusCode == 201) {  
24                return;  
25            } else {  
26                throw Exception(response.statusText);  
27            }  
28        } on DioException catch (e) {  
29            throw Exception(e);  
30        } catch (e) {  
31            throw Exception(e);  
32        }  
33    }
```

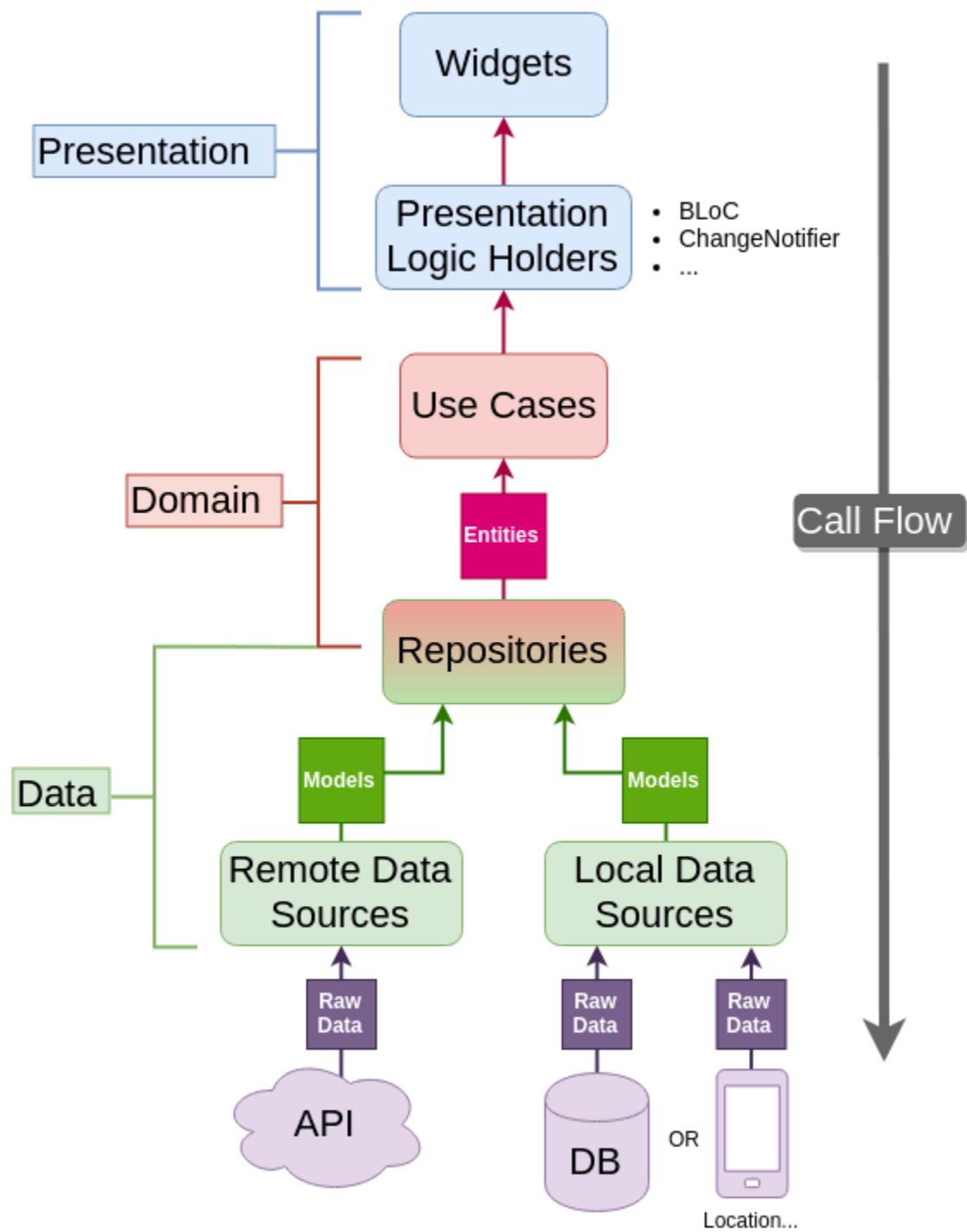


# Remote Repository

```
7 class BatchRemoteRepository implements IBatchRepository {  
8     final BatchRemoteDataSource remoteDataSource;  
9  
10    BatchRemoteRepository({required this.remoteDataSource});  
11  
12    @override  
13    Future<Either<Failure, void>> createBatch(BatchEntity batch) async {  
14        try {  
15            remoteDataSource.createBatch(batch);  
16            return Right(null);  
17        } catch (e) {  
18            return Left(  
19                ApiFailure(  
20                    message: e.toString(),  
21                    ), // ApiFailure  
22                ); // Left  
23        }  
24    }  
}
```

# Remote Repository

```
26     @override
27     Future<Either<Failure, void>> deleteBatch(String id) {
28         // TODO: implement deleteBatch
29         throw UnimplementedError();
30     }
31
32     @override
33     Future<Either<Failure, List<BatchEntity>>> getBatches() {
34         // TODO: implement getBatches
35         throw UnimplementedError();
36     }
37 }
```



# Get\_It : Api Service

```
30 Future<void> initDependencies() async {
31     // First initialize hive service
32     await _initHiveService();
33     await _init ApiService();
34
35     await _initBatchDependencies();
36     await _initCourseDependencies();
37     await _initHomeDependencies();
38     await _initRegisterDependencies();
39     await _initLoginDependencies();
40
41     await _initSplashscreenDependencies();
42 }
43
44 _init ApiService() {
45     // Remote Data Source
46     getIt.registerLazySingleton< Dio >(
47         () => ApiService(Dio()).dio,
48     );
49 }
```

# Get\_It

```
122 _initBatchDependencies() async {
123     // Local Data Source
124     getIt.registerFactory<BatchLocalDataSource>(
125         () => BatchLocalDataSource(hiveService: getIt<HiveService>()));
126
127     // Remote Data Source
128     getIt.registerLazySingleton<BatchRemoteDataSource>(
129         () => BatchRemoteDataSource(
130             dio: getIt<dio>(),
131         ),
132     );
133
134     // Batch Local Repository
135     getIt.registerLazySingleton<BatchLocalRepository>(() => BatchLocalRepository(
136         batchLocalDataSource: getIt<BatchLocalDataSource>()));
137
138     // Batch Remote Repository
139     getIt.registerLazySingleton(
140         () => BatchRemoteRepository(
141             remoteDataSource: getIt<BatchRemoteDataSource>(),
142         ),
143     );
144
145     // Usecases
146     getIt.registerLazySingleton<CreateBatchUseCase>(
147         () => CreateBatchUseCase(batchRepository: getIt<BatchRemoteRepository>()),
148     );
```

```
150     getIt.registerLazySingleton<GetAllBatchUseCase>(
151         () => GetAllBatchUseCase(batchRepository: getIt<BatchRemoteRepository>()),
152     );
153
154     getIt.registerLazySingleton<DeleteBatchUsecase>(
155         () => DeleteBatchUsecase(batchRepository: getIt<BatchRemoteRepository>()),
156     );
157
158     // Bloc
159     getIt.registerFactory<BatchBloc>(
160         () => BatchBloc(
161             createBatchUseCase: getIt<CreateBatchUseCase>(),
162             getAllBatchUseCase: getIt<GetAllBatchUseCase>(),
163             deleteBatchUsecase: getIt<DeleteBatchUsecase>(),
164         ),
165     );
166 }
```

# Run the project and check mongodb for the data

The screenshot shows the MongoDB Compass interface. On the left, there is a tree view of databases and collections:

- localhost:27017
  - admin
  - config
  - local
  - studentPe
  - student\_batch
    - batches
    - courses
    - students

On the right, there is a list of documents from the 'batches' collection:

- `_id: ObjectId('67887fcb1a38ef59888d75ae')`  
`batchName : "33-A"`  
`--v : 0`
- `_id: ObjectId('678880551a38ef59888d75b0')`  
`batchName : "33-B"`  
`--v : 0`

The screenshot shows the MongoDB Compass interface. On the left, there is a list of documents from the 'batches' collection:

- `_id: ObjectId('67887fcb1a38ef59888d75ae')`  
`batchName : "33-A"`  
`--v : 0`
- `_id: ObjectId('678880551a38ef59888d75b0')`  
`batchName : "33-B"`  
`--v : 0`

On the right, there is a detailed view of one of the documents:

- `_id: ObjectId('67887fcb1a38ef59888d75ae')`  
`batchName : "33-A"`  
`--v : 0`

At the top of this view, there are four buttons: a green plus sign (+) button, a dropdown arrow button, a pencil edit button, and a trash can delete button.

Now try to insert course by yourself

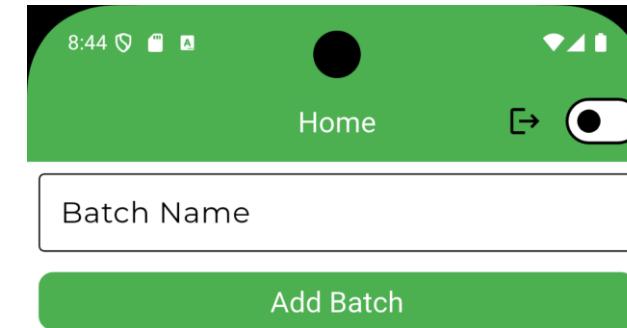
# Get all batches from API

Type a query: { field: 'value' } or [Generate query](#) ⚡

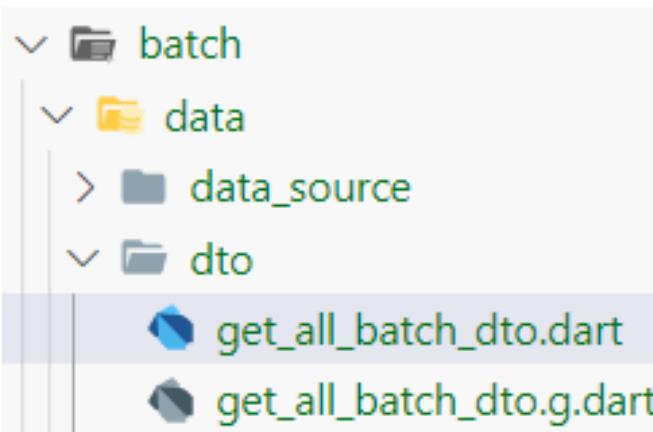
[ADD DATA](#) [EXPORT DATA](#) [UPDATE](#) [DELETE](#)

```
_id: ObjectId('67887fcb1a38ef59888d75ae')
batchName : "33-A"
__v : 0
```

```
_id: ObjectId('678880551a38ef59888d75b0')
batchName : "33-B"
__v : 0
```



# Step 1: DTO (Data to Object)



```
4 part 'get_all_batch_dto.g.dart';
5
6 @JsonSerializable()
7 class GetAllBatchDTO {
8   final bool success;
9   final int count;
10  final List<BatchApiModel> data;
11
12  GetAllBatchDTO({
13    required this.success,
14    required this.count,
15    required this.data,
16  });
17
18  Map<String, dynamic> toJson() => _$GetAllBatchDTOToJson(this);
19
20  factory GetAllBatchDTO.fromJson(Map<String, dynamic> json) =>
21    _$GetAllBatchDTOFromJson(json);
22 }
```

```
{
  "success": true,
  "count": 6,
  "data": [
    {
      "_id": "6489a560571a9025bb2f5199",
      "batchName": "30-B",
      "__v": 0
    },
    {
      "_id": "6489a566571a9025bb2f519b",
      "batchName": "30-A",
      "__v": 0
    }
  ]
}
```

```
dart run build_runner build -d
```

# Step 2 : Remote Data Source

```
42    @override
43    Future<List<BatchEntity>> getBatches() async {
44        try {
45            var response = await _dio.get(ApiEndpoints.getAllBatch);
46            if (response.statusCode == 200) {
47                GetAllBatchDTO batchAddDTO = GetAllBatchDTO.fromJson(response.data);
48                return BatchApiModel.toEntityList(batchAddDTO.data);
49            } else {
50                throw Exception(response.statusMessage);
51            }
52        } on DioException catch (e) {
53            throw Exception(e);
54        } catch (e) {
55            throw Exception(e);
56        }
57    }
```

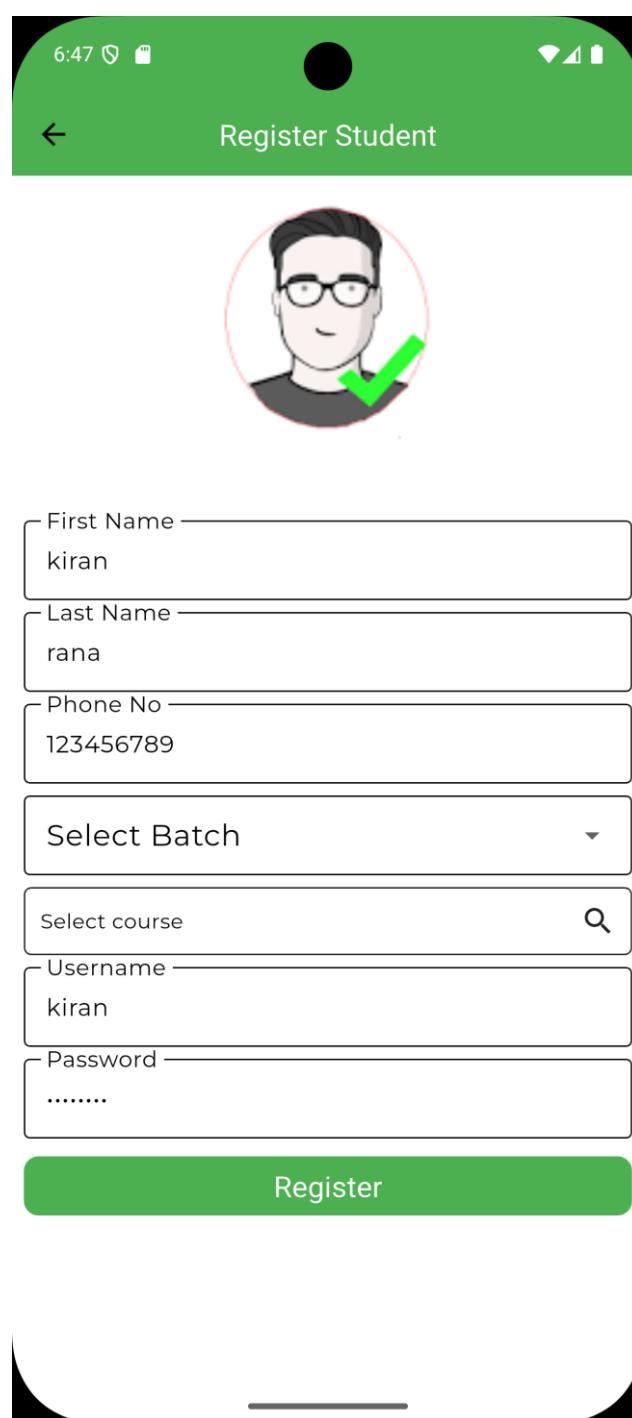
# Step 3 : Batch Remote Repository

```
32    @override
33    Future<Either<Failure, List<BatchEntity>>> getBatches() async {
34        try {
35            final batches = await remoteDataSource.getBatches();
36            return Right(batches);
37        } catch (e) {
38            return Left(
39                ApiFailure(
40                    message: e.toString(),
41                    ), // ApiFailure
42                ); // Left
43        }
44    }
```

# Step 4: GetIt

```
145 // Usecases
146 getIt.registerLazySingleton<CreateBatchUseCase>(
147   () => CreateBatchUseCase(batchRepository: getIt<BatchRemoteRepository>()),
148 );
149
150 getIt.registerLazySingleton<GetAllBatchUseCase>(
151   () => GetAllBatchUseCase(batchRepository: getIt<BatchRemoteRepository>()),
152 );
153
154 getIt.registerLazySingleton<DeleteBatchUsecase>(
155   () => DeleteBatchUsecase(batchRepository: getIt<BatchRemoteRepository>()),
156 );
```

# Register User



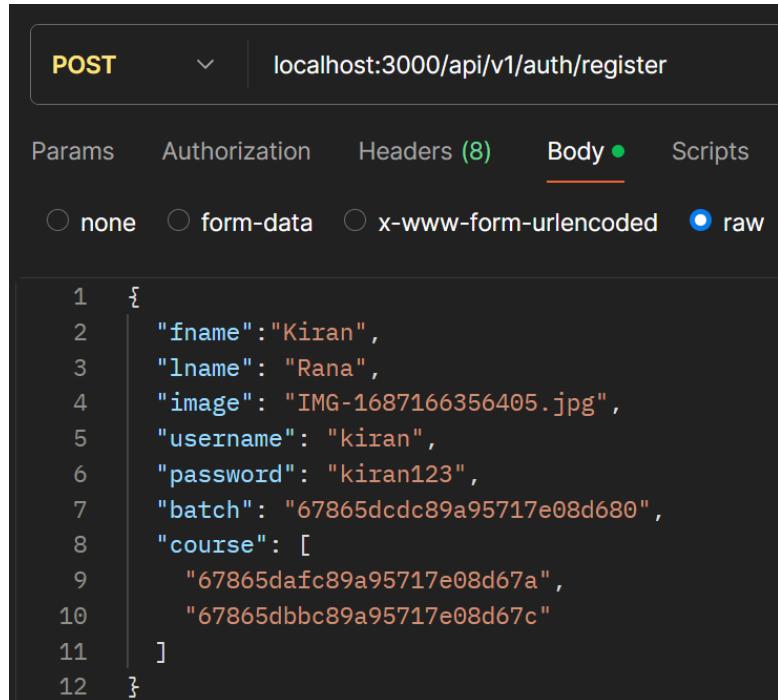
# 1. Auth API Model

```
7 part 'auth_api_model.g.dart';
8
9 @JsonSerializable()
10 class AuthApiModel extends Equatable {
11   @JsonKey(name: '_id')
12   final String? id;
13   final String fname;
14   final String lname;
15   final String? image;
16   final String phone;
17   final BatchApiModel batch;
18   final List<CourseApiModel> courses;
19   final String username;
20   final String? password;
21
22   const AuthApiModel(
23     this.id,
24     required this.fname,
25     required this.lname,
26     required this.image,
27     required this.phone,
28     required this.batch,
29     required this.courses,
30     required this.username,
31     required this.password,
32   );
```

# 1. Auth API Mode

```
34     factory AuthApiModel.fromJson(Map<String, dynamic> json) =>
35         _$AuthApiModelFromJson(json);
36
37     Map<String, dynamic> toJson() => _$AuthApiModelToJson(this);
38
39     // To Entity
40     AuthEntity toEntity() {
41         return AuthEntity(
42             userId: id,
43             fName: fname,
44             lName: lname,
45             image: image,
46             phone: phone,
47             batch: batch.toEntity(),
48             courses: courses.map((e) => e.toEntity()).toList(),
49             username: username,
50             password: password ?? '',
51         );
52     }
53
54     // From Entity
55     factory AuthApiModel.fromEntity(AuthEntity entity) {
56         return AuthApiModel(
57             fname: entity.fname,
58             lname: entity.lname,
59             image: entity.image,
60             phone: entity.phone,
61             batch: BatchApiModel.fromEntity(entity.batch),
62             courses: entity.courses.map((e) => CourseApiModel.fromEntity(e)).toList(),
63             username: entity.username,
64             password: entity.password,
65         ); // AuthApiModel
66     }
```

# 2. Batch Remote Data Source



```
13 @override
14 Future<void> registerStudent(AuthEntity student) async {
15     try {
16         Response response = await _dio.post(
17             ApiEndpoints.register,
18             data: {
19                 "fname": student.fName,
20                 "lname": student.lName,
21                 "phone": student.phone,
22                 "image": student.image,
23                 "username": student.username,
24                 "password": student.password,
25                 "batch": student.batch.batchId,
26                 // "course": ["6489a5908dbc6d39719ec19c", "6489a5968dbc6d39719ec19e"]
27                 "course": student.courses.map((e) => e.courseId).toList(),
28             },
29         );
30     }
31     if (response.statusCode == 200) {
32         return;
33     } else {
34         throw Exception(response.statusMessage);
35     }
36 } on DioException catch (e) {
37     throw Exception(e);
38 } catch (e) {
39     throw Exception(e);
40 }
41 }
```

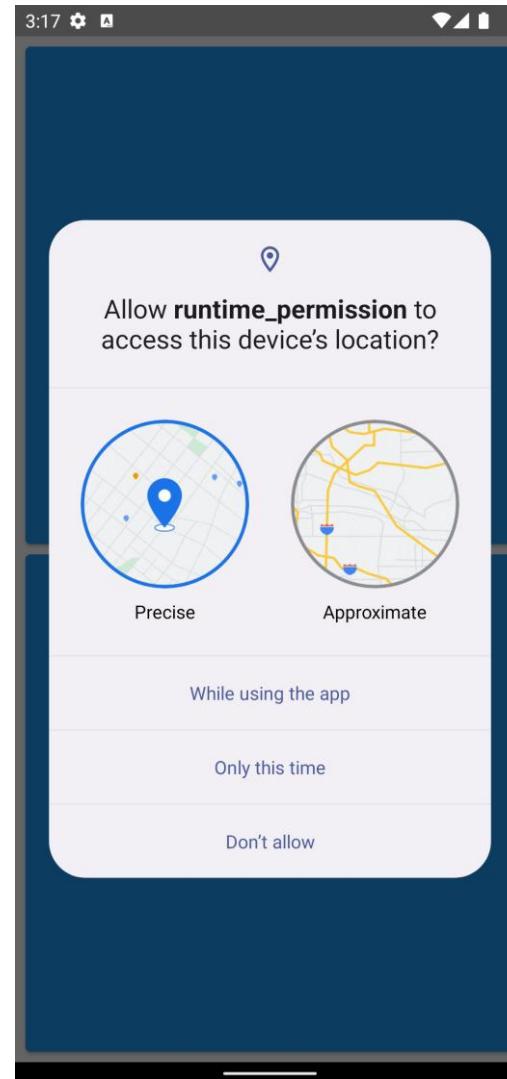
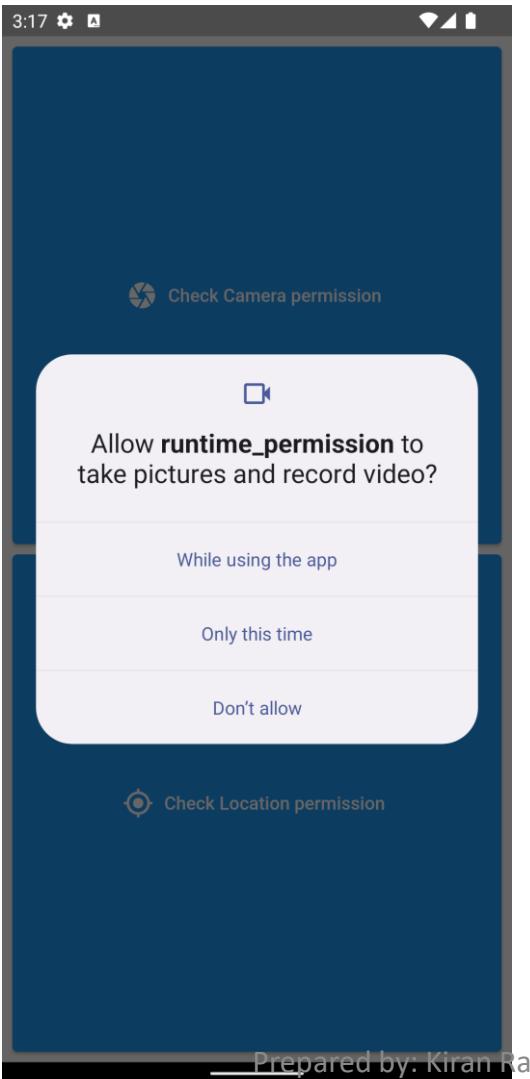
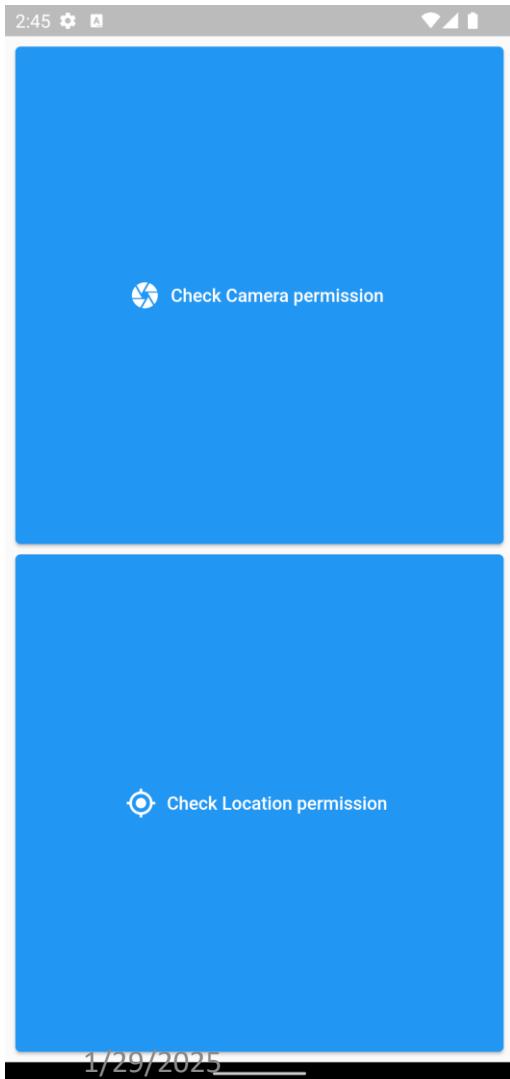
# 3. Batch Remote Repository

```
9  class AuthRemoteRepository implements IAuthRepository {
10    final AuthRemoteDataSource _authRemoteDataSource;
11
12    AuthRemoteRepository(this._authRemoteDataSource);
13
14    @override
15    Future<Either<Failure, void>> registerStudent(AuthEntity student) async {
16      try {
17        await _authRemoteDataSource.registerStudent(student);
18        return Right(null);
19      } catch (e) {
20        return Left(ApiFailure(message: e.toString()));
21      }
22    }
23
24    @override
25    Future<Either<Failure, AuthEntity>> getCurrentUser() {
26      // TODO: implement getCurrentUser
27      throw UnimplementedError();
28    }
29
30    @override
31    Future<Either<Failure, String>> loginStudent(
32      String username, String password) {
33      // TODO: implement loginStudent
34      throw UnimplementedError();
35    }
36
37    @override
38    Future<Either<Failure, String>> uploadProfilePicture(File file) {
39      // TODO: implement uploadProfilePicture
40      throw UnimplementedError();
41    }
42 }
```

## 4. GetIt

```
59 _initRegisterDependencies() {
60 // ===== Data Source =====
61
62 getIt.registerLazySingleton<AuthLocalDataSource>(
63   () => AuthLocalDataSource(getIt< HiveService>()),
64 );
65
66 getIt.registerLazySingleton<AuthRemoteDataSource>(
67   () => AuthRemoteDataSource(getIt< Dio>()),
68 );
69
70 // ===== Repository =====
71
72 getIt.registerLazySingleton(
73   () => AuthLocalRepository(getIt<AuthLocalDataSource>()),
74 );
75 getIt.registerLazySingleton<AuthRemoteRepository>(
76   () => AuthRemoteRepository(getIt<AuthRemoteDataSource>()),
77 );
78
79 // register use usecase
80 getIt.registerLazySingleton<RegisterUseCase>(
81   () => RegisterUseCase(
82     getIt<AuthRemoteRepository>(),
83   ),
84 );
85
86 getIt.registerFactory<RegisterBloc>(
87   () => RegisterBloc(
88     batchBloc: getIt<BatchBloc>(),
89     courseBloc: getIt<CourseBloc>(),
90     registerUseCase: getIt(),
91   ),
92 );
93 }
```

# Runtime permission



# Permissions on Android

- App permissions help support user privacy by protecting access to the following:
  - **Restricted data**, such as system state and a user's contact information.
  - **Restricted actions**, such as connecting to a paired device and recording audio.

# Step 1 - Add permission\_handler dependency for runtime permission

## permission\_handler 10.2.0

Published 2 months ago •  [baseflow.com](#) Null safety

SDK

FLUTTER

PLATFORM

ANDROID

IOS

WINDOWS

1K 3.3K

Readme

Changelog

Example

Installing

Versions

Scores

### Use this package as a library

Depend on it

Run this command:

With Flutter:

```
$ flutter pub add permission_handler
```

This will add a line like this to your package's pubspec.yaml (and run an implicit flutter pub get):

```
dependencies:  
  permission_handler: ^10.2.0
```

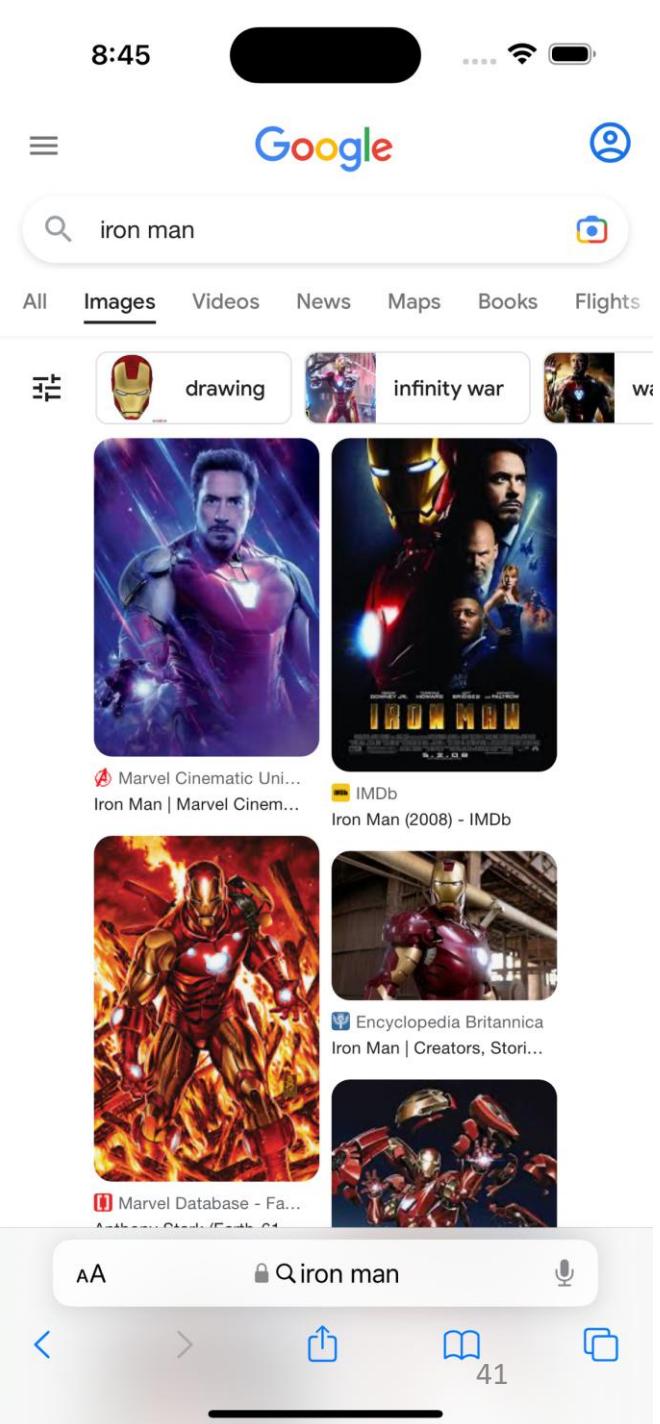
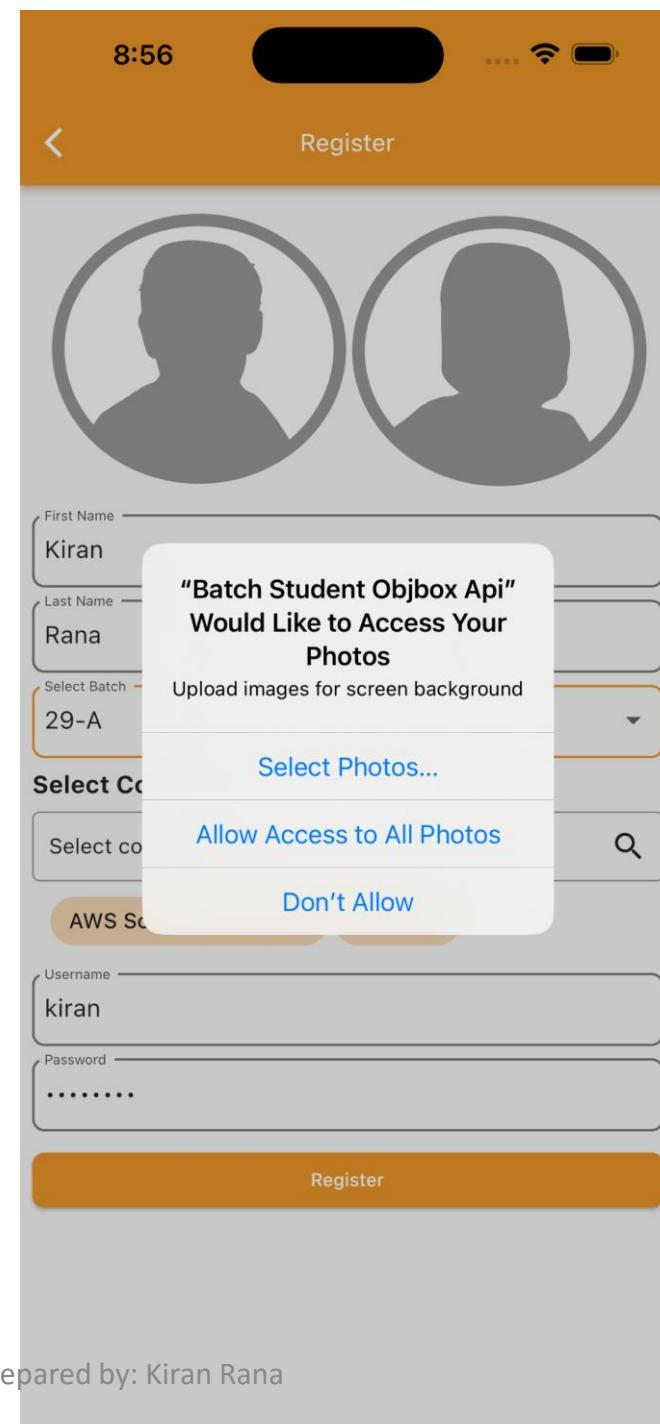
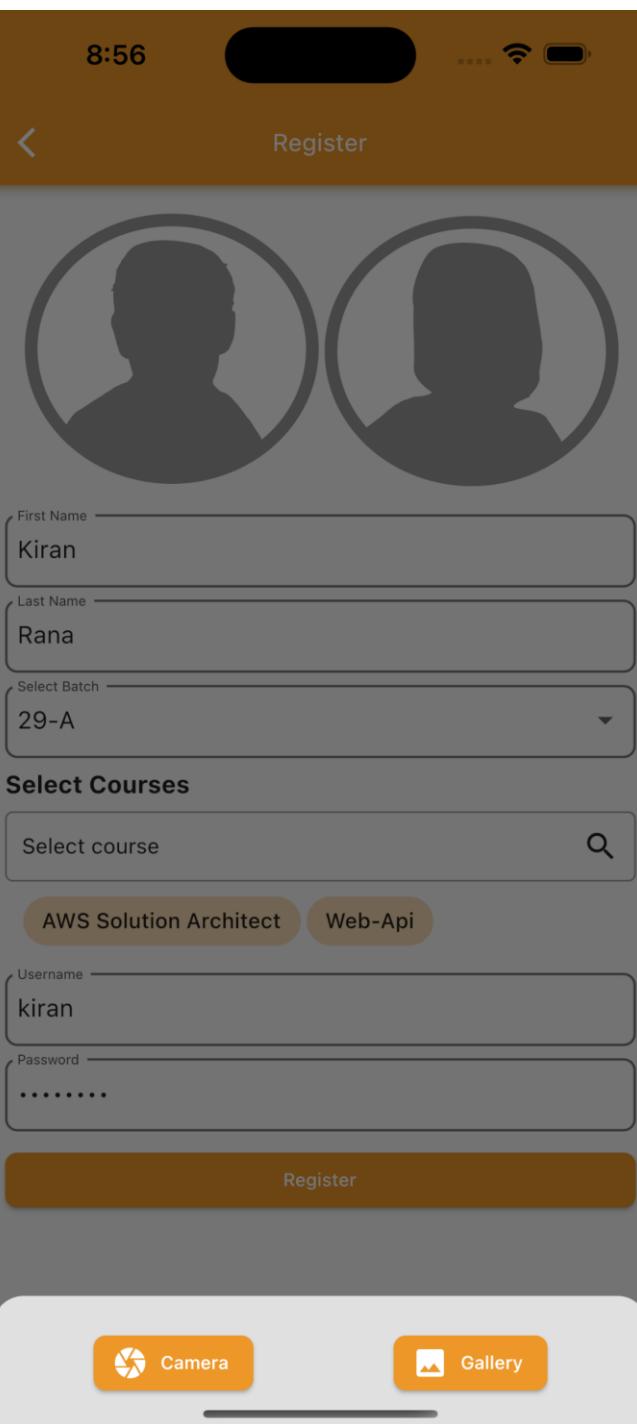
# Step 2 - Add permissions to AndroidManifest.xml file

Goto Your\_Flutter\_Project -> android -> app -> src -> main -> AndroidManifest.xml



```
1 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
2   package="com.example.runtime_permission">
3
4   <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
5   <uses-permission android:name="android.permission.CAMERA"/>
6
7   <application
8     android:label="runtime_permission"
9     android:name="${applicationName}"
10    android:icon="@mipmap/ic_launcher">
11      <activity ...
12        </activity>
13        <!-- Don't delete the meta-data below.
14            | This is used by the Flutter tool to generate GeneratedPluginRegis
15        <meta-data
16          android:name="flutterEmbedding"
17          android:value="2" />
18      </application>
19  </manifest>
```

# For iOS



# For iOS

The screenshot shows a file explorer interface with the following structure:

- ios
  - .symlinks
  - Flutter
  - Pods
- Runner
  - Assets.xcassets
  - Base.lproj
  - AppDelegate.swift
  - GeneratedPluginRegistrant.h
  - GeneratedPluginRegistrant.m
- Info.plist (selected)
- Runner-Bridging-Header.h
- Runner.xcodeproj
- Runner.xcworkspace
- .gitignore
- Podfile
- Podfile.lock

At the bottom left, the date is shown as 1/29/2025.

The screenshot shows the content of the Info.plist file:

```
ios > Runner > Info.plist
```

```
40 <string>UIInterfaceOrientationPortraitUpsideDown</string>
41 <string>UIInterfaceOrientationLandscapeLeft</string>
42 <string>UIInterfaceOrientationLandscapeRight</string>
43 </array>
44 <key>UIViewControllerBasedStatusBarAppearance</key>
45 <false/>
46 <key>CADisableMinimumFrameDurationOnPhone</key>
47 <true/>
48 <key>UIApplicationSupportsIndirectInputEvents</key>
49 <true/>
50
51 <key>NSPhotoLibraryUsageDescription</key>
52 <string>Upload images for screen background</string>
53 <key>NSCameraUsageDescription</key>
54 <string>Upload image from camera for screen background</string>
55 <key>NSMicrophoneUsageDescription</key>
56 <string>Post videos to profile</string>
57
58 </dict>
59 </plist>
```

A red box highlights the following permissions in the Info.plist file:

- <key>NSPhotoLibraryUsageDescription</key>  
<string>Upload images for screen background</string>
- <key>NSCameraUsageDescription</key>  
<string>Upload image from camera for screen background</string>
- <key>NSMicrophoneUsageDescription</key>  
<string>Post videos to profile</string>

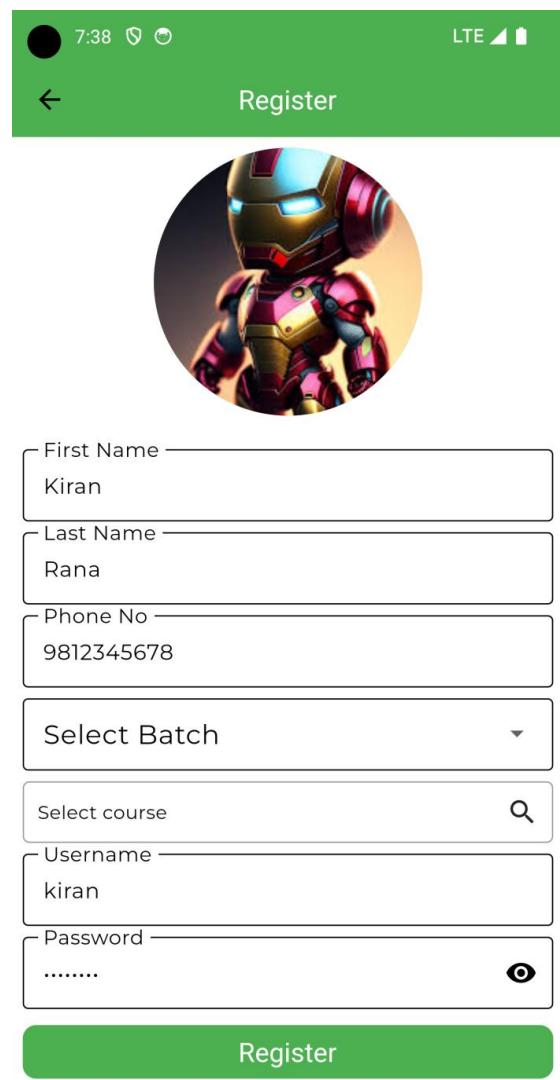
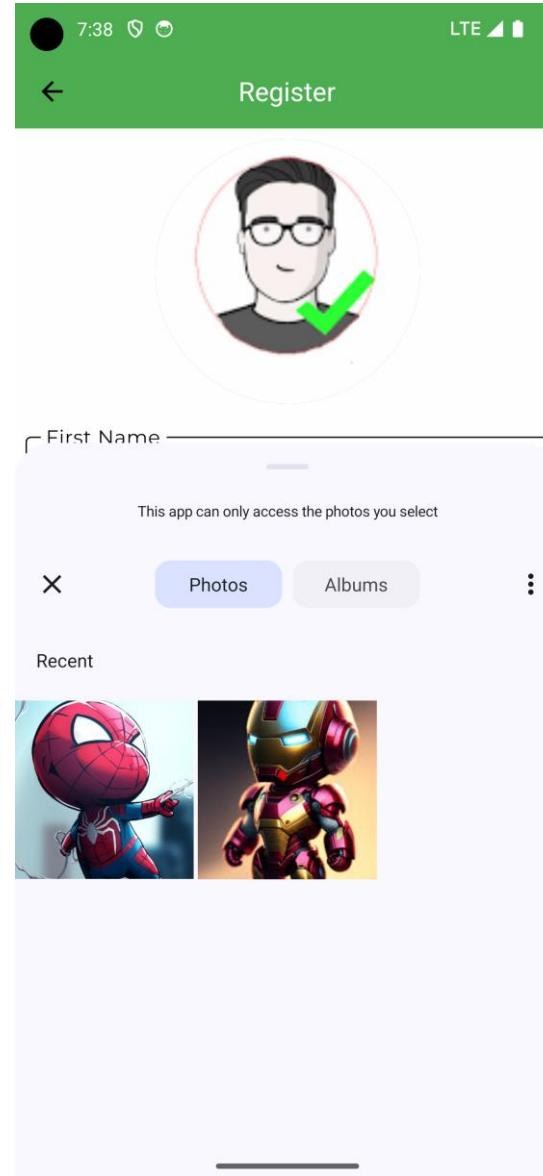
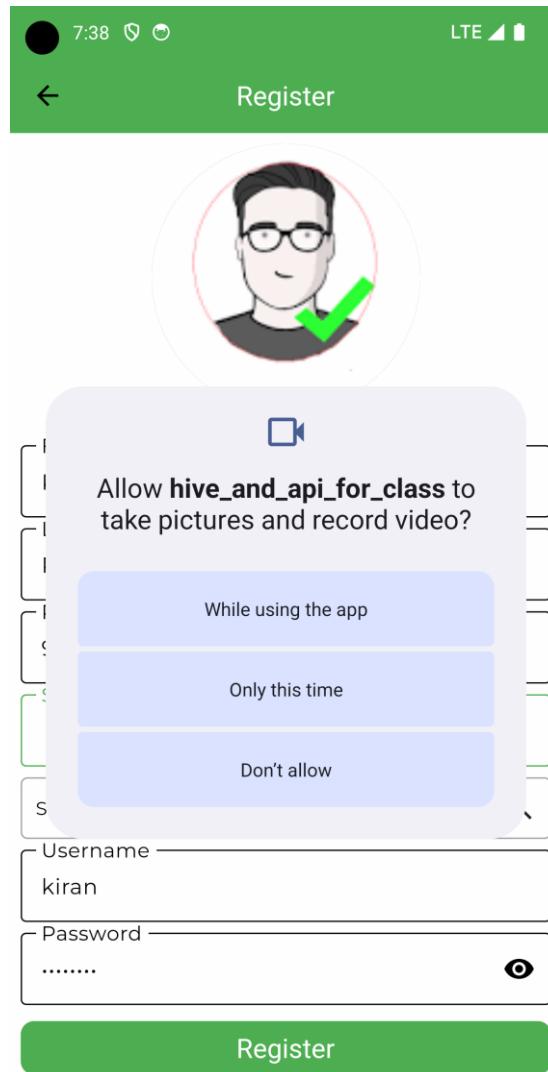
At the bottom right, it says "Prepared by: Kiran Rana".

# Copy paste this text in Info.plist

```
<key>NSPhotoLibraryUsageDescription</key>
<string>Upload images for screen background</string>
<key>NSCameraUsageDescription</key>
<string>Upload image from camera for screen background</string>
<key>NSMicrophoneUsageDescription</key>
<string>Post videos to profile</string>
```

# Run time permission and Camera

```
image_picker: ^0.8.9  
permission_handler: ^10.3.0
```



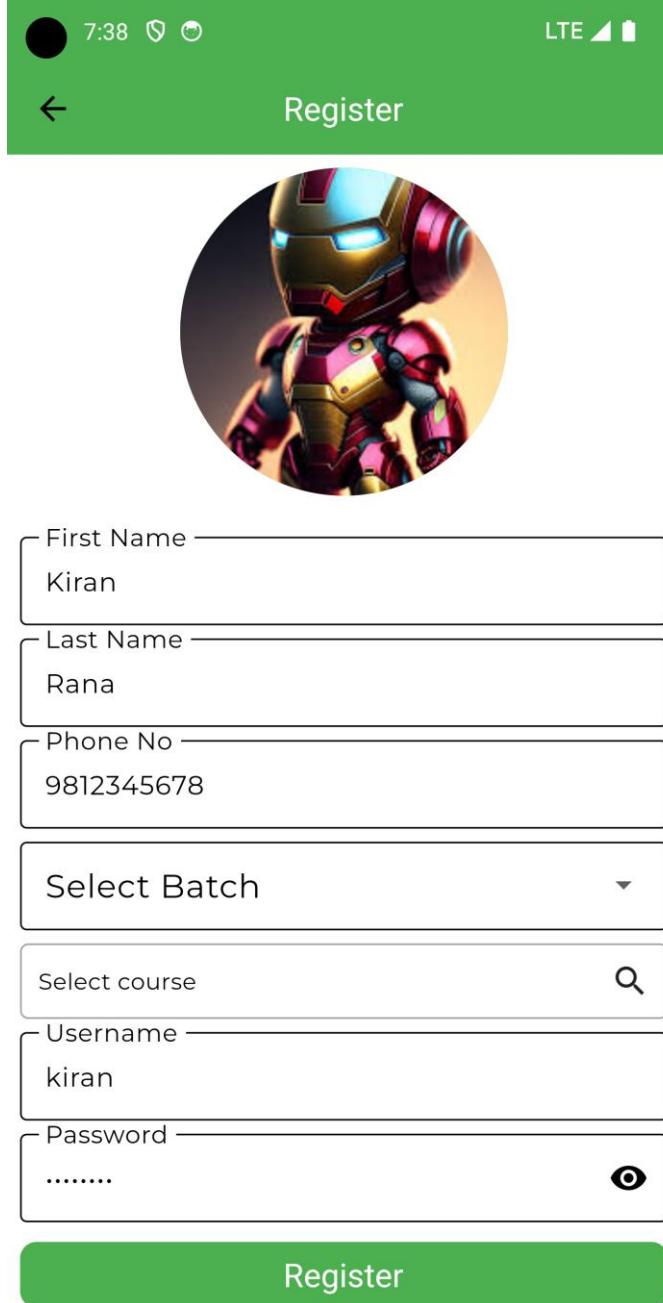
# Step 1 : Register page

```
checkCameraPermission() async {
  if (await Permission.camera.request().isRestricted ||
      await Permission.camera.request().isDenied) {
    await Permission.camera.request();
  }
}
```

```
43 | File? _img;
44 | Future _browseImage(ImageSource imageSource) async {
45 |   try {
46 |     final image = await ImagePicker().pickImage(source: imageSource);
47 |     if (image != null) {
48 |       setState(() {
49 |         _img = File(image.path);
50 |         // Send image to server
51 |       });
52 |     } else {
53 |       return;
54 |     }
55 |   } catch (e) {
56 |     debugPrint(e.toString());
57 |   }
58 | }
```

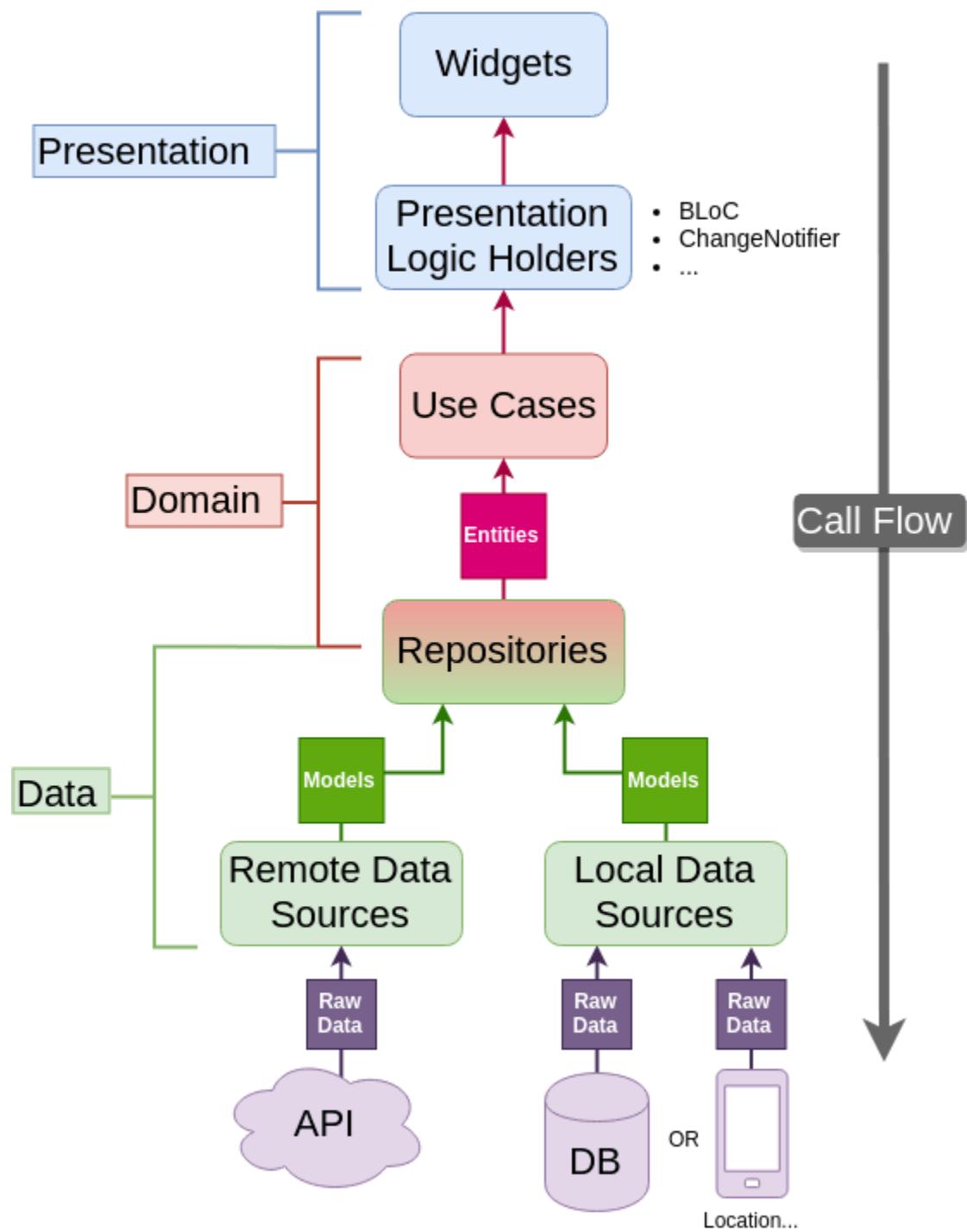
# Step 2 : Load image in imageview

```
child: SizedBox(  
    height: 200,  
    width: 200,  
    child: CircleAvatar(  
        radius: 50,  
        backgroundImage: _img != null  
            ? FileImage(_img!)  
            : const AssetImage('assets/images/profile.png')  
                as ImageProvider,  
        ), // CircleAvatar  
, // SizedBox
```



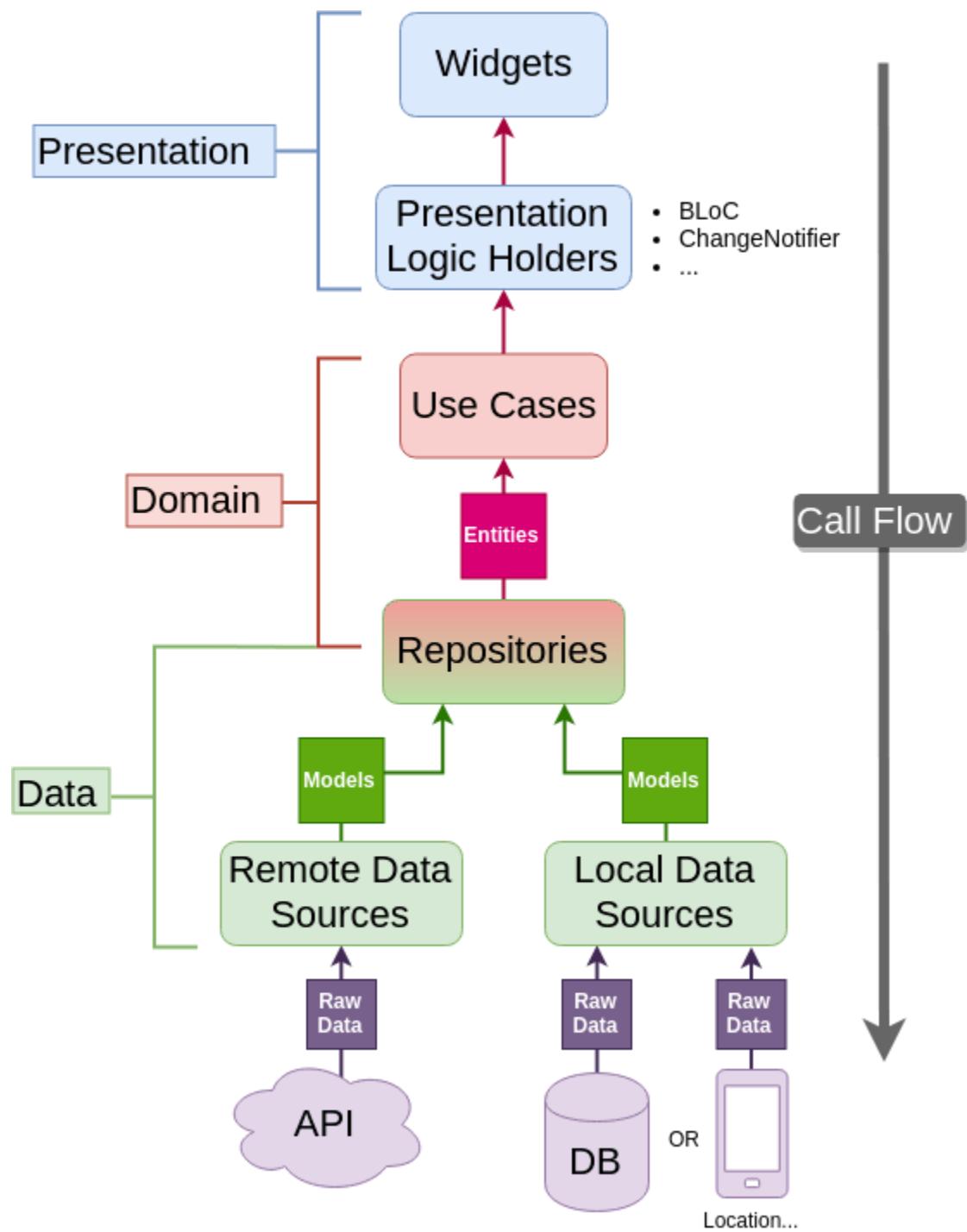
# Step 3 : On Camera click

```
ElevatedButton.icon(  
    onPressed: () {  
        checkCameraPermission();  
        _browseImage(ImageSource.camera);  
        Navigator.pop(context);  
    },  
    icon: const Icon(Icons.camera),  
    label: const Text('Camera'),  
, // ElevatedButton.icon  
ElevatedButton.icon(  
    onPressed: () {  
        _browseImage(ImageSource.gallery);  
        Navigator.pop(context);  
    },  
    icon: const Icon(Icons.image),  
    label: const Text('Gallery'),  
, // ElevatedButton.icon
```



# Step 4 : Auth Repository

```
7 abstract interface class IAuthRepository {  
8     Future<Either<Failure, void>> registerStudent(AuthEntity student);  
9  
10    Future<Either<Failure, String>> loginStudent(String username, String password);  
11  
12    Future<Either<Failure, String>> uploadProfilePicture(File file);  
13  
14    Future<Either<Failure, AuthEntity>> getCurrentUser();  
15 }
```



# Step 5 : Auth Remote Data Source

The screenshot shows the Postman application interface. At the top, it displays a POST request to `localhost:3000/api/v1/auth/uploadImage`. The 'Body' tab is selected, showing a form-data payload with two fields: `profilePicture` (selected as a file) and `fname` (set to `sdasd`). The response at the bottom is a 200 OK status with a response time of 7 ms, a body size of 998 B, and a global icon. The JSON response is displayed below, with the `"data": "IMG-1737981750996.png"` field highlighted by a red box.

POST localhost:3000/api/v1/auth/uploadImage Send

Params Authorization Headers (9) **Body** Scripts Settings Cookies

none  form-data  x-www-form-urlencoded  raw  binary  GraphQL

Key	Value	Description	Bulk Edit
<input checked="" type="checkbox"/> profilePicture	File <input type="button" value="Screenshot 2025-01-27 18..."/>		
<input type="checkbox"/> fname	Text <input type="button" value="sdasd"/>		

Body Cookies (1) Headers (20) Test Results 200 OK • 7 ms • 998 B •

{ } JSON ▾ Preview ⚡ Visualize

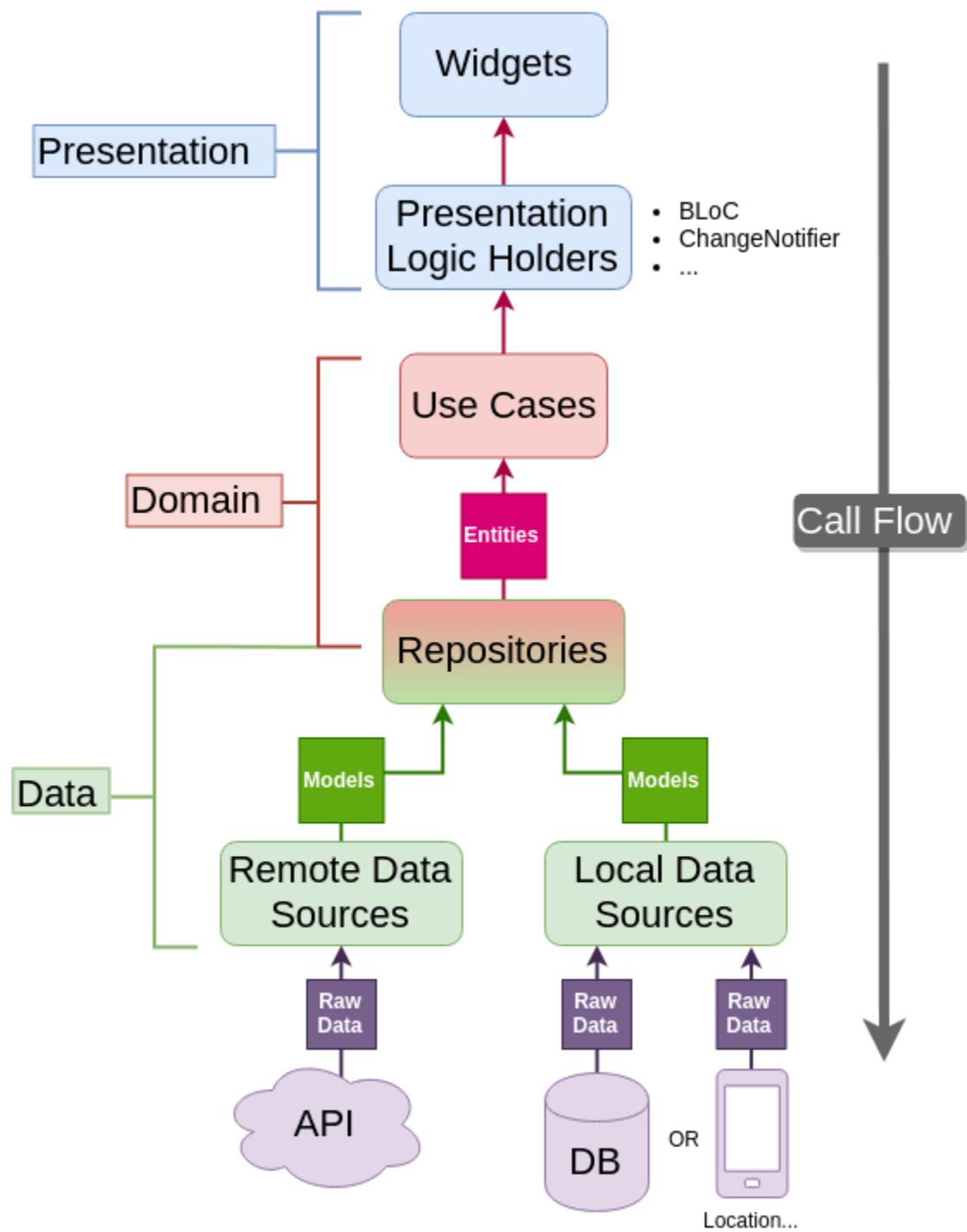
```
1 {  
2   "success": true,  
3   "data": "IMG-1737981750996.png"  
4 }
```

# Step 5 : Auth Remote Data Source

```
54 @override
55 Future<String> uploadProfilePicture(File file) async {
56   try {
57     String fileName = file.path.split('/').last;
58     FormData formData = FormData.fromMap(
59       {
60         'profilePicture': await MultipartFile.fromFile(
61           file.path,
62           filename: fileName,
63           ),
64       },
65     );
66
67     Response response = await _dio.post(
68       ApiEndpoints.uploadImage,
69       data: formData,
70     );
71
72     if (response.statusCode == 200) {
73       // Extract the image name from the response
74       return response.data['data'];
75     } else {
76       throw Exception(response.statusText);
77     }
78   } on DioException catch (e) {
79     throw Exception(e);
80   } catch (e) {
81     throw Exception(e);
82   }
83 }
84 }
```

# Sep 6 : Auth Remote Repository

```
37     @override
38     Future<Either<Failure, String>> uploadProfilePicture(File file) async {
39         try {
40             final imageName = await _authRemoteDataSource.uploadProfilePicture(file);
41             return Right(imageName);
42         } catch (e) {
43             return Left(ApiFailure(message: e.toString()));
44         }
45     }
46 }
```



## Step 6 : Use Case

```
8  class UploadImageParams {  
9      final File file;  
10  
11     const UploadImageParams({  
12         required this.file,  
13     });  
14 }  
15  
16 class UploadImageUsecase  
17     implements UsecaseWithParams<String, UploadImageParams> {  
18     final IAuthRepository _repository;  
19  
20     UploadImageUsecase(this._repository);  
21  
22     @override  
23     Future<Either<Failure, String>> call(UploadImageParams params) {  
24         return _repository.uploadProfilePicture(params.file);  
25     }  
26 }
```

# Step 7 : Auth View Model (Auth State)

```
1 part of 'register_bloc.dart';
2
3 class RegisterState extends Equatable {
4   final bool isLoading;
5   final bool isSuccess;
6   final String? imageName;
7
8   const RegisterState({
9     required this.isLoading,
10    required this.isSuccess,
11    this.imageName,
12  });
13
14 const RegisterState.initial()
15   : isLoading = false,
16   isSuccess = false,
17   imageName = null;
18
19 RegisterState copyWith({
20   bool? isLoading,
21   bool? isSuccess,
22   String? imageName,
23 }) {
24   return RegisterState(
25     isLoading: isLoading ?? this.isLoading,
26     isSuccess: isSuccess ?? this.isSuccess,
27     imageName: imageName ?? this.imageName,
28   );
29 }
30
31 @override
32 List<Object?> get props => [isLoading, isSuccess, imageName];
33 }
```

# Step 7 : Auth Event

```
1 part of 'register_bloc.dart';
2
3 sealed class RegisterEvent extends Equatable {
4   const RegisterEvent();
5
6   @override
7   List<Object> get props => [];
8 }
9
10 class LoadCoursesAndBatches extends RegisterEvent {}
11
12 class LoadImage extends RegisterEvent {
13   final File file;
14
15   const LoadImage({
16     required this.file,
17   });
18 }
19
20 class RegisterStudent extends RegisterEvent { ...
41 |
```

# Step 7 : Auth Bloc

```
17 class RegisterBloc extends Bloc<RegisterEvent, RegisterState> {
18   final BatchBloc _batchBloc;
19   final CourseBloc _courseBloc;
20   final RegisterUseCase _registerUseCase;
21   final UploadImageUsecase _uploadImageUsecase;  

22
23   RegisterBloc({
24     required BatchBloc batchBloc,
25     required CourseBloc courseBloc,
26     required RegisterUseCase registerUseCase,
27     required UploadImageUsecase uploadImageUsecase,  

28   ) : _batchBloc = batchBloc,
29       _courseBloc = courseBloc,
30       _registerUseCase = registerUseCase,
31       _uploadImageUsecase = uploadImageUsecase,  

32       super(RegisterState.initial()) {
33     on<LoadCoursesAndBatches>(_onLoadCoursesAndBatches);
34     on<RegisterStudent>(_onRegisterEvent);
35     on<LoadImage>(_onLoadImage);
36
37     add(LoadCoursesAndBatches());
38 }
```

# Step 7 :Auth Bloc

```
75 void _onLoadImage(  
76   LoadImage event,  
77   Emitter<RegisterState> emit,  
78 ) async {  
79   emit(state.copyWith(isLoading: true));  
80   final result = await _uploadImageUsecase.call(  
81     UploadImageParams(  
82       file: event.file,  
83     ),  
84   );  
85  
86   result.fold(  
87     (l) => emit(state.copyWith(isLoading: false, isSuccess: false)),  
88     (r) {  
89       emit(state.copyWith(isLoading: false, isSuccess: true, imageName: r));  
90     },  
91   );  
92 }  
93 }
```

# Step 8 : Register UI

```
43 |     File? _img;
44 |     Future _browseImage(ImageSource imageSource) async {
45 |         try {
46 |             final image = await ImagePicker().pickImage(source: imageSource);
47 |             if (image != null) {
48 |                 setState(() {
49 |                     img = File(image.path);
50 |                     // Send image to server
51 |                     context.read<RegisterBloc>().add(
52 |                         LoadImage(file: _img!),
53 |                     );
54 |                 });
55 |             } else {
56 |                 return;
57 |             }
58 |         } catch (e) {
59 |             debugPrint(e.toString());
60 |         }
61 |     }
```

## Step 9 : Get It

```
80 // ===== Usecases =====
81 getIt.registerLazySingleton<RegisterUseCase>(
82     () => RegisterUseCase(
83         getIt<AuthRemoteRepository>(),
84     ),
85 );
86
87 getIt.registerLazySingleton<UploadImageUsecase>(
88     () => UploadImageUsecase(
89         getIt<AuthRemoteRepository>(),
90     ),
91 );
92
93 getIt.registerFactory<RegisterBloc>(
94     () => RegisterBloc(
95         batchBloc: getIt<BatchBloc>(),
96         courseBloc: getIt<CourseBloc>(),
97         registerUseCase: getIt(),
98         uploadImageUsecase: getIt(),
99     ),
100 );
101 }
```

# Now run the program

The image shows a mobile application interface for "Register Student". At the top, there is a navigation bar with icons for back, forward, and other functions. The main screen has a green header with the title "Register Student". Below the header, there is a circular profile picture of a man with short dark hair, smiling. Underneath the profile picture, there are four input fields:

- First Name: kiran
- Last Name: rana
- Phone No: 123456789
- Select Batch: (dropdown menu)

To the right of the application screen, there is a vertical toolbar with various icons for audio, camera, search, and other functions.

At the bottom left, there is a screenshot of a browser or file manager showing two files: "permission\_denied.txt" and "IMG-1737983194414.jpg". The "IMG-1737983194414.jpg" file is displayed as a thumbnail of a man in a maroon sweater.

The bottom of the screen also shows a timeline with several entries:

- 1.991 ms - 166
- 1.955 ms - 166
- 1.424 ms - 221
- 56.907 ms - 47
- 2.370 ms - 221

# Now insert the image name in database

The screenshot shows the MongoDB Compass interface. On the left, the connection list includes 'localhost:27017' with databases 'admin', 'config', 'local', and 'student\_batch'. The 'student\_batch' database is expanded, showing 'batches', 'courses', and 'students'. The 'students' collection is selected and highlighted in grey. On the right, the document view displays the following data:

```
_id: ObjectId('6797a262247d15339a7ad2bb')
fname : "kiran"
lname : "rana"
phone : "123456789"
image : "IMG-1737990740492.jpg"
username : "kiran"
password : "$2a$10$kJ5VQm7LdCX2BE91rpAM2uSGX3fu/kL48ky99AyZjEAjFq2GUU9Se"
batch : ObjectId('6792d751bf7332f19165cbae')
course : Array (2)
__v : 0
```

Below the document, there are buttons for 'ADD DATA', 'EXPORT DATA', 'UPDATE', and 'DELETE'.

# Step 1 : Register user usecase

```
10  class RegisterUserParams extends Equatable {  
11    final String fname;  
12    final String lname;  
13    final String phone;  
14    final BatchEntity batch;  
15    final List<CourseEntity> courses;  
16    final String username;  
17    final String password;  
18    final String? image;  
19  
20  const RegisterUserParams({  
21    required this.fname,  
22    required this.lname,  
23    required this.phone,  
24    required this.batch,  
25    required this.courses,  
26    required this.username,  
27    required this.password,  
28    this.image,  
29  });
```

# 1. Register Use Case

```
48 class RegisterUseCase implements UsecaseWithParams<void, RegisterUserParams> {  
49     final IAuthRepository repository;  
50  
51     RegisterUseCase(this.repository);  
52  
53     @override  
54     Future<Either<Failure, void>> call(RegisterUserParams params) {  
55         final authEntity = AuthEntity(  
56             fName: params.fname,  
57             lName: params.lname,  
58             phone: params.phone,  
59             batch: params.batch,  
60             courses: params.courses,  
61             username: params.username,  
62             password: params.password,  
63             image: params.image,  
64         );  
65         return repository.registerStudent(authEntity);  
66     }  
67 }
```

## 2. Register Event

```
20 class RegisterStudent extends RegisterEvent {  
21     final BuildContext context;  
22     final String fName;  
23     final String lName;  
24     final String phone;  
25     final BatchEntity batch;  
26     final List<CourseEntity> courses;  
27     final String username;  
28     final String password;  
29     final String? image;  
30  
31     const RegisterStudent(  
32         required this.context,  
33         required this.fName,  
34         required this.lName,  
35         required this.phone,  
36         required this.batch,  
37         required this.courses,  
38         required this.username,  
39         required this.password,  
40         this.image,  
41     );  
42 }
```

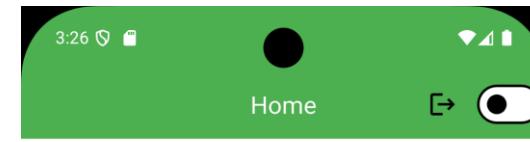
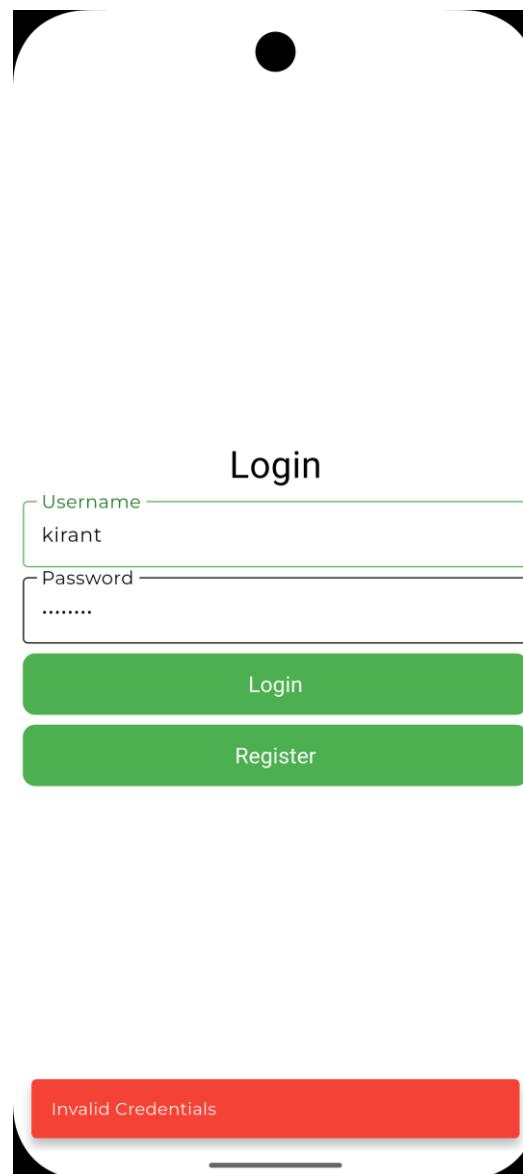
### 3. Register Bloc

```
50 void _onRegisterEvent(  
51   RegisterStudent event,  
52   Emitter<RegisterState> emit,  
53 ) async {  
54   emit(state.copyWith(isLoading: true));  
55   final result = await _registerUseCase.call(RegisterUserParams(  
56     fname: event.fName,  
57     lname: event.lName,  
58     phone: event.phone,  
59     batch: event.batch,  
60     courses: _courseBloc.state.courses,  
61     username: event.username,  
62     password: event.password,  
63     image: state.imageName,  
64   ));  
65  
66   result.fold(  
67     (l) => emit(state.copyWith(isLoading: false, isSuccess: false)),  
68     (r) {  
69       emit(state.copyWith(isLoading: false, isSuccess: true));  
70       showMySnackBar(  
71         context: event.context, message: "Registration Successful");  
72     },  
73   );  
74 }
```

## 4. Register View

```
sizedBox(  
    width: double.infinity,  
    child: ElevatedButton(  
        onPressed: () {  
            if (_key.currentState!.validate()) {  
                final registerState =  
                    context.read<RegisterBloc>().state;  
                final imageName = registerState.imageName;  
                context.read<RegisterBloc>().add(  
                    RegisterStudent(  
                        context: context,  
                        fName: _fnameController.text,  
                        lName: _lnameController.text,  
                        phone: _phoneController.text,  
                        batch: _dropDownValue!,  
                        courses: _lstCourseSelected,  
                        username: _usernameController.text,  
                        password: _passwordController.text,  
                        image: imageName,  
                    ), // Registerstudent  
                );  
            }  
        },  
        child: const Text('Register'),  
    ), // ElevatedButton  
, // SizedBox
```

# Login Page



Dashboard

POST

localhost:3000/api/v1/auth/login

Send

Params

Authorization

Headers (9)

Body

Scripts

Settings

Cookie

## Query Params

Key	Value	Description	...	Bulk Edit
Key	Value	Description		

Body

Cookies (1)

Headers (21)

Test Results



200 OK

• 65 ms

• 1.37 KB



e.g. Save Response



{ } JSON

▷ Preview

❖ Visualize



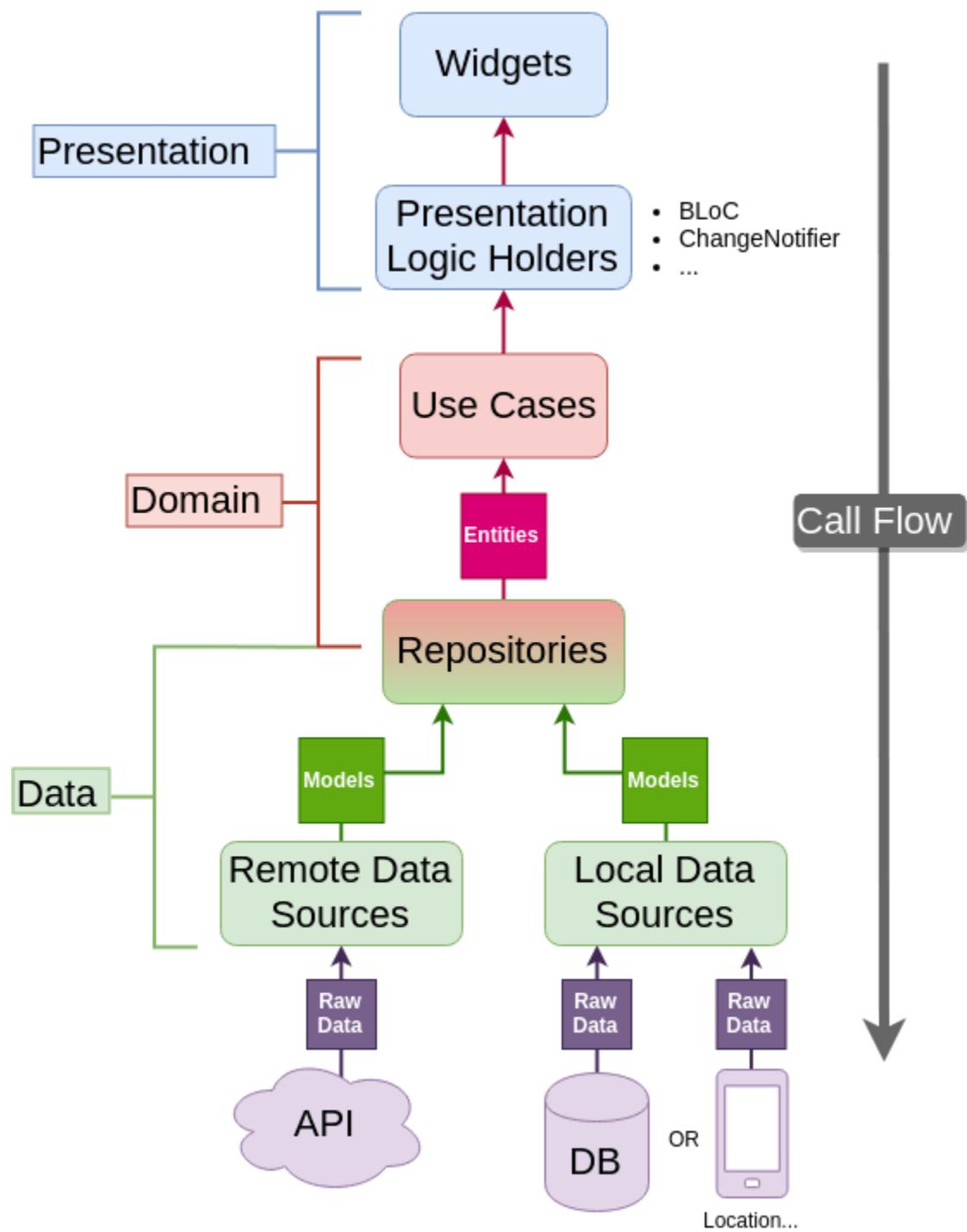
```
1  {
2      "success": true,
3      "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
4          eyJpZCI6IjY30TdhMjYyMjQ3ZDE1MzM5YTdhZDJiYiIsImhdCI6MTczODA1NzQ2MiwiZXhwIjoxNzQwNjQ5NDYyfQ.
5          NjicKaM41M4q8Yw1AebpVgsSP-6bLkVvspSidwwETIk"
```

## Step 1 : Auth Remote Data Source

```
48 @override
49 Future<String> loginStudent(String username, String password) async {
50   try {
51     Response response = await _dio.post(
52       ApiEndpoints.login,
53       data: {
54         "username": username,
55         "password": password,
56       },
57     );
58
59     if (response.statusCode == 200) {
60       final str = response.data['token'];
61       return str;
62     } else {
63       throw Exception(response.statusMessage);
64     }
65   } on DioException catch (e) {
66     throw Exception(e);
67   } catch (e) {
68     throw Exception(e);
69   }
70 }
```

## Step 2 : Auth Remote Repository

```
30     @override
31     Future<Either<Failure, String>> loginStudent(
32         String username, String password) async {
33     try {
34         final token =
35             await _authRemoteDataSource.loginStudent(username, password);
36         return Right(token);
37     } catch (e) {
38         return Left(ApiFailure(message: e.toString()));
39     }
40 }
```



## Step 3 : GetIt

---

```
206 _initLoginDependencies() async {
207 | // ===== Usecases =====
208 | getIt.registerLazySingleton<LoginUseCase>(
209 |   () => LoginUseCase(
210 |     getIt<AuthRemoteRepository>(),
211 |   ),
212 | );
213 |
214 | getIt.registerFactory<LoginBloc>(
215 |   () => LoginBloc(
216 |     registerBloc: getIt<RegisterBloc>(),
217 |     homeCubit: getIt<HomeCubit>(),
218 |     loginUseCase: getIt<LoginUseCase>(),
219 |   ),
220 | );
221 }
222 |
223 _initSplashScreenDependencies() async {
224 | getIt.registerFactory<SplashCubit>(
225 |   () => SplashCubit(getIt<LoginBloc>()),
226 | );
227 }
```

Now save the token in Shared Preferences

## Step 1 : Shared Pref Failure

```
3 abstract class Failure extends Equatable {  
4   final String message;  
5  
6   const Failure({required this.message});  
7  
8   @override  
9   List<Object> get props => [message];  
10 }  
11  
12 +class LocalDatabaseFailure extends Failure { ...  
15  
16 +class ApiFailure extends Failure { ...  
23  
24 class SharedPrefsFailure extends Failure {  
25   const SharedPrefsFailure(  
26     | required super.message,  
27     | );  
28 }
```

## Step 2 : Token Shared Preferences

```
5 class TokenSharedPrefs {  
6     final SharedPreferences _sharedPreferences;  
7  
8     TokenSharedPrefs(this._sharedPreferences);  
9  
10    Future<Either<Failure, void>> saveToken(String token) async {  
11        try {  
12            await _sharedPreferences.setString('token', token);  
13            return Right(null);  
14        } catch (e) {  
15            return Left(SharedPrefsFailure(message: e.toString()));  
16        }  
17    }  
18  
19    Future<Either<Failure, String>> getToken() async {  
20        try {  
21            final token = _sharedPreferences.getString('token');  
22            return Right(token ?? '');  
23        } catch (e) {  
24            return Left(SharedPrefsFailure(message: e.toString()));  
25        }  
26    }  
27}
```

## Step 3 : Login Use Case

```
26 class LoginUseCase implements UsecaseWithParams<String, LoginParams>
27     final IAuthRepository repository;
28     final TokenSharedPrefs tokenSharedPrefs;
29
30     LoginUseCase(this.repository, this.tokenSharedPrefs);
31
32     @override
33     Future<Either<Failure, String>> call(LoginParams params) {
34         // Save token in Shared Preferences
35         return repository
36             .loginStudent(params.username, params.password)
37             .then((value) {
38                 return value.fold(
39                     (failure) => Left(failure),
40                     (token) {
41                         tokenSharedPrefs.saveToken(token);
42                         tokenSharedPrefs.getToken().then((value) {
43                             print(value);
44                         });
45                         return Right(token);
46                     },
47                 );
48             });
49     }
50 }
```

## Step 4 : GetIt

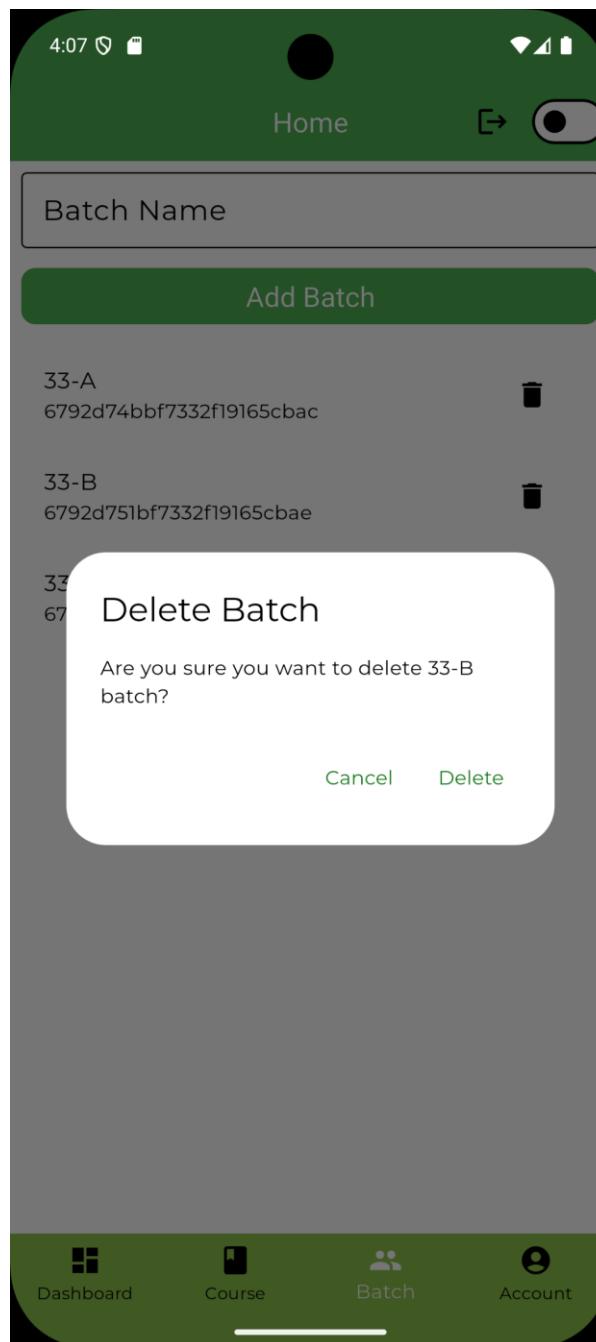
---

```
37 Future<void> initDependencies() async {  
38     // First initialize hive service  
39     await _initHiveService();  
40     await _init ApiService();  
41     await _initSharedPreferences();  
42     await _initBatchDependencies();  
43     await _initCourseDependencies();  
44     await _initHomeDependencies();  
45     await _initRegisterDependencies();  
46     await _initLoginDependencies();  
47  
48     await _initSplashScreenDependencies();  
49 }  
50  
51 Future<void> _initSharedPreferences() async {  
52     final SharedPreferences = await SharedPreferences.getInstance();  
53     getIt.registerLazySingleton<SharedPreferences>(() => sharedPreferences);  
54 }
```

# Step 4 : getIt

```
213     _initLoginDependencies() async {
214         // ===== Token Shared Preferences =====
215         getIt.registerLazySingleton<TokenSharedPrefs>(
216             () => TokenSharedPrefs(getIt<SharedPreferences>()),
217         );
218
219         // ===== Usecases =====
220         getIt.registerLazySingleton<LoginUseCase>(
221             () => LoginUseCase(
222                 getIt<AuthRemoteRepository>(),
223                 getIt<TokenSharedPrefs>(),
224             ),
225         );
226
227         getIt.registerFactory<LoginBloc>(
228             () => LoginBloc(
229                 registerBloc: getIt<RegisterBloc>(),
230                 homeCubit: getIt<HomeCubit>(),
231                 loginUseCase: getIt<LoginUseCase>(),
232             ),
233         );
234     }
```

# Delete Batch



# Postman

The screenshot shows the Postman application interface with the following details:

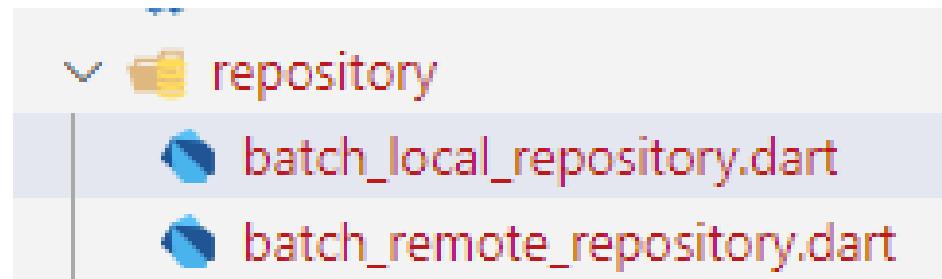
- Method:** `DELETE`
- URL:** `localhost:3000/api/v1/batch/6792d74bbf7332f19165cbac`
- Authorization:** `Bearer Token`
- Token:** eyJhbGciOiJIUzI1NilsInR5cCI6IkpXVCJ9.eyJ...
- Body:** `{ } JSON`
- Headers:** (8)
- Test Results:** (20)
- Status:** 200 OK
- Response Time:** 22 ms
- Response Size:** 1.01 KB
- Save Response:** `e.g. Save Response`
- Preview:** `{ "success": true, "data": { "_id": "6792d74bbf7332f19165cbac", "batchName": "33-A", "__v": 0 } }`

# Step 1 : Api Endpoints

```
--  
21 // ====== Batch Routes ======  
22 static const String createBatch = "batch/createBatch";  
23 static const String getAllBatch = "batch/getAllBatches";  
24 static const String deleteBatch = "batch/";
```

# Step 2 : Batch Repository

```
5 abstract interface class IBatchRepository {  
6   Future<Either<Failure, List<BatchEntity>>> getBatches();  
7   Future<Either<Failure, void>> createBatch(BatchEntity batch);  
8   Future<Either<Failure, void>> deleteBatch(String id, String? token);  
9 }
```

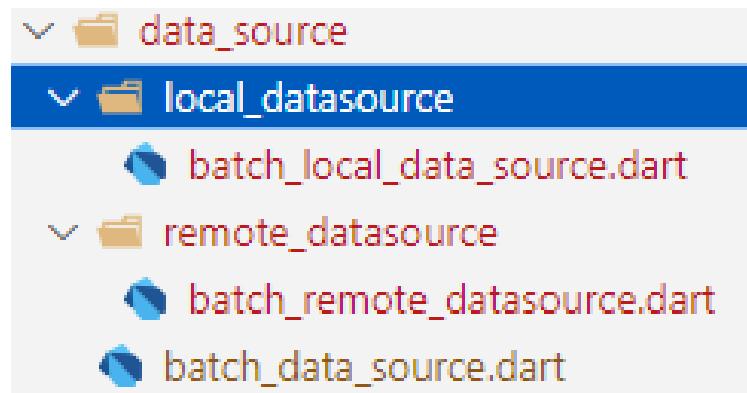


# Step 3 : Batch Delete UseCase

```
21 class DeleteBatchUsecase implements UsecaseWithParams<void, DeleteBatchParams> {  
22     final IBatchRepository batchRepository;  
23     final TokenSharedPrefs tokenSharedPrefs;  
24  
25     DeleteBatchUsecase(  
26         {required this.batchRepository, required this.tokenSharedPrefs});  
27  
28     @override  
29     Future<Either<Failure, void>> call(DeleteBatchParams params) async {  
30         // Get token from Shared Preferences and send it to the server  
31         final token = await tokenSharedPrefs.getToken();  
32         return token.fold((l) {  
33             return Left(l);  
34         }, (r) async {  
35             return await batchRepository.deleteBatch(params.batchId, r);  
36         });  
37     }  
38 }
```

# Batch Data Source

```
3 abstract interface class IDataSource {  
4     Future<List<BatchEntity>> getBatches();  
5     Future<void> createBatch(BatchEntity batch);  
6     Future<void> deleteBatch(String id, String? token);  
7 }
```



# Postman

The screenshot shows the Postman application interface with the following details:

- Method:** `DELETE`
- URL:** `localhost:3000/api/v1/batch/6792d74bbf7332f19165cbac`
- Authorization:** `Bearer Token`
- Token:** eyJhbGciOiJIUzI1NilsInR5cCI6IkpXVCJ9.eyJ...
- Body:** `{ } JSON`
- Headers:** (8)
- Test Results:** (20)
- Status:** 200 OK
- Response Time:** 22 ms
- Response Size:** 1.01 KB
- Save Response:** `e.g. Save Response`
- Preview:** `{ "success": true, "data": { "_id": "6792d74bbf7332f19165cbac", "batchName": "33-A", "__v": 0 } }`

# Step 4: Batch Remote Data Source

---

```
36 @override
37 Future<void> deleteBatch(String id, String? token) async {
38   try {
39     var response = await _dio.delete(
40       ApiEndpoints.deleteBatch + id,
41       options: Options(
42         headers: {
43           'Authorization': 'Bearer $token',
44         },
45       ),
46     );
47
48     if (response.statusCode == 200) {
49       return;
50     } else {
51       throw Exception(response.statusText);
52     }
53   } on DioException catch (e) {
54     throw Exception(e);
55   } catch (e) {
56     throw Exception(e);
57   }
58 }
```

# Step 5: Batch Remote Repository

```
26     @override
27     Future<Either<Failure, void>> deleteBatch(String id, String? token) async {
28         try {
29             remoteDataSource.deleteBatch(id, token);
30             return Right(null);
31         } catch (e) {
32             return Left(
33                 ApiFailure(
34                     message: e.toString(),
35                 ), // ApiFailure
36             ); // Left
37         }
38     }
```

# Step 6: Batch View

---

```
trailing: IconButton(  
    icon: Icon(Icons.delete),  
    onPressed: () {  
        showDialog(  
            context: context,  
            builder: (BuildContext context2) {  
                return AlertDialog(  
                    title: Text('Delete Batch'),  
                    content: Text(  
                        'Are you sure you want to delete ${state.batches[index].batchName} batch?'), // Text  
                    actions: [  
                        TextButton(  
                            child: Text('Cancel'),  
                            onPressed: () {  
                                Navigator.of(context).pop();  
                            },  
                        ), // TextButton  
                        TextButton(  
                            child: Text('Delete'),  
                            onPressed: () {  
                                context.read<BatchBloc>().add(  
                                    DeleteBatch(  
                                        state.batches[index].batchId!,  
                                    ), // DeleteBatch  
                                );  
  
                                Navigator.of(context).pop();  
                            },  
                        ), // TextButton  
                    ],  
                ); // AlertDialog  
            },  
        );  
    }, // IconButton
```

# Step : GetIt

---

```
183 // ===== Usecases =====
184
185 getIt.registerLazySingleton<CreateBatchUseCase>(
186   () => CreateBatchUseCase(batchRepository: getIt<BatchRemoteRepository>()),
187 );
188
189 getIt.registerLazySingleton<GetAllBatchUseCase>(
190   () => GetAllBatchUseCase(batchRepository: getIt<BatchRemoteRepository>()),
191 );
192
193 getIt.registerLazySingleton<DeleteBatchUsecase>(
194   () => DeleteBatchUsecase(
195     batchRepository: getIt<BatchRemoteRepository>(),
196     tokenSharedPrefs: getIt<TokenSharedPrefs>(),
197   ),
198 );
199
200 // ===== Bloc =====
201 getIt.registerFactory<BatchBloc>(
202   () => BatchBloc(
203     createBatchUseCase: getIt<CreateBatchUseCase>(),
204     getAllBatchUseCase: getIt<GetAllBatchUseCase>(),
205     deleteBatchUsecase: getIt<DeleteBatchUsecase>(),
206   ),
207 );
208 }
```