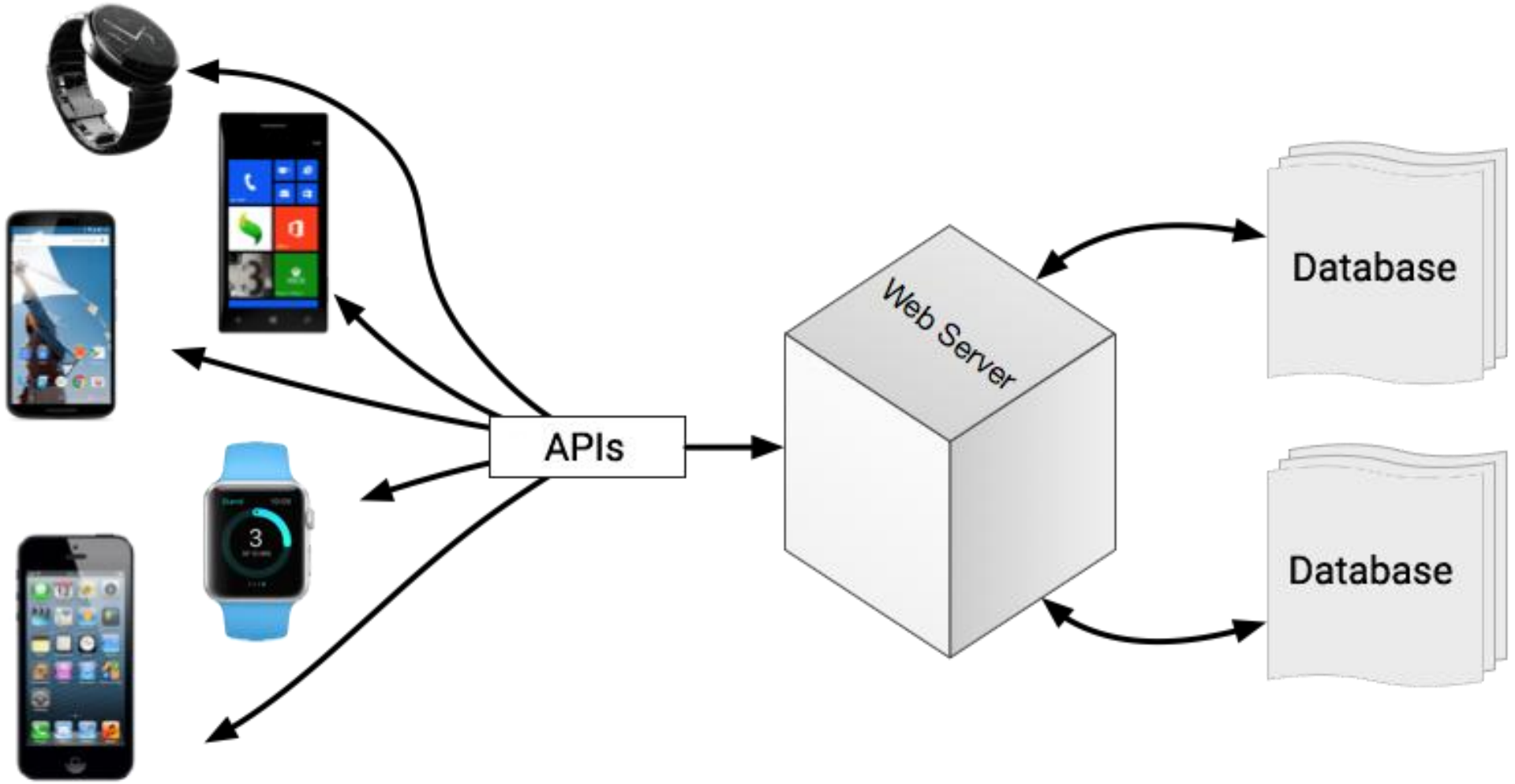
API (Application Programming Interface)

What is an API?

API is the acronym for **A**pplication **P**rogramming **I**nterface, 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



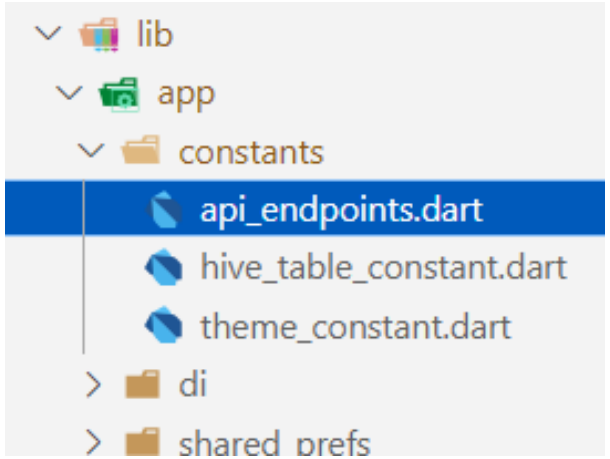
Data Interchange format : JSON vs XML

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: "ttl231",
    title: "Pride and Prejudice"
    year: 2093,
    director: "Michael Bay",
    genres: [
        "Horror",
        "Comedy"
    ],
    stars: [
        {
            name: "Kiera Knightley",
            id: 9863
        },
        {
            name: "Danny DeVito",
            id: 2031
        }
    ]
}
```

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

Step 1 : Constant




```dart
class ApiEndpoints {
  ApiEndpoints._();

  static const Duration connectionTimeout = Duration(seconds: 1000);
  static const Duration receiveTimeout = Duration(seconds: 1000);
  static const String baseUrl = "http://10.0.2.2:3000/api/v1/";
  // For iPhone
  //static const String baseUrl = "http://localhost:3000/api/v1/";

  // ====================== Auth Routes ======================
  static const String login = "auth/login";
  static const String register = "auth/register";
  static const String getAllStudent = "auth/getAllStudents";
  static const String getStudentsByBatch = "auth/getStudentsByBatch/";
  static const String getStudentsByCourse = "auth/getStudentsByCourse/";
  static const String updateStudent = "auth/updateStudent/";
  static const String deleteStudent = "auth/deleteStudent/";
  static const String imageUrl = "http://10.0.2.2:3000/uploads/";
  static const String uploadImage = "auth/uploadImage";

  // ====================== Batch Routes ======================
  static const String createBatch = "batch/createBatch";
  static const String getAllBatch = "batch/getAllBatches";

  // ====================== Course Routes ======================
  static const String createCourse = "course/createCourse";
  static const String deleteCourse = "course/";
  static const String getAllCourse = "course/getAllCourse";
}
```

Dio

dio 5.4.3+1

Published 58 days ago • flutter.cn Dart 3 compatible

SDK DART FLUTTER | PLATFORM ANDROID IOS LINUX MACOS WEB WINDOWS

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
```

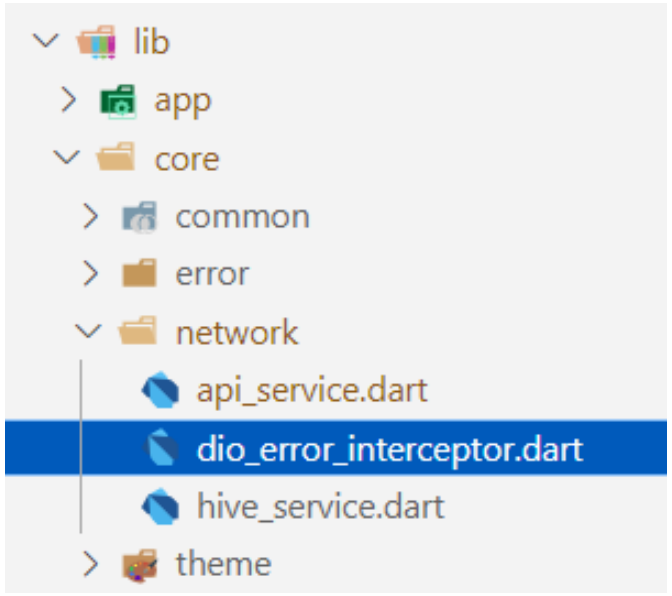
With Flutter:

```
$ 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




```dart
import 'package:dio/dio.dart';

class DioErrorInterceptor extends Interceptor {
  @override
  void onError(DioException err, ErrorInterceptorHandler handler) {
    if (err.response != null) {
      // Handle server errors
      if (err.response!.statusCode! >= 300) {
        err = DioException(
          requestOptions: err.requestOptions,
          response: err.response,
          error: err.response!.data['message'] ?? err.response!.statusMessage!,
          type: err.type,
        );
      } else {
        err = DioException(
          requestOptions: err.requestOptions,
          response: err.response,
          error: 'Something went wrong',
          type: err.type,
        );
      }
    } else {
      // Handle connection errors
      err = DioException(
        requestOptions: err.requestOptions,
        error: 'Connection error',
        type: err.type,
      );
    }
    super.onError(err, handler);
  }
}
```

Step 2 : Dio

```dart
class ApiService {
  final Dio _dio;

  Dio get dio => _dio;

  ApiService(this._dio) {
    _dio
      ..options.baseUrl = ApiEndpoints.baseUrl
      ..options.connectTimeout = ApiEndpoints.connectionTimeout
      ..options.receiveTimeout = ApiEndpoints.receiveTimeout
      ..interceptors.add(DioErrorInterceptor())
      ..interceptors.add(PrettyDioLogger(
          requestHeader: true, requestBody: true, responseHeader: true))
      ..options.headers = {
        'Accept': 'application/json',
        'Content-Type': 'application/json',
      };
  }
}
```

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
```

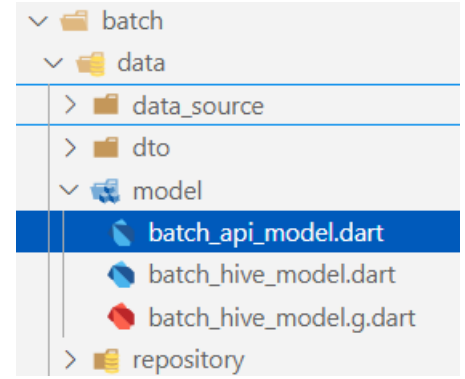
With Flutter:

```
$ 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




```
{
    "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
        },
```


```dart
@JsonSerializable()
class BatchApiModel extends Equatable {
  @JsonKey(name: '_id')
  final String? batchId;
  final String batchName;

  const BatchApiModel({
    this.batchId,
    required this.batchName,
  });

  const BatchApiModel.empty()
      : batchId = '',
        batchName = '';

```

Step 3 : Add toJson and fromJson

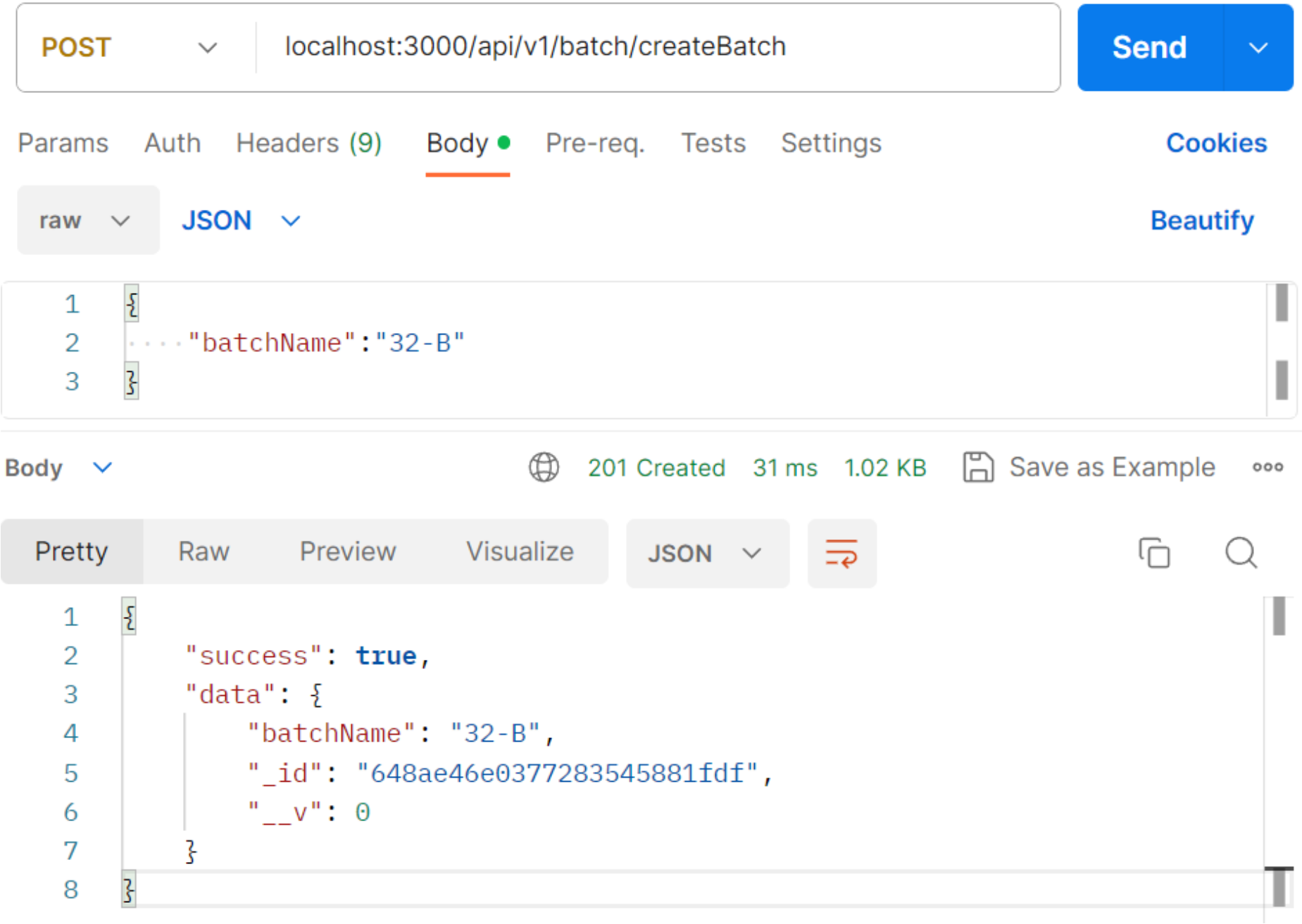
```dart
// From Json , write full code without generator
factory BatchApiModel.fromJson(Map<String, dynamic> json) {
  return BatchApiModel(
    batchId: json['_id'],
    batchName: json['batchName'],
  );
}

// To Json , write full code without generator
Map<String, dynamic> toJson() {
  return {
    '_id': batchId,
    'batchName': batchName,
  };
}
```


```json
{
    "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

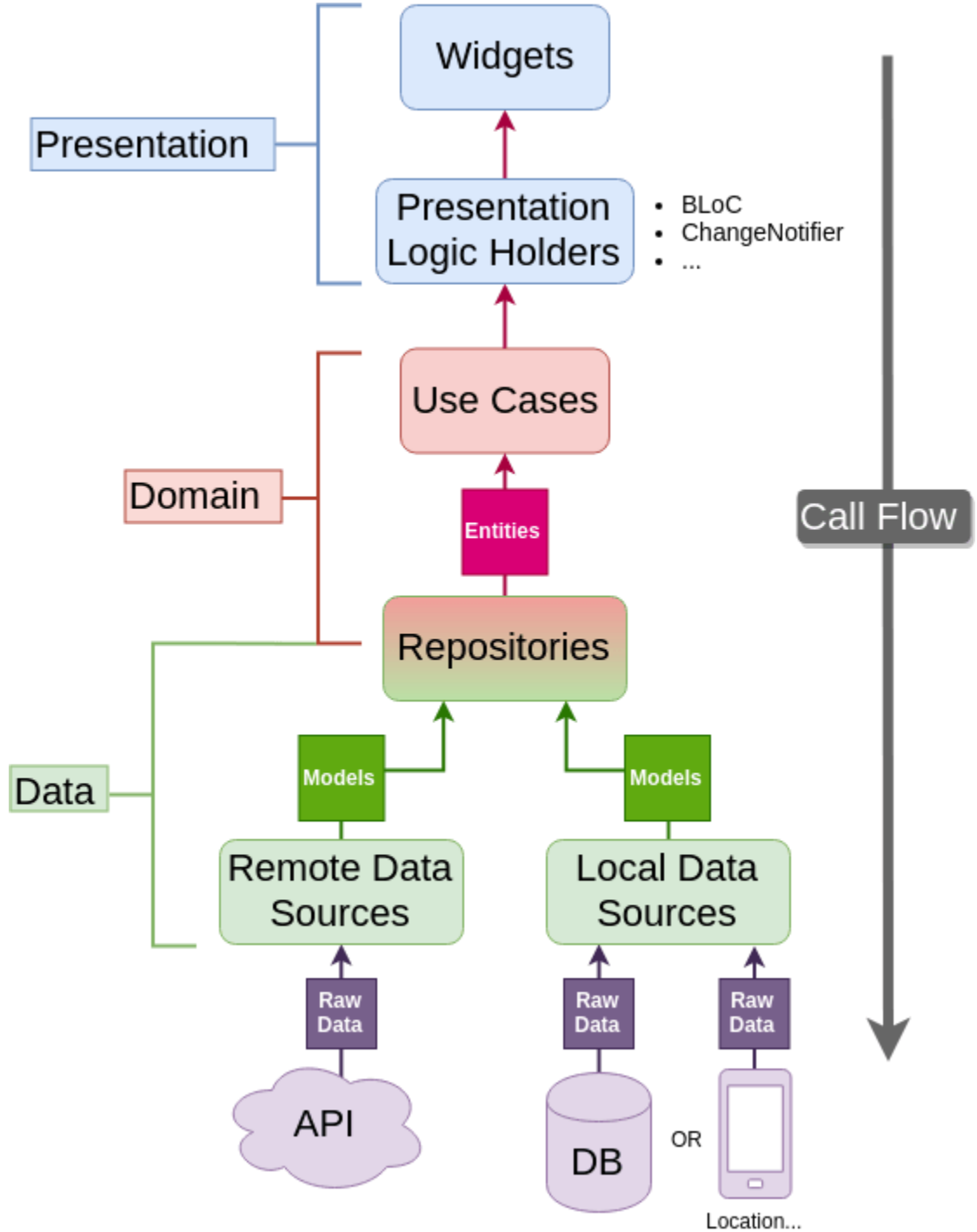
```dart
40    // Convert API Object to Entity
41    BatchEntity toEntity() => BatchEntity(
42          batchId: batchId,
43          batchName: batchName,
44        );
45
46    // Convert Entity to API Object
47    BatchApiModel fromEntity(BatchEntity entity) => BatchApiModel(
48          batchId: entity.batchId ?? '',
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

```dart
class BatchRemoteDataSource implements IBatchDataSource {
  final Dio _dio;

  BatchRemoteDataSource({
    required Dio dio,
  }) : _dio = dio;

  @override
  Future<void> createBatch(BatchEntity batch) async {
    try {
      // Convert entity to model
      var batchApiModel = BatchApiModel.fromEntity(batch);
      var response = await _dio.post(
        ApiEndpoints.createBatch,
        data: batchApiModel.toJson(),
      );
      if (response.statusCode == 201) {
        return;
      } else {
        throw Exception(response.statusMessage);
      }
    } on DioException catch (e) {
      throw Exception(e);
    } catch (e) {
      throw Exception(e);
    }
  }

  @override
  Future<void> deleteBatch(String id) {
    // TODO: implement deleteBatch
    throw UnimplementedError();
  }

  @override
  Future<List<BatchEntity>> getBatches() {
    // TODO: implement deleteBatch
    throw UnimplementedError();
  }
}
```

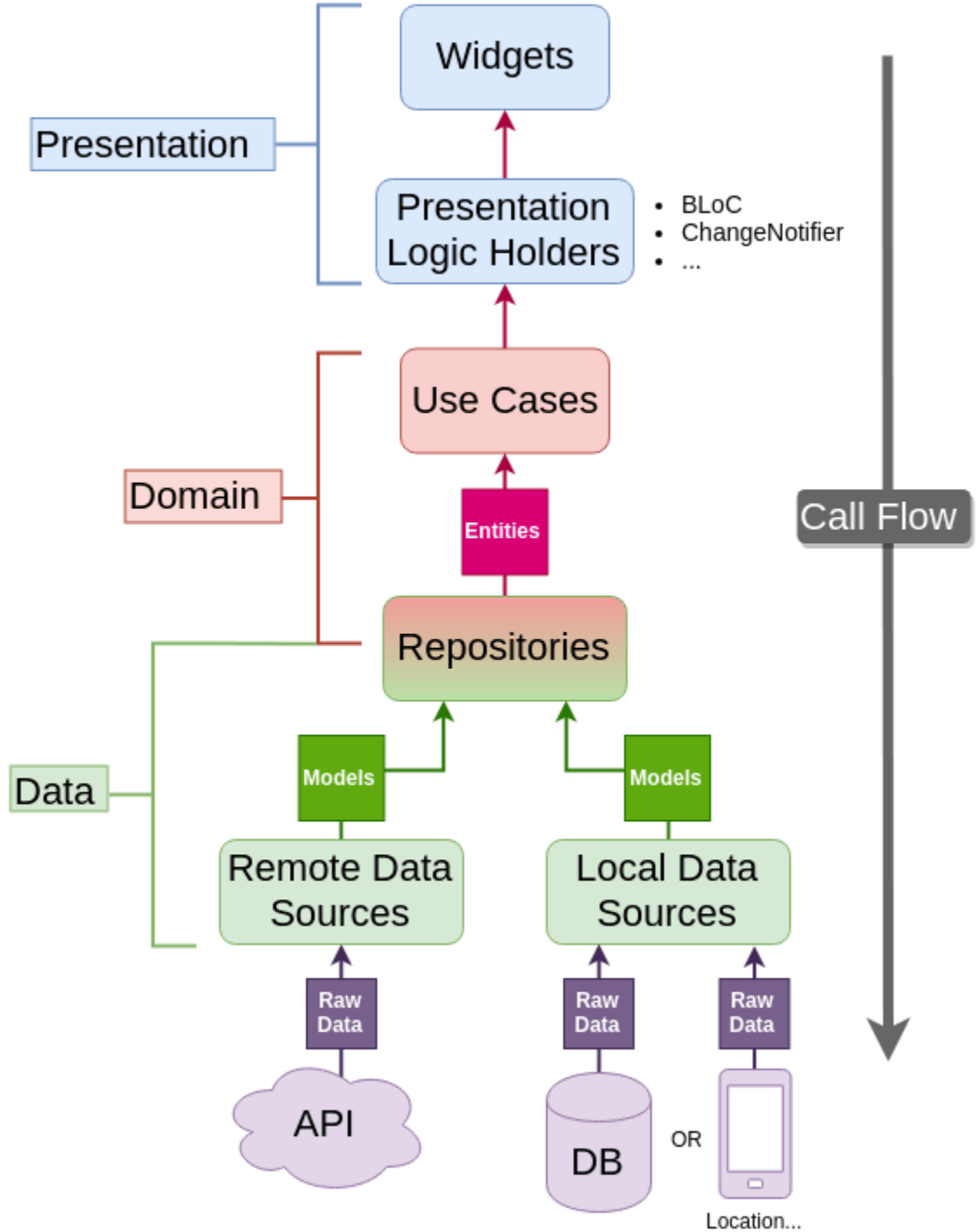

Widgets
Presentation
Presentation Logic Holders
BLoC
ChangeNotifier
...
Use Cases
Domain
Entities
Repositories
Call Flow
Data
Models
Models
Remote Data Sources
Local Data Sources
Raw Data
Raw Data
Raw Data
API
DB
OR
Location...

Remote Repository

```dart
class BatchRemoteRepository implements IBatchRepository {
  final BatchRemoteDataSource remoteDataSource;

  BatchRemoteRepository({required this.remoteDataSource});

  @override
  Future<Either<Failure, void>> createBatch(BatchEntity batch) async {
    try {
      remoteDataSource.createBatch(batch);
      return Right(null);
    } catch (e) {
      return Left(
        ApiFailure(
          message: e.toString(),
        ), // ApiFailure
      ); // Left
    }
  }
```

Remote Repository

```dart
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  }
```


Widgets
Presentation
Presentation Logic Holders
BLoC
ChangeNotifier
...
Use Cases
Domain
Entities
Repositories
Call Flow
Data
Models
Models
Remote Data Sources
Local Data Sources
Raw Data
Raw Data
Raw Data
API
DB
OR
Location...

Get_It : Api Service

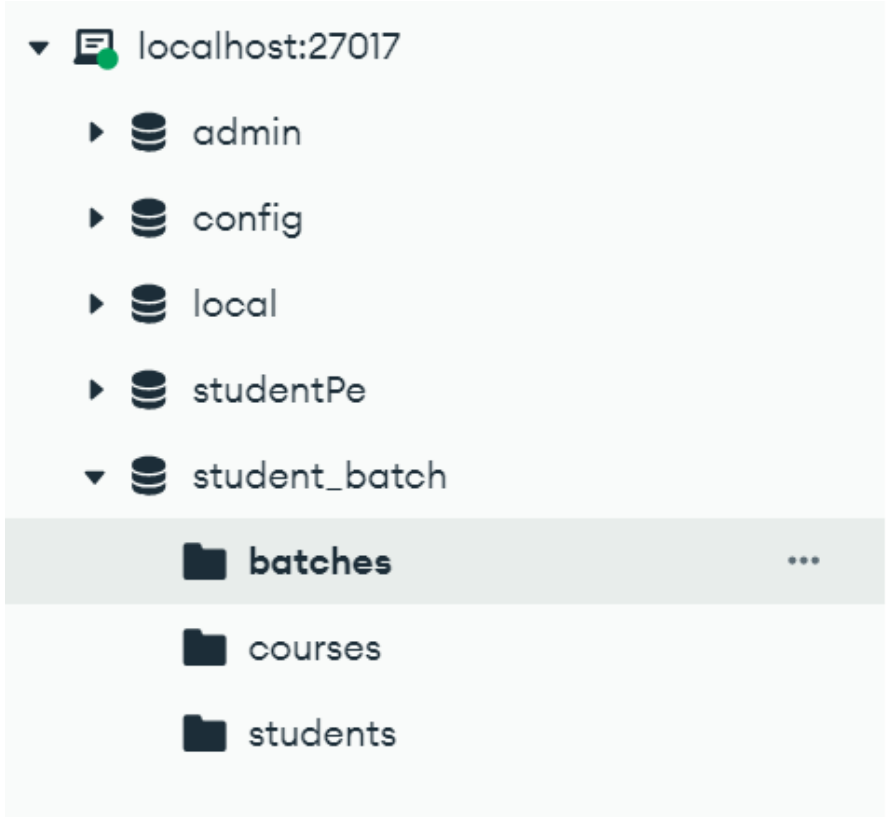
```dart
30  Future<void> initDependencies() async {
31    // First initialize hive service
32    await _initHiveService();
33    await _initApiService();
34
35    await _initBatchDependencies();
36    await _initCourseDependencies();
37    await _initHomeDependencies();
38    await _initRegisterDependencies();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

Get_It

```dart
_initBatchDependencies() async {
  // Local Data Source
  getIt.registerFactory<BatchLocalDataSource>(
      () => BatchLocalDataSource(hiveService: getIt<HiveService>()));

  // Remote Data Source
  getIt.registerLazySingleton<BatchRemoteDataSource>(
    () => BatchRemoteDataSource(
      dio: getIt<Dio>(),
    ),
  );

  // Batch Local Repository
  getIt.registerLazySingleton<BatchLocalRepository>(() => BatchLocalRepository(
      batchLocalDataSource: getIt<BatchLocalDataSource>()));

  // Batch Remote Repository
  getIt.registerLazySingleton(
    () => BatchRemoteRepository(
      remoteDataSource: getIt<BatchRemoteDataSource>(),
    ),
  );

  // Usecases
  getIt.registerLazySingleton<CreateBatchUseCase>(
    () => CreateBatchUseCase(batchRepository: getIt<BatchRemoteRepository>()),
  );
```

```dart
  getIt.registerLazySingleton<GetAllBatchUseCase>(
    () => GetAllBatchUseCase(batchRepository: getIt<BatchRemoteRepository>()),
  );

  getIt.registerLazySingleton<DeleteBatchUsecase>(
    () => DeleteBatchUsecase(batchRepository: getIt<BatchRemoteRepository>()),
  );

  // Bloc
  getIt.registerFactory<BatchBloc>(
    () => BatchBloc(
      createBatchUseCase: getIt<CreateBatchUseCase>(),
      getAllBatchUseCase: getIt<GetAllBatchUseCase>(),
      deleteBatchUsecase: getIt<DeleteBatchUsecase>(),
    ),
  );
}
```

Run the project and check mongodb for the data



Now try to insert course by yourself

Get all batches from API

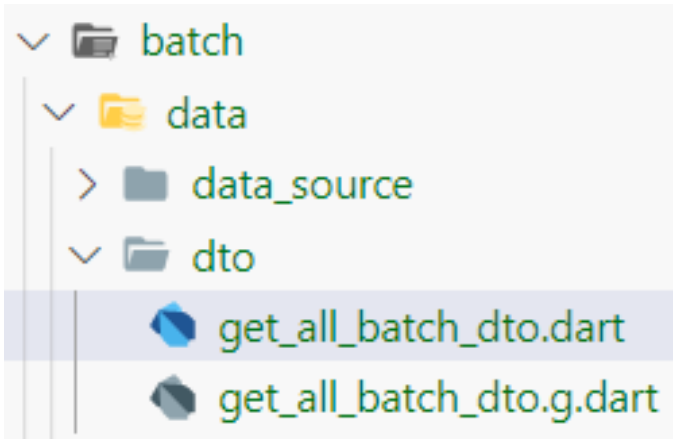
Type a query: { field: 'value' } or Generate query

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)




```dart
part 'get_all_batch_dto.g.dart';

@JsonSerializable()
class GetAllBatchDTO {
  final bool success;
  final int count;
  final List<BatchApiModel> data;

  GetAllBatchDTO({
    required this.success,
    required this.count,
    required this.data,
  });

  Map<String, dynamic> toJson() => _$GetAllBatchDTOToJson(this);

  factory GetAllBatchDTO.fromJson(Map<String, dynamic> json) =>
      _$GetAllBatchDTOFromJson(json);
}
```

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

Step 2 : Remote Data Source

```dart
@override
Future<List<BatchEntity>> getBatches() async {
  try {
    var response = await _dio.get(ApiEndpoints.getAllBatch);
    if (response.statusCode == 200) {
      GetAllBatchDTO batchAddDTO = GetAllBatchDTO.fromJson(response.data);
      return BatchApiModel.toEntityList(batchAddDTO.data);
    } else {
      throw Exception(response.statusMessage);
    }
  } on DioException catch (e) {
    throw Exception(e);
  } catch (e) {
    throw Exception(e);
  }
}
```

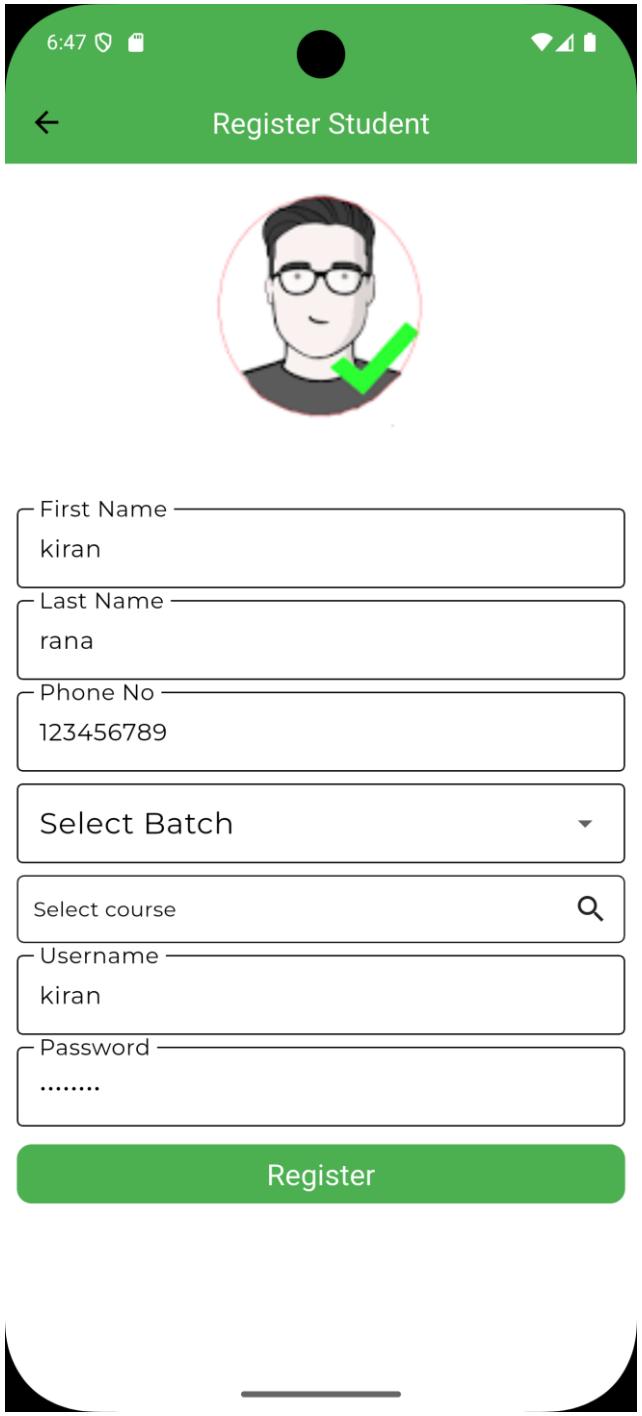
Step 3 : Batch Remote Repository

```dart
@override
Future<Either<Failure, List<BatchEntity>>> getBatches() async {
  try {
    final batches = await remoteDataSource.getBatches();
    return Right(batches);
  } catch (e) {
    return Left(
      ApiFailure(
        message: e.toString(),
      ), // ApiFailure
    ); // Left
  }
}
```

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



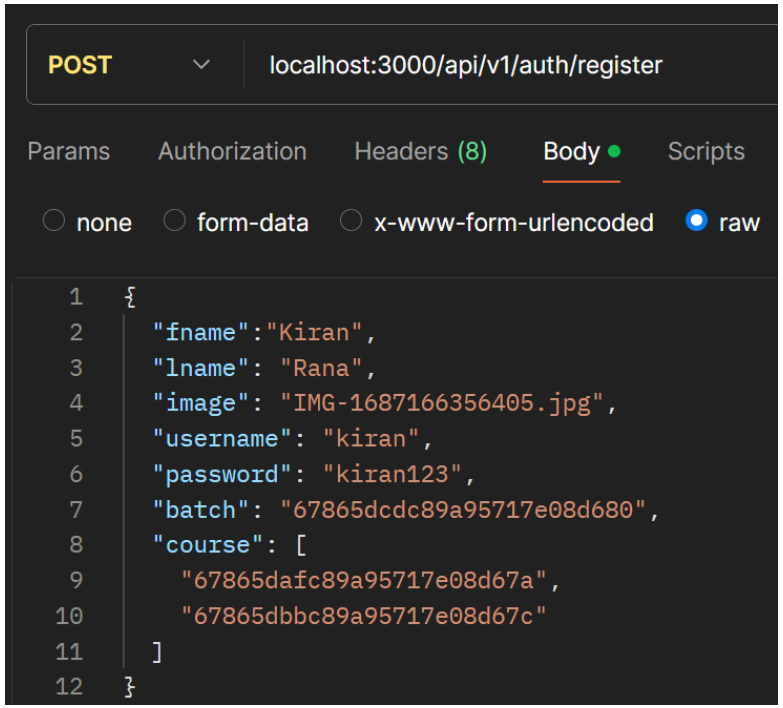
1. Auth API Model

```dart
part 'auth_api_model.g.dart';

@JsonSerializable()
class AuthApiModel extends Equatable {
  @JsonKey(name: '_id')
  final String? id;
  final String fname;
  final String lname;
  final String? image;
  final String phone;
  final BatchApiModel batch;
  final List<CourseApiModel> courses;
  final String username;
  final String? password;

  const AuthApiModel({
    this.id,
    required this.fname,
    required this.lname,
    required this.image,
    required this.phone,
    required this.batch,
    required this.courses,
    required this.username,
    required this.password,
  });
```

1. Auth API Mode

```dart
factory AuthApiModel.fromJson(Map<String, dynamic> json) =>
    _$AuthApiModelFromJson(json);

Map<String, dynamic> toJson() => _$AuthApiModelToJson(this);

// To Entity
AuthEntity toEntity() {
  return AuthEntity(
    userId: id,
    fName: fname,
    lName: lname,
    image: image,
    phone: phone,
    batch: batch.toEntity(),
    courses: courses.map((e) => e.toEntity()).toList(),
    username: username,
    password: password ?? '',
  );
}

// From Entity
factory AuthApiModel.fromEntity(AuthEntity entity) {
  return AuthApiModel(
    fname: entity.fName,
    lname: entity.lName,
    image: entity.image,
    phone: entity.phone,
    batch: BatchApiModel.fromEntity(entity.batch),
    courses: entity.courses.map((e) => CourseApiModel.fromEntity(e)).toList(),
    username: entity.username,
    password: entity.password,
  ); // AuthApiModel
}
```

2. Batch Remote Data Source




```dart
@override
Future<void> registerStudent(AuthEntity student) async {
  try {
    Response response = await _dio.post(
      ApiEndpoints.register,
      data: {
        "fname": student.fName,
        "lname": student.lName,
        "phone": student.phone,
        "image": student.image,
        "username": student.username,
        "password": student.password,
        "batch": student.batch.batchId,
        // "course": ["6489a5908dbc6d39719ec19c", "6489a5968dbc6d39719ec19e"]
        "course": student.courses.map((e) => e.courseId).toList(),
      },
    );

    if (response.statusCode == 200) {
      return;
    } else {
      throw Exception(response.statusMessage);
    }
  } on DioException catch (e) {
    throw Exception(e);
  } catch (e) {
    throw Exception(e);
  }
}
```

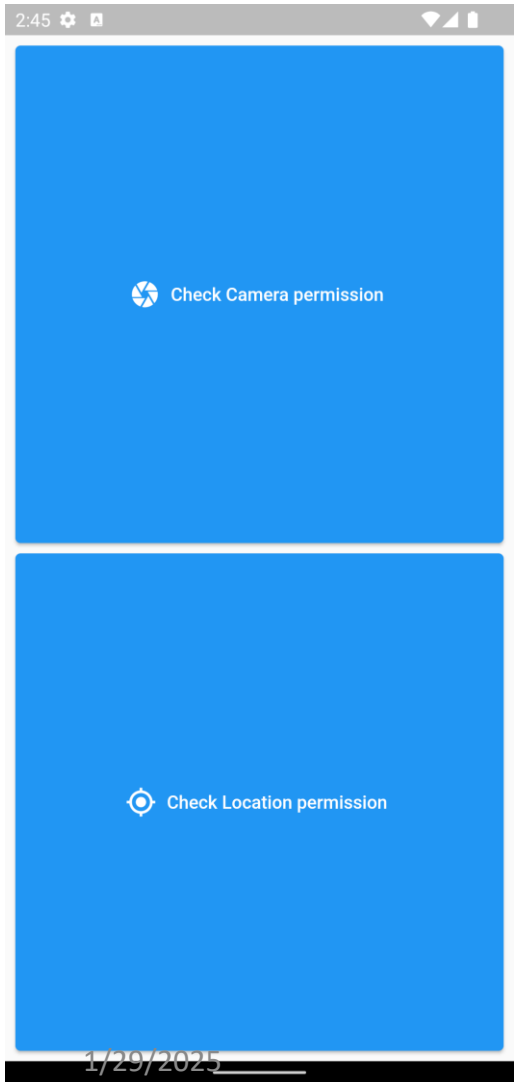
3. Batch Remote Repository

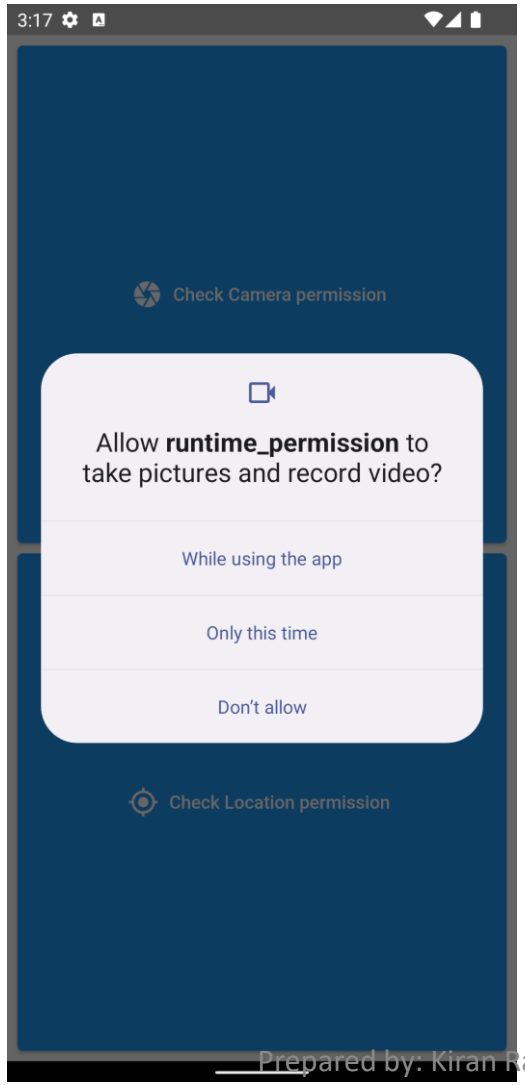
```dart
class AuthRemoteRepository implements IAuthRepository {
  final AuthRemoteDataSource _authRemoteDataSource;

  AuthRemoteRepository(this._authRemoteDataSource);

  @override
  Future<Either<Failure, void>> registerStudent(AuthEntity student) async {
    try {
      await _authRemoteDataSource.registerStudent(student);
      return Right(null);
    } catch (e) {
      return Left(ApiFailure(message: e.toString()));
    }
  }

  @override
  Future<Either<Failure, AuthEntity>> getCurrentUser() {
    // TODO: implement getCurrentUser
    throw UnimplementedError();
  }

  @override
  Future<Either<Failure, String>> loginStudent(
      String username, String password) {
    // TODO: implement loginStudent
    throw UnimplementedError();
  }

  @override
  Future<Either<Failure, String>> uploadProfilePicture(File file) {
    // TODO: implement uploadProfilePicture
    throw UnimplementedError();
  }
}
```

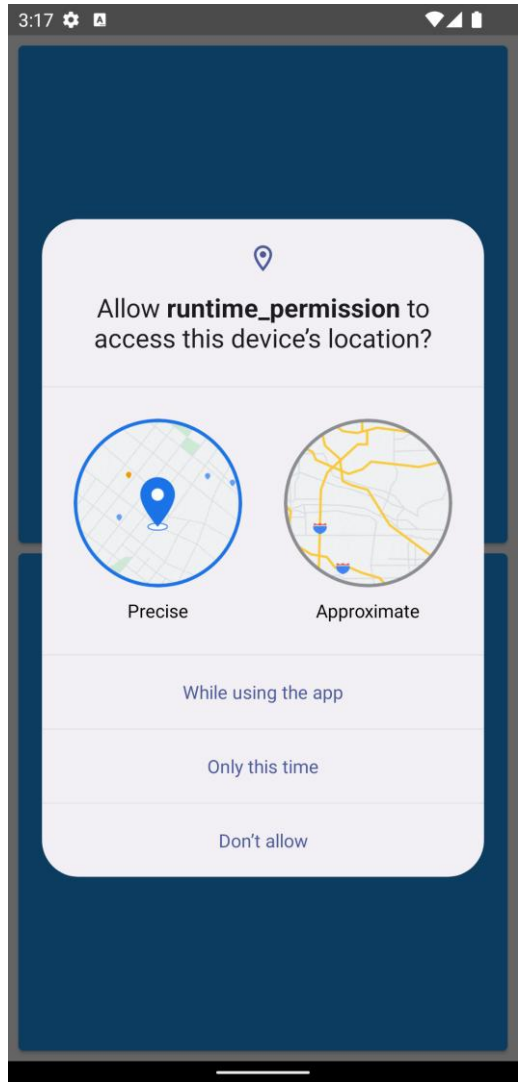
4. GetIt

```dart
_initRegisterDependencies() {
// ============================ Data Source ============================

  getIt.registerLazySingleton<AuthLocalDataSource>(
    () => AuthLocalDataSource(getIt<HiveService>()),
  );

  getIt.registerLazySingleton<AuthRemoteDataSource>(
    () => AuthRemoteDataSource(getIt<Dio>()),
  );

  // ============================ Repository ============================

  getIt.registerLazySingleton(
    () => AuthLocalRepository(getIt<AuthLocalDataSource>()),
  );
  getIt.registerLazySingleton<AuthRemoteRepository>(
    () => AuthRemoteRepository(getIt<AuthRemoteDataSource>()),
  );

  // register use usecase
  getIt.registerLazySingleton<RegisterUseCase>(
    () => RegisterUseCase(
      getIt<AuthRemoteRepository>(),
    ),
  );

  getIt.registerFactory<RegisterBloc>(
    () => RegisterBloc(
      batchBloc: getIt<BatchBloc>(),
      courseBloc: getIt<CourseBloc>(),
      registerUseCase: getIt(),
    ),
  );
}
```

Runtime permission



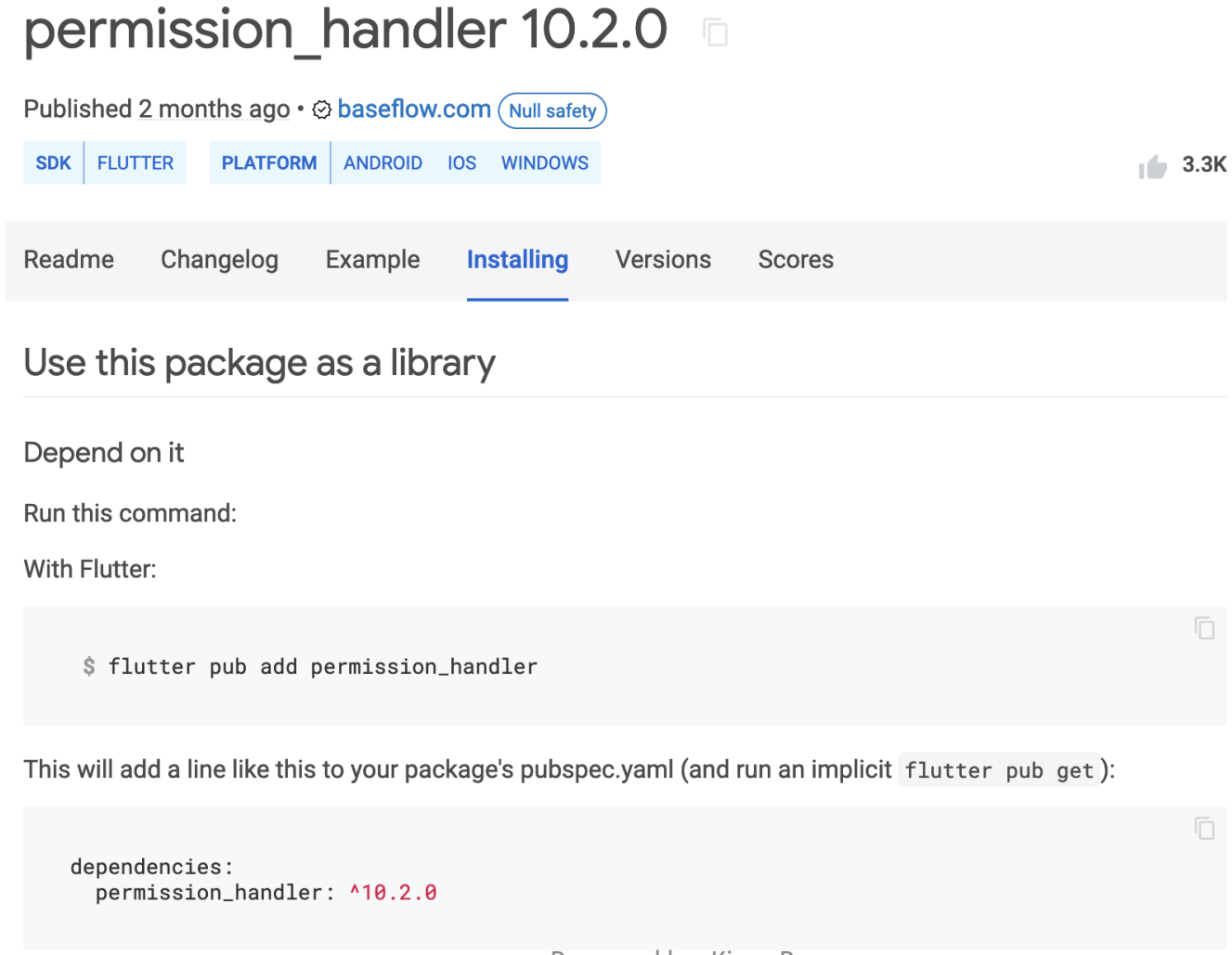




Permissions on Android

- 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 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
```

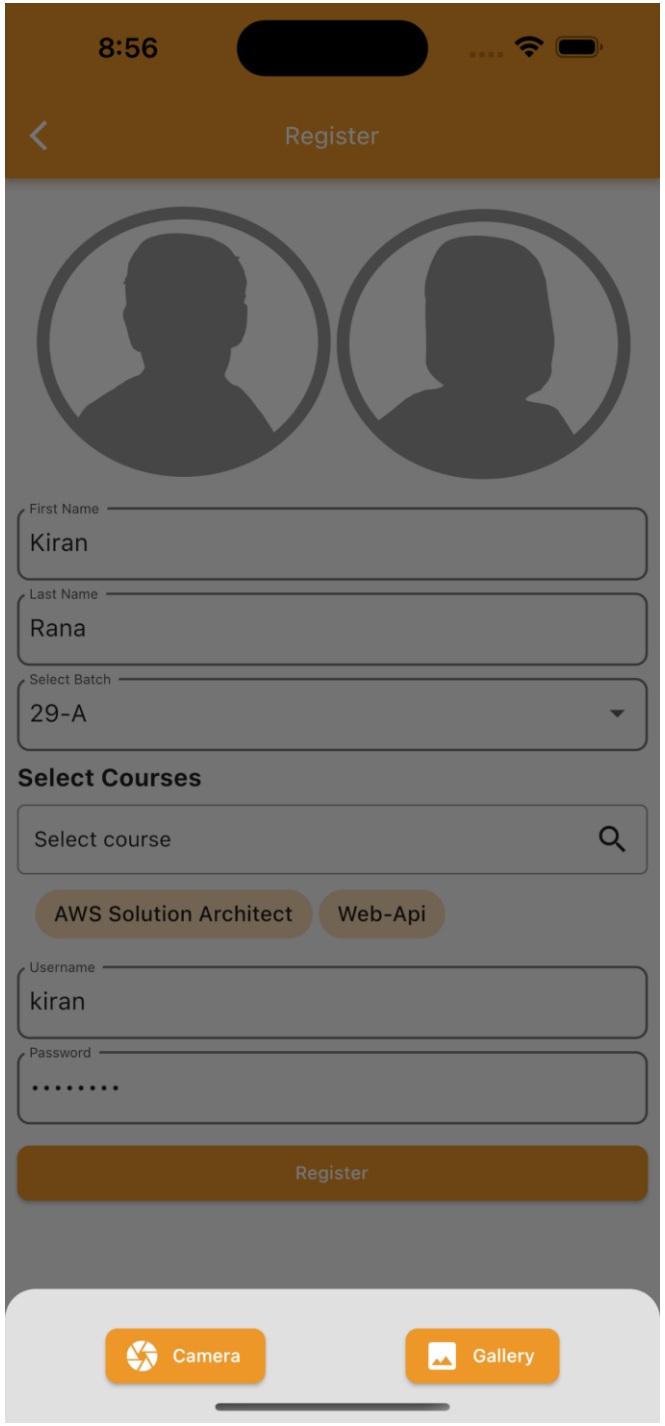
This will add a line like this to your package's pubspec.yaml (and run an implicit `flutter pub get`):

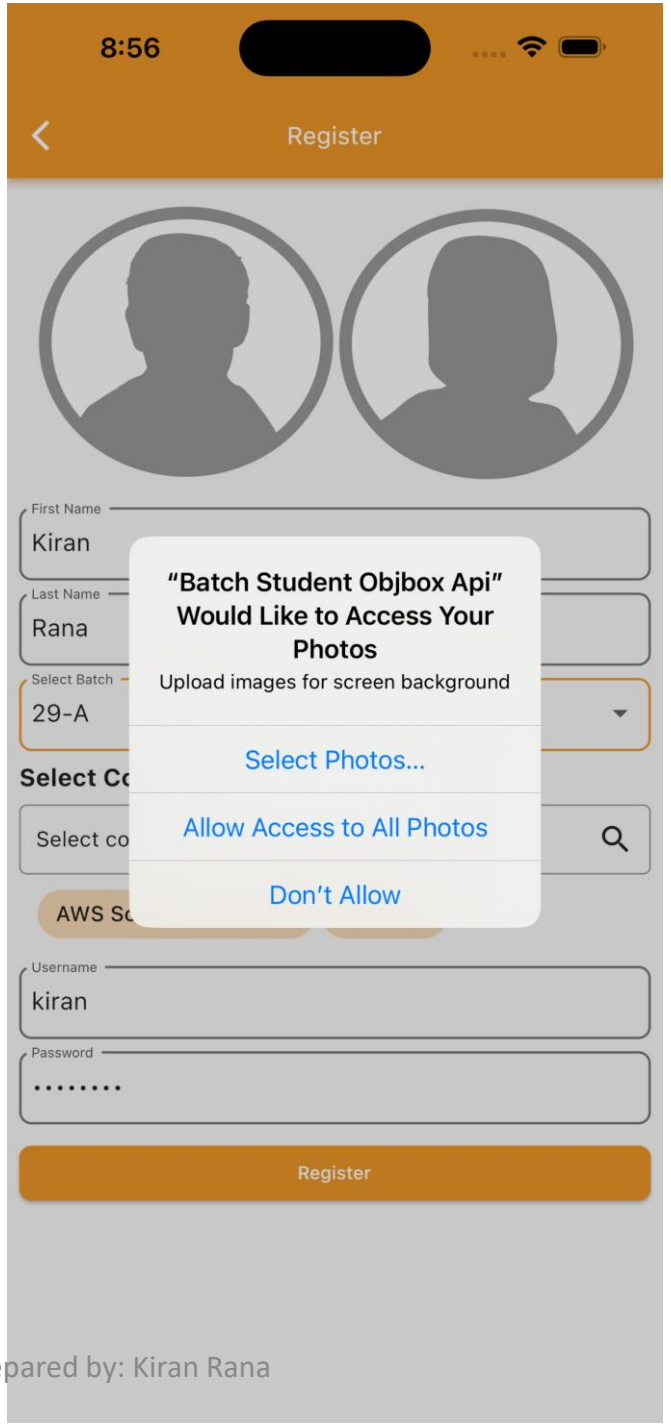
```
dependencies:
  permission_handler: ^10.2.0
```

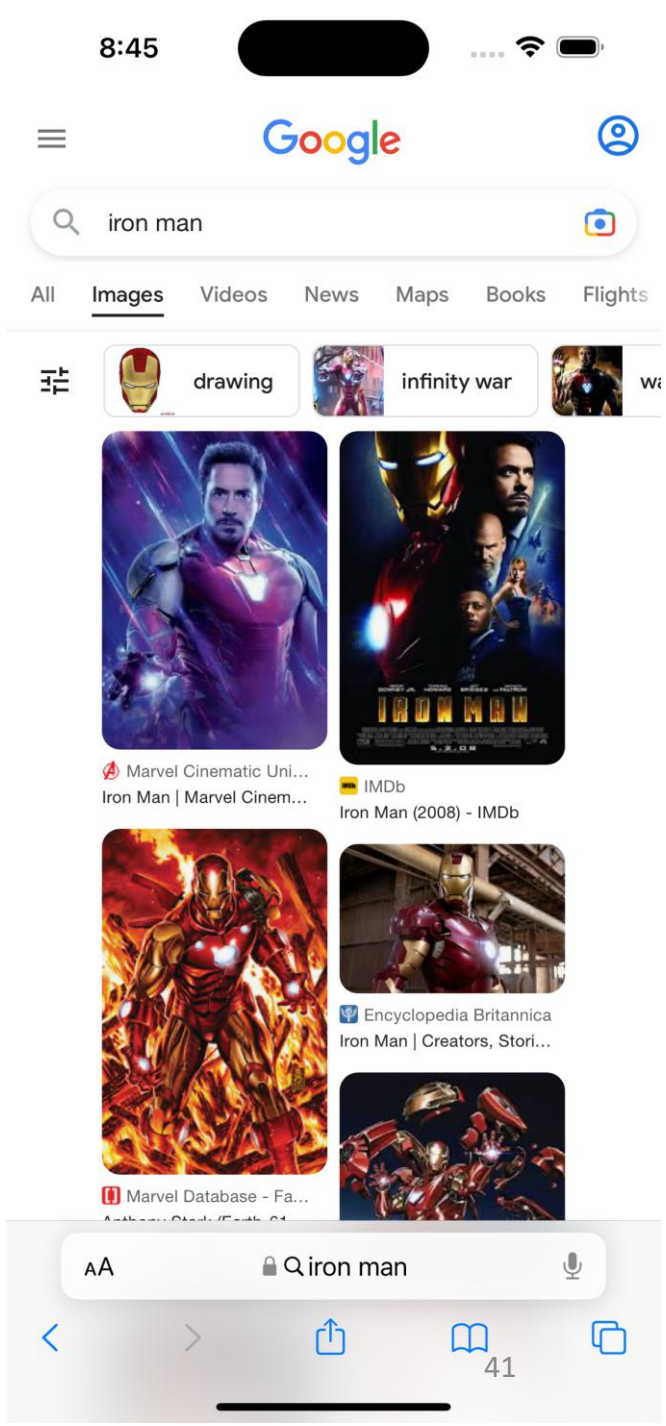
Step 2 - Add permissions to AndroidManifest.xml file

Goto Your_Flutter_Project -> android -> app -> src -> main -> AndroidManifest.xml

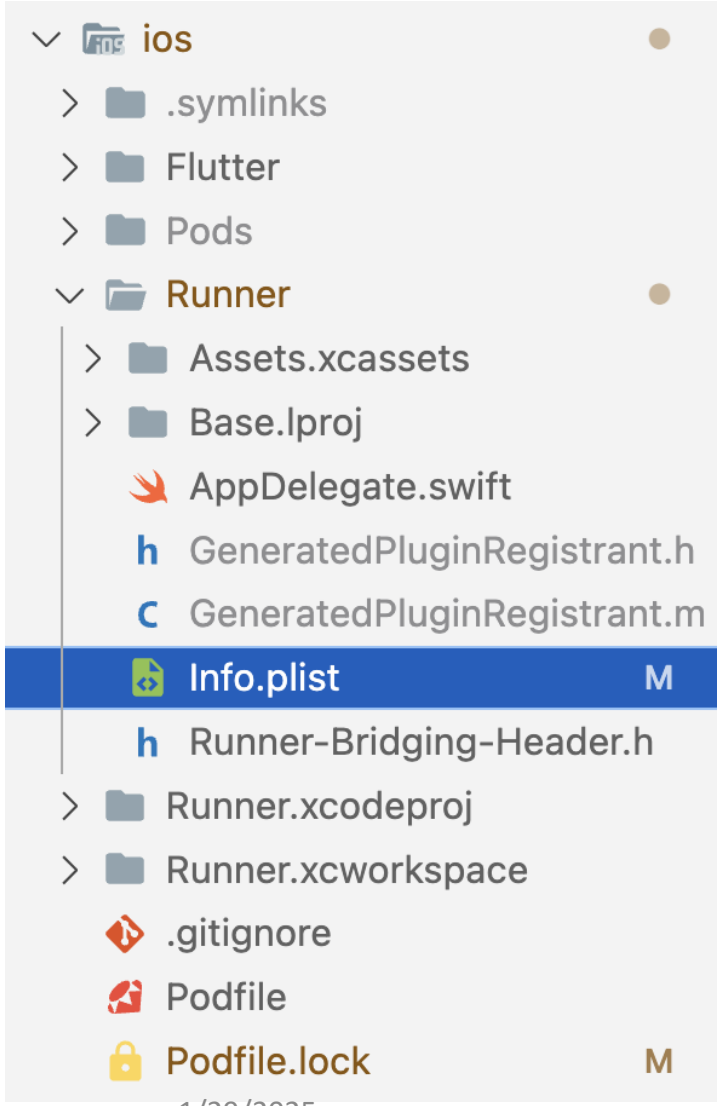
For iOS


8:56
Register
First Name
Kiran
Last Name
Rana
Select Batch
29-A
Select Courses
Select course
AWS Solution Architect
Web-Api
Username
kiran
Password
Register
Camera
Gallery


8:56
Register
First Name
Kiran
Last Name
Rana
Select Batch
29-A
"Batch Student Objbox Api" Would Like to Access Your Photos
Upload images for screen background
Select Photos...
Allow Access to All Photos
Don't Allow
Username
kiran
Password
Register


8:45
Google
iron man
All
Images
Videos
News
Maps
Books
Flights
drawing
infinity war
Marvel Cinematic Uni...
Iron Man | Marvel Cinem...
IMDb
Iron Man (2008) - IMDb
Encyclopedia Britannica
Iron Man | Creators, Stori...
Marvel Database - Fa...
AA
iron man

For iOS



Info.plist M

ios > Runner > Info.plist

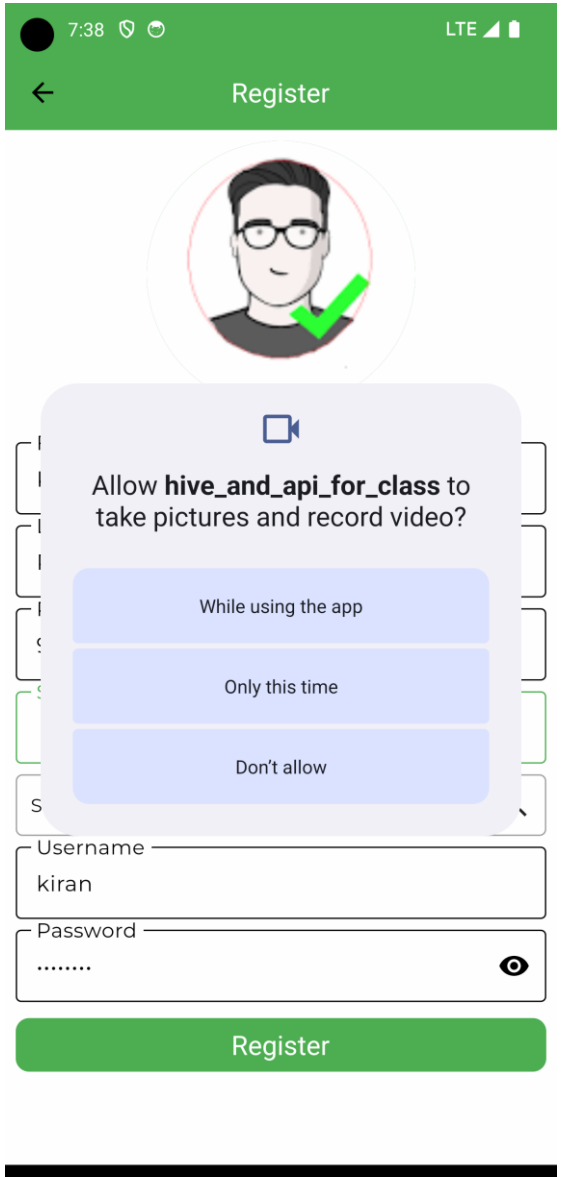
```
40          <string>UIInterfaceOrientationPortraitUpsideDown</string>
41          <string>UIInterfaceOrientationLandscapeLeft</string>
42          <string>UIInterfaceOrientationLandscapeRight</string>
43      </array>
44      <key>UIViewControllerBasedStatusBarAppearance</key>
45      <false/>
46      <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>
```

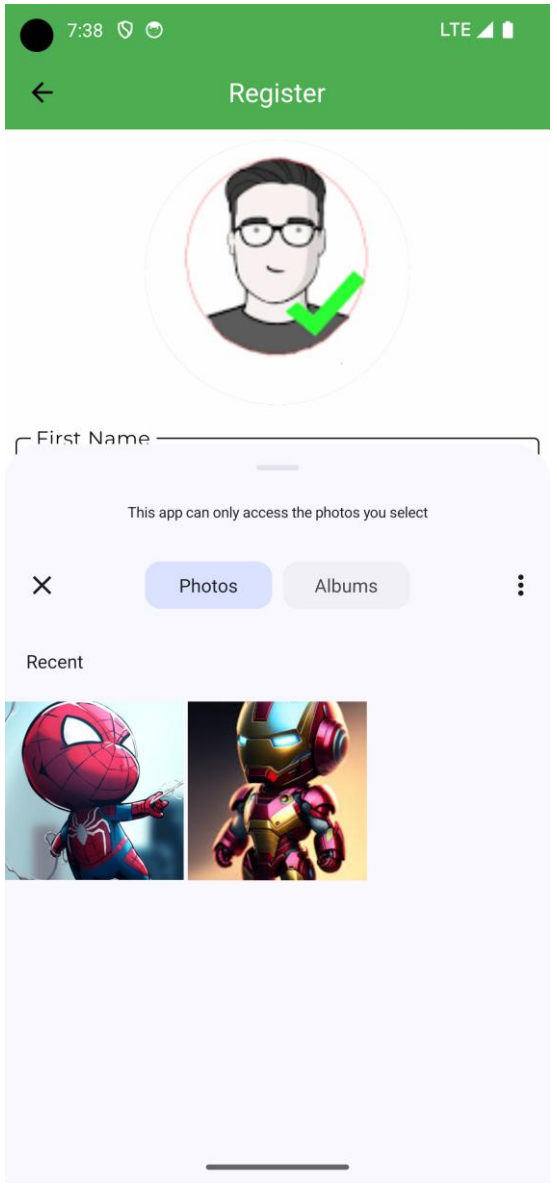
Copy paste this text in Info.plist

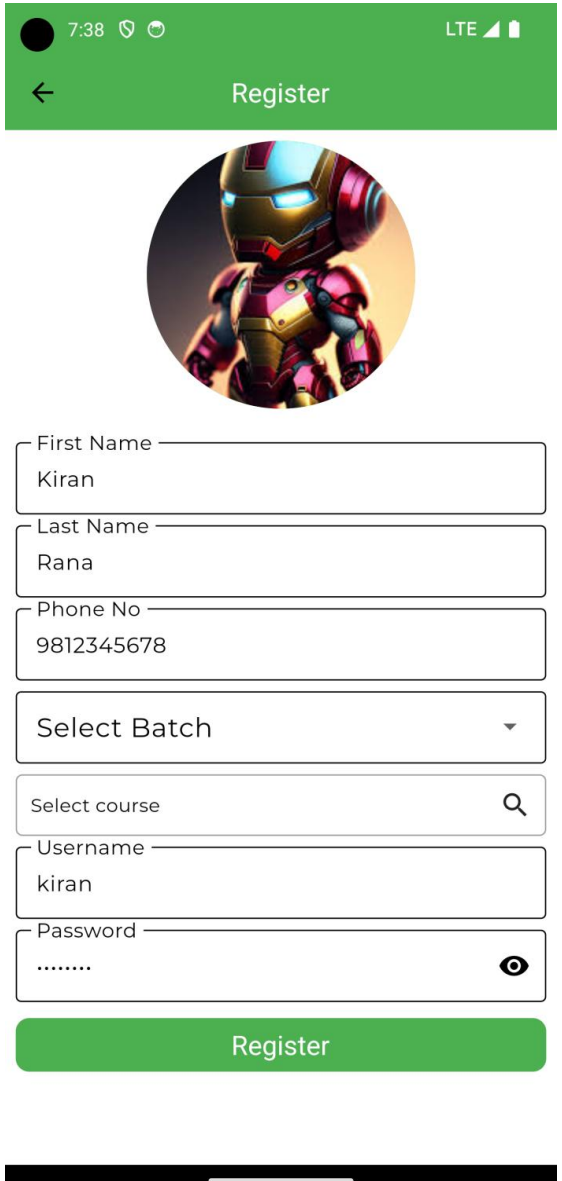
```
<key>NSPhotoLibraryUsageDescription</key>
<string>Upload images for screen background</string>
<key>NSCameraUsageDescription</key>
<string>Upload image from camera for screen background</string>
<key>NSMicrophoneUsageDescription</key>
<string>Post videos to profile</string>
```

Run time permission and Camera

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







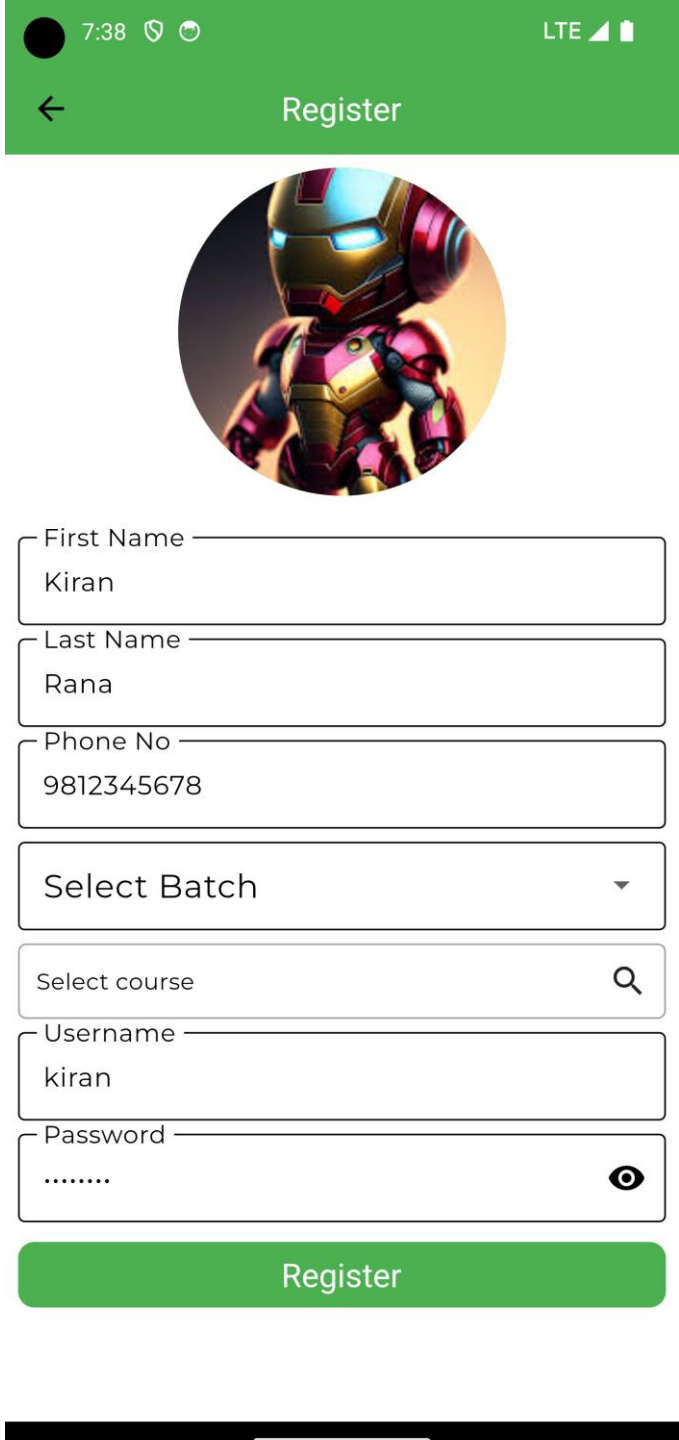
Step 1 : Register page

```
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  }
```

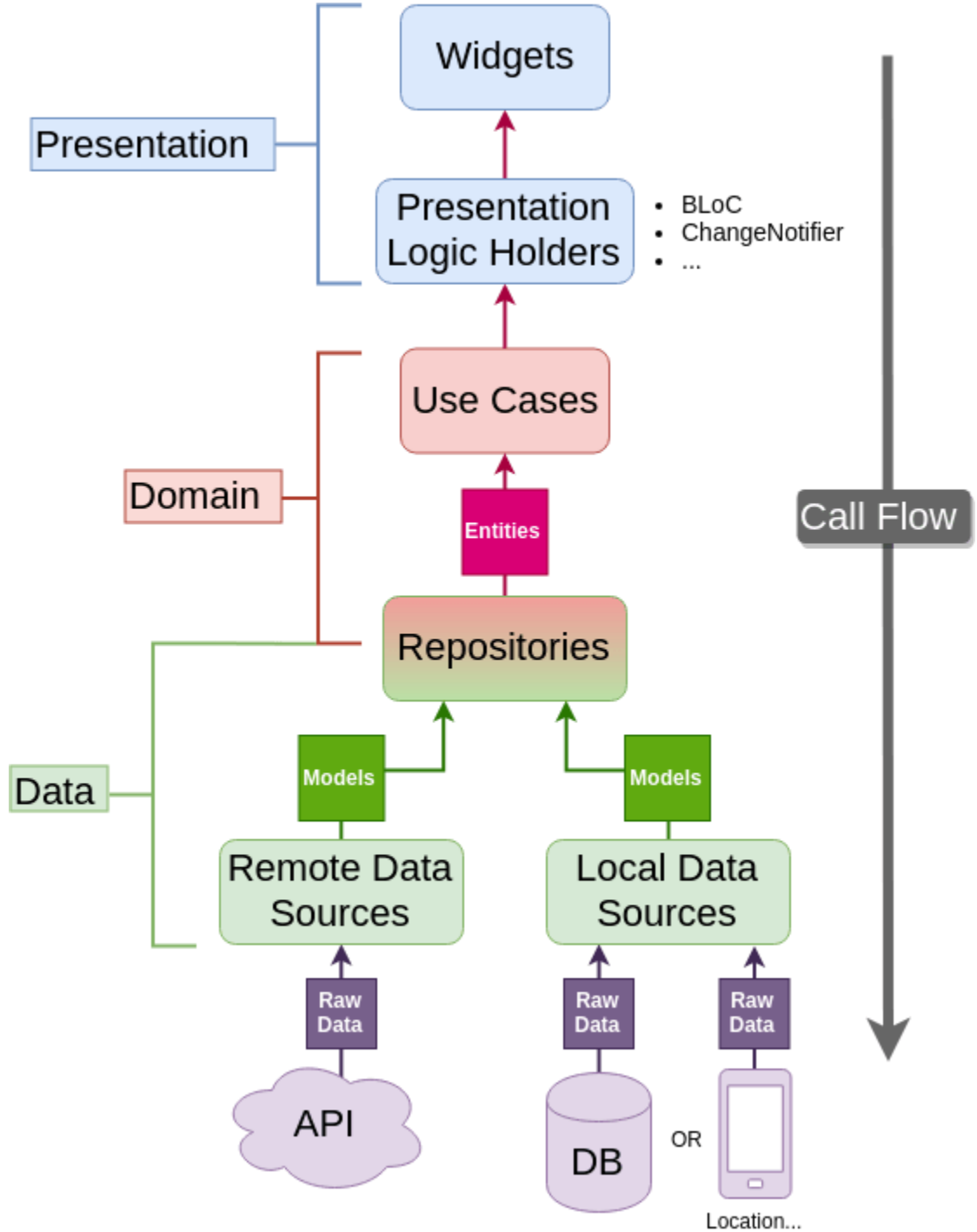
Step 2 : Load image in imageview

```
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
```



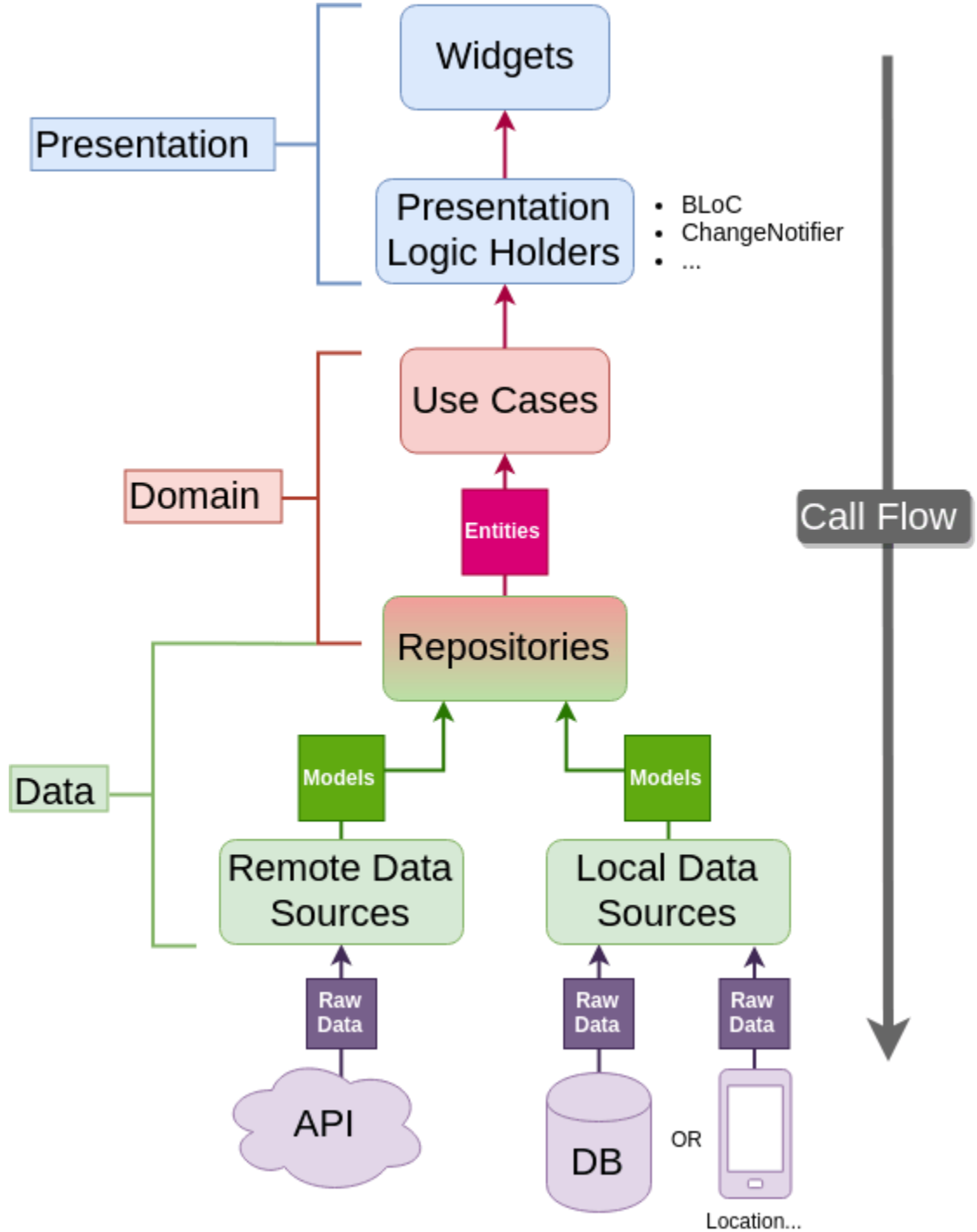
Step 3 : On Camera click

```
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
```

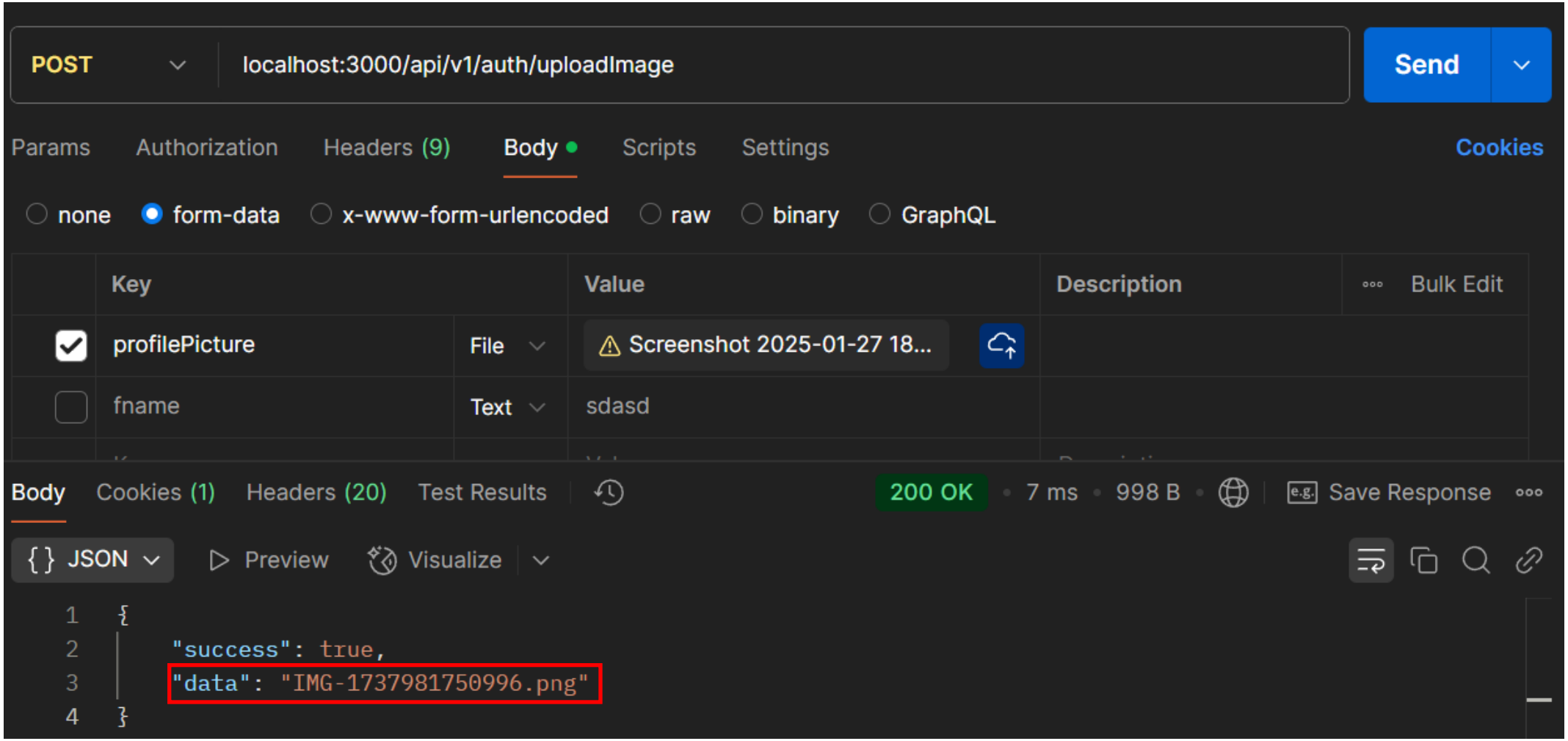

Widgets
Presentation
Presentation Logic Holders
BLoC
ChangeNotifier
...
Use Cases
Domain
Entities
Repositories
Call Flow
Data
Models
Models
Remote Data Sources
Local Data Sources
Raw Data
Raw Data
Raw Data
API
DB
OR
Location...

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 }
```


Widgets
Presentation
Presentation Logic Holders
BLoC
ChangeNotifier
...
Use Cases
Domain
Entities
Repositories
Call Flow
Data
Models
Models
Remote Data Sources
Local Data Sources
Raw Data
Raw Data
Raw Data
API
DB
OR
Location...

Step 5 : Auth Remote Data Source

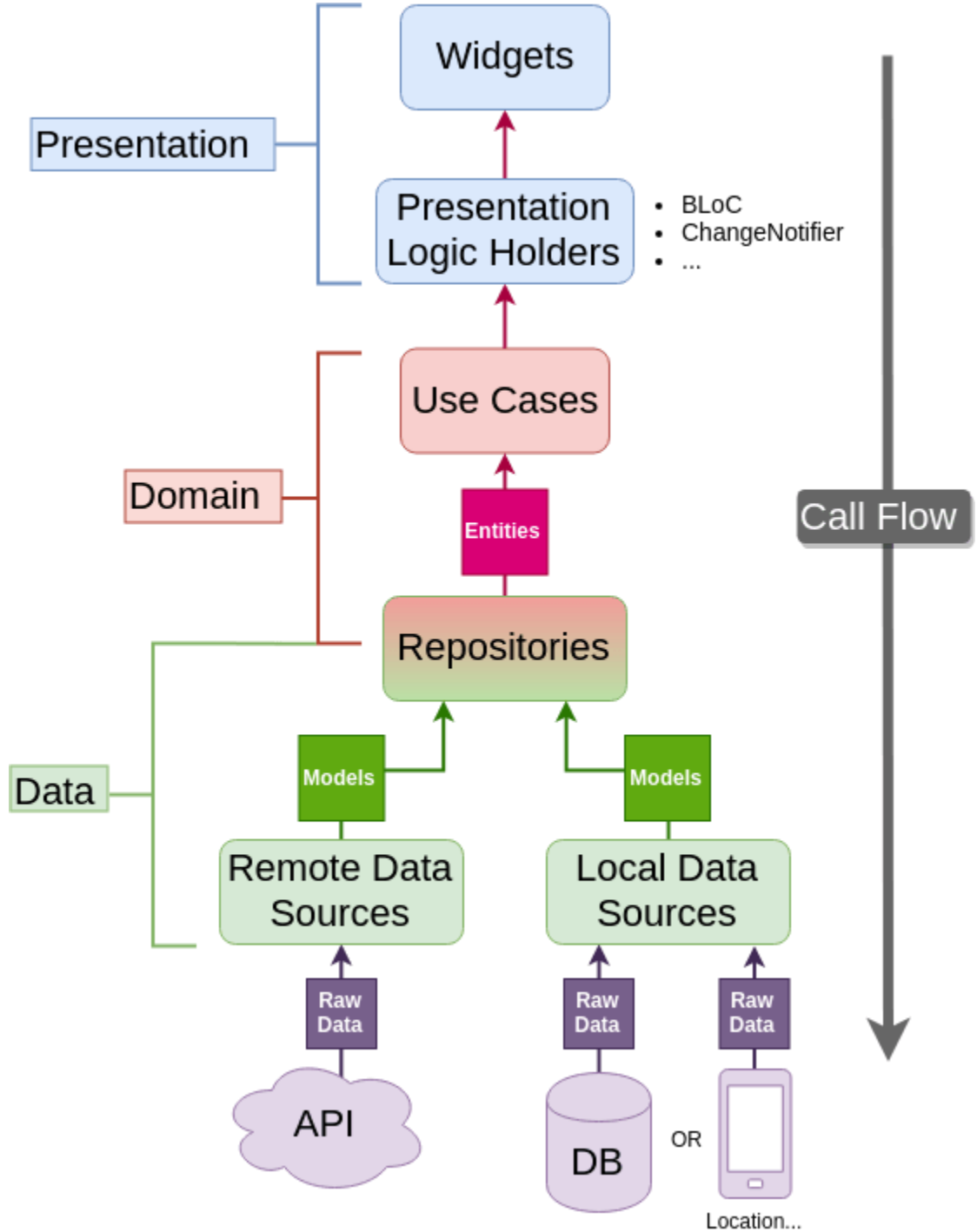


Step 5 : Auth Remote Data Source

```dart
@override
Future<String> uploadProfilePicture(File file) async {
  try {
    String fileName = file.path.split('/').last;
    FormData formData = FormData.fromMap(
      {
        'profilePicture': await MultipartFile.fromFile(
          file.path,
          filename: fileName,
        ),
      },
    );

    Response response = await _dio.post(
      ApiEndpoints.uploadImage,
      data: formData,
    );

    if (response.statusCode == 200) {
      // Extract the image name from the response
      return response.data['data'];
    } else {
      throw Exception(response.statusMessage);
    }
  } on DioException catch (e) {
    throw Exception(e);
  } catch (e) {
    throw Exception(e);
  }
}
}
```

Sep 6 : Auth Remote Repository

```dart
  @override
  Future<Either<Failure, String>> uploadProfilePicture(File file) async {
    try {
      final imageName = await _authRemoteDataSource.uploadProfilePicture(file);
      return Right(imageName);
    } catch (e) {
      return Left(ApiFailure(message: e.toString()));
    }
  }
}
```


Widgets
Presentation
Presentation Logic Holders
BLoC
ChangeNotifier
...
Use Cases
Domain
Entities
Repositories
Call Flow
Data
Models
Models
Remote Data Sources
Local Data Sources
Raw Data
Raw Data
Raw Data
API
DB
OR
Location...

Step 6 : Use Case

```dart
class UploadImageParams {
  final File file;

  const UploadImageParams({
    required this.file,
  });
}

class UploadImageUsecase
    implements UsecaseWithParams<String, UploadImageParams> {
  final IAuthRepository _repository;

  UploadImageUsecase(this._repository);

  @override
  Future<Either<Failure, String>> call(UploadImageParams params) {
    return _repository.uploadProfilePicture(params.file);
  }
}
```

Step 7 : Auth View Model (Auth State)

```dart
part of 'register_bloc.dart';

class RegisterState extends Equatable {
  final bool isLoading;
  final bool isSuccess;
  final String? imageName;

  const RegisterState({
    required this.isLoading,
    required this.isSuccess,
    this.imageName,
  });

  const RegisterState.initial()
      : isLoading = false,
        isSuccess = false,
        imageName = null;

  RegisterState copyWith({
    bool? isLoading,
    bool? isSuccess,
    String? imageName,
  }) {
    return RegisterState(
      isLoading: isLoading ?? this.isLoading,
      isSuccess: isSuccess ?? this.isSuccess,
      imageName: imageName ?? this.imageName,
    );
  }

  @override
  List<Object?> get props => [isLoading, isSuccess, imageName];
}
```

Step 7 : Auth Event

```dart
part of 'register_bloc.dart';

sealed class RegisterEvent extends Equatable {
  const RegisterEvent();

  @override
  List<Object> get props => [];
}

class LoadCoursesAndBatches extends RegisterEvent {}

class LoadImage extends RegisterEvent {
  final File file;

  const LoadImage({
    required this.file,
  });
}

class RegisterStudent extends RegisterEvent {…
```

Step 7 : Auth Bloc

```dart
class RegisterBloc extends Bloc<RegisterEvent, RegisterState> {
  final BatchBloc _batchBloc;
  final CourseBloc _courseBloc;
  final RegisterUseCase _registerUseCase;
  final UploadImageUsecase _uploadImageUsecase;

  RegisterBloc({
    required BatchBloc batchBloc,
    required CourseBloc courseBloc,
    required RegisterUseCase registerUseCase,
    required UploadImageUsecase uploadImageUsecase,
  })  : _batchBloc = batchBloc,
        _courseBloc = courseBloc,
        _registerUseCase = registerUseCase,
        _uploadImageUsecase = uploadImageUsecase,
        super(RegisterState.initial()) {
    on<LoadCoursesAndBatches>(_onLoadCoursesAndBatches);
    on<RegisterStudent>(_onRegisterEvent);
    on<LoadImage>(_onLoadImage);

    add(LoadCoursesAndBatches());
  }
```

Step 7 :Auth Bloc

```dart
  void _onLoadImage(
    LoadImage event,
    Emitter<RegisterState> emit,
  ) async {
    emit(state.copyWith(isLoading: true));
    final result = await _uploadImageUsecase.call(
      UploadImageParams(
        file: event.file,
      ),
    );

    result.fold(
      (l) => emit(state.copyWith(isLoading: false, isSuccess: false)),
      (r) {
        emit(state.copyWith(isLoading: false, isSuccess: true, imageName: r));
      },
    );
  }
}
```

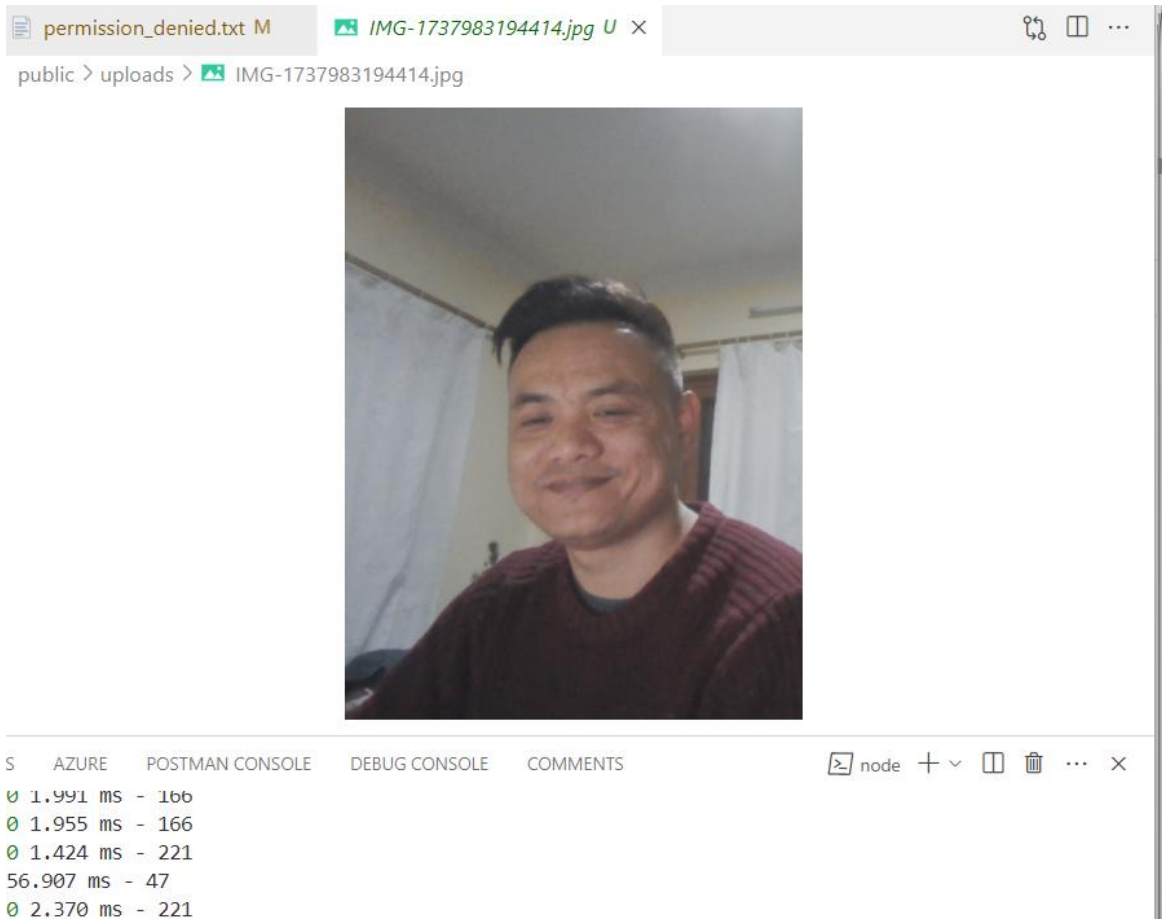
Step 8 : Register UI

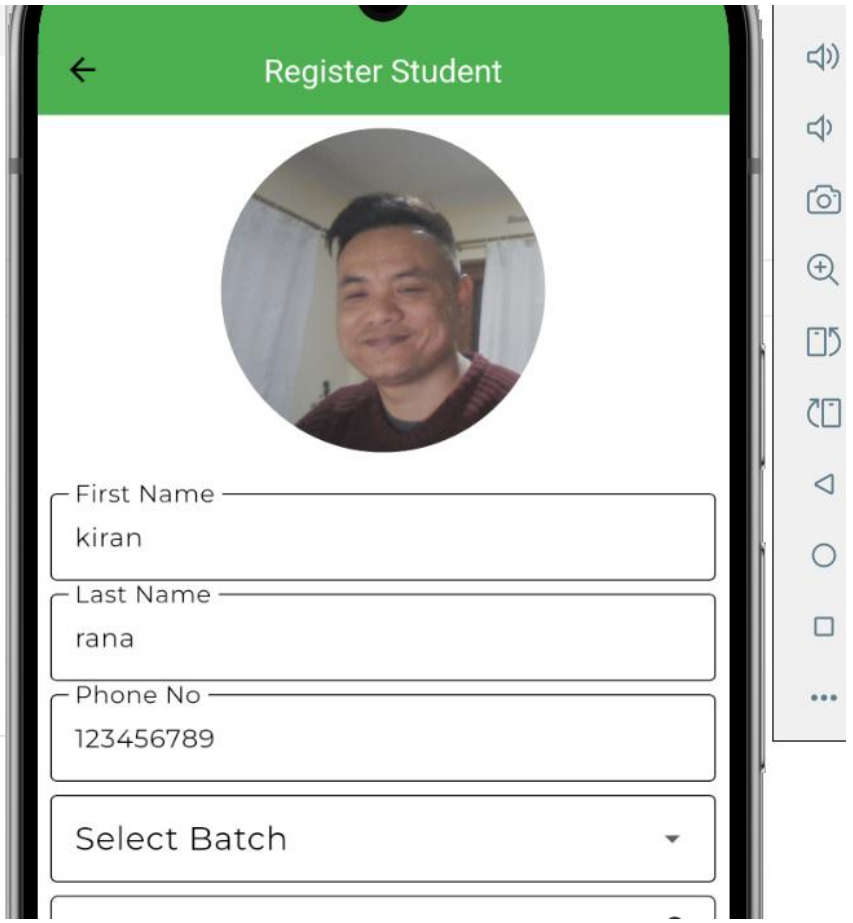
```dart
File? _img;
Future _browseImage(ImageSource imageSource) async {
  try {
    final image = await ImagePicker().pickImage(source: imageSource);
    if (image != null) {
      setState(() {
        _img = File(image.path);
        // Send image to server
        context.read<RegisterBloc>().add(
              LoadImage(file: _img!),
            );
      });
    } else {
      return;
    }
  } catch (e) {
    debugPrint(e.toString());
  }
}
```

Step 9 : Get It

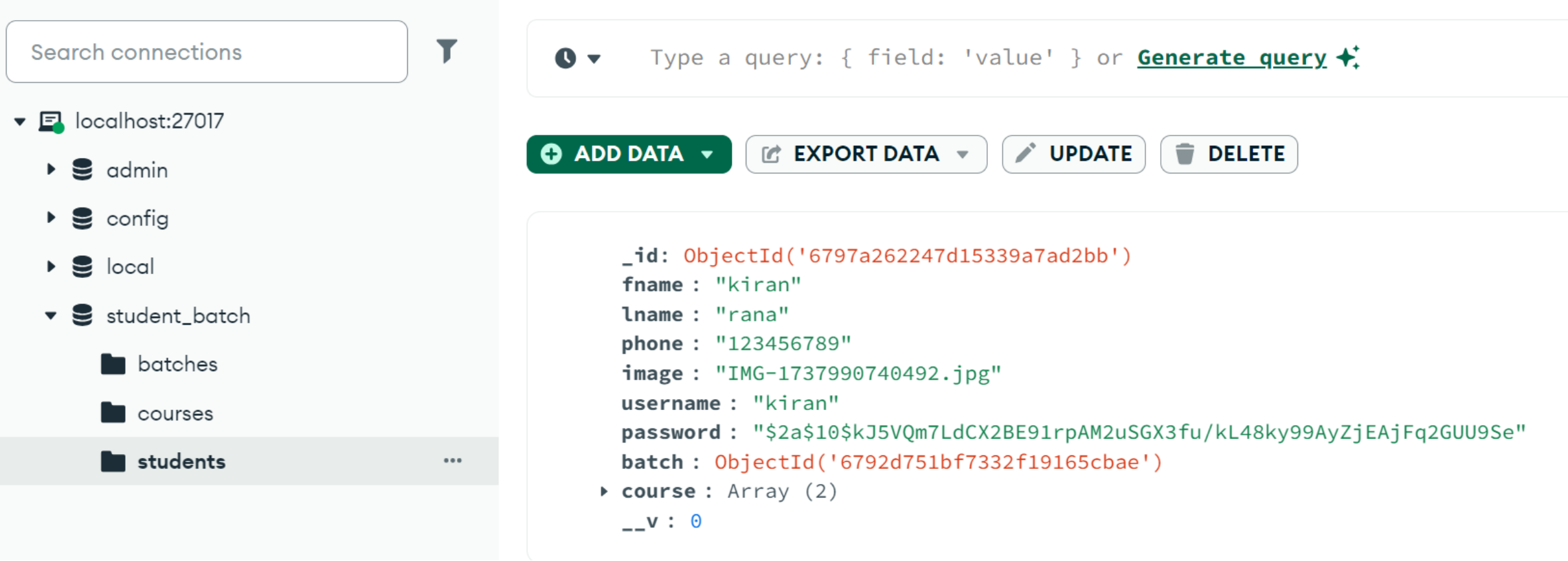
```dart
// ============================ Usecases ============================
getIt.registerLazySingleton<RegisterUseCase>(
  () => RegisterUseCase(
    getIt<AuthRemoteRepository>(),
  ),
);

getIt.registerLazySingleton<UploadImageUsecase>(
  () => UploadImageUsecase(
    getIt<AuthRemoteRepository>(),
  ),
);

getIt.registerFactory<RegisterBloc>(
  () => RegisterBloc(
    batchBloc: getIt<BatchBloc>(),
    courseBloc: getIt<CourseBloc>(),
    registerUseCase: getIt(),
    uploadImageUsecase: getIt(),
  ),
);
}
```

Now run the program





Now insert the image name in database



Step 1 : Register user usecase

```dart
class RegisterUserParams extends Equatable {
  final String fname;
  final String lname;
  final String phone;
  final BatchEntity batch;
  final List<CourseEntity> courses;
  final String username;
  final String password;
  final String? image;

  const RegisterUserParams({
    required this.fname,
    required this.lname,
    required this.phone,
    required this.batch,
    required this.courses,
    required this.username,
    required this.password,
    this.image,
  });
```

1. Register Use Case

```dart
class RegisterUseCase implements UsecaseWithParams<void, RegisterUserParams> {
  final IAuthRepository repository;

  RegisterUseCase(this.repository);

  @override
  Future<Either<Failure, void>> call(RegisterUserParams params) {
    final authEntity = AuthEntity(
      fName: params.fname,
      lName: params.lname,
      phone: params.phone,
      batch: params.batch,
      courses: params.courses,
      username: params.username,
      password: params.password,
      image: params.image,
    );
    return repository.registerStudent(authEntity);
  }
}
```

2. Register Event

```dart
class RegisterStudent extends RegisterEvent {
  final BuildContext context;
  final String fName;
  final String lName;
  final String phone;
  final BatchEntity batch;
  final List<CourseEntity> courses;
  final String username;
  final String password;
  final String? image;

  const RegisterStudent({
    required this.context,
    required this.fName,
    required this.lName,
    required this.phone,
    required this.batch,
    required this.courses,
    required this.username,
    required this.password,
    this.image,
  });
}
```

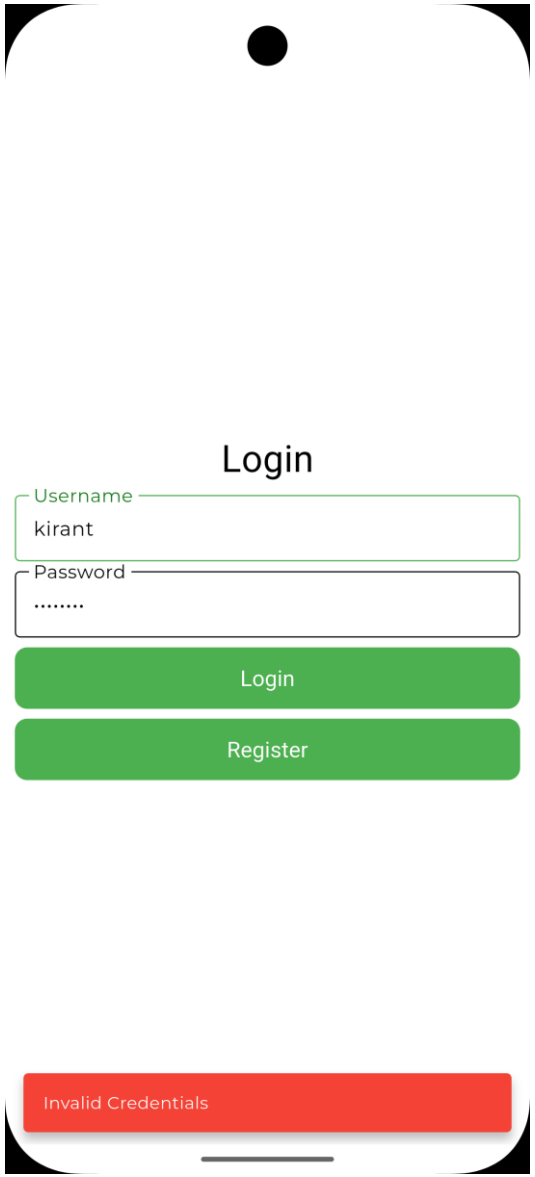
3. Register Bloc

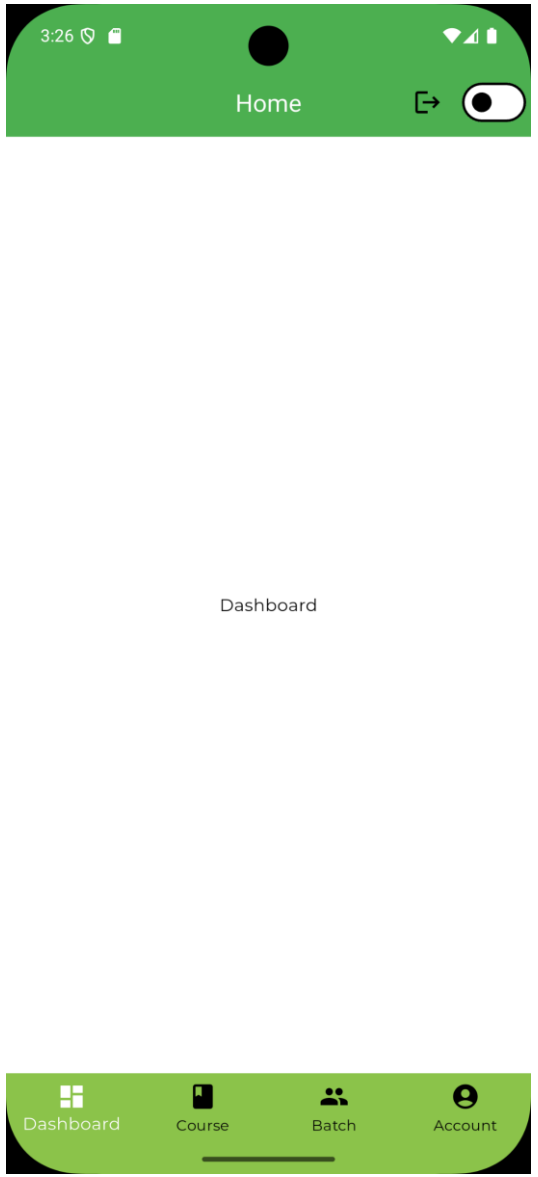
```dart
void _onRegisterEvent(
  RegisterStudent event,
  Emitter<RegisterState> emit,
) async {
  emit(state.copyWith(isLoading: true));
  final result = await _registerUseCase.call(RegisterUserParams(
    fname: event.fName,
    lname: event.lName,
    phone: event.phone,
    batch: event.batch,
    courses: _courseBloc.state.courses,
    username: event.username,
    password: event.password,
    image: state.imageName,
  ));

  result.fold(
    (l) => emit(state.copyWith(isLoading: false, isSuccess: false)),
    (r) {
      emit(state.copyWith(isLoading: false, isSuccess: true));
      showMySnackBar(
          context: event.context, message: "Registration Successful");
    },
  );
}
```

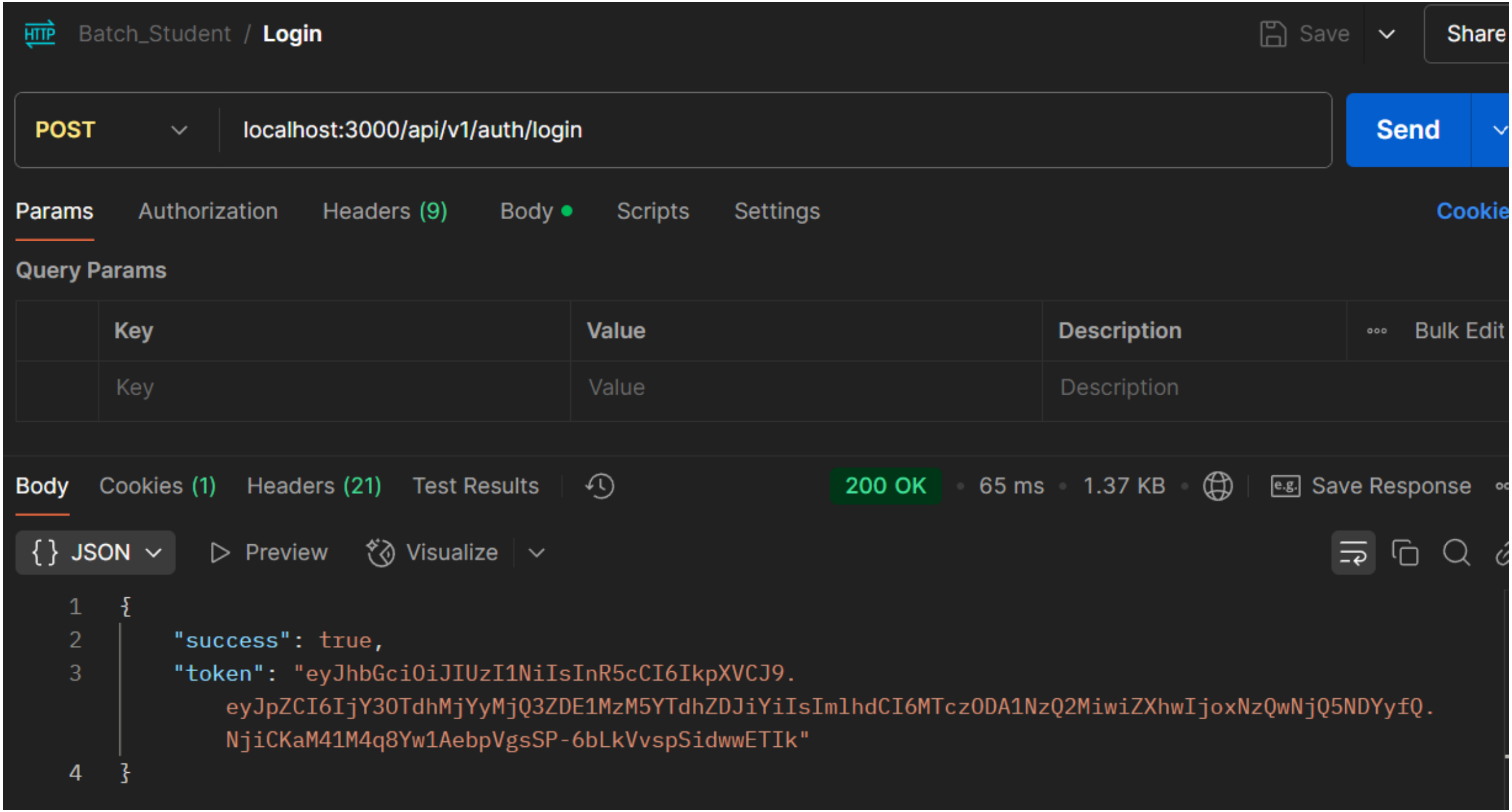
4. Register View

```
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


Login
Username
kirant
Password
.....
Login
Register
Invalid Credentials


3:26
Home
Dashboard
Dashboard
Course
Batch
Account

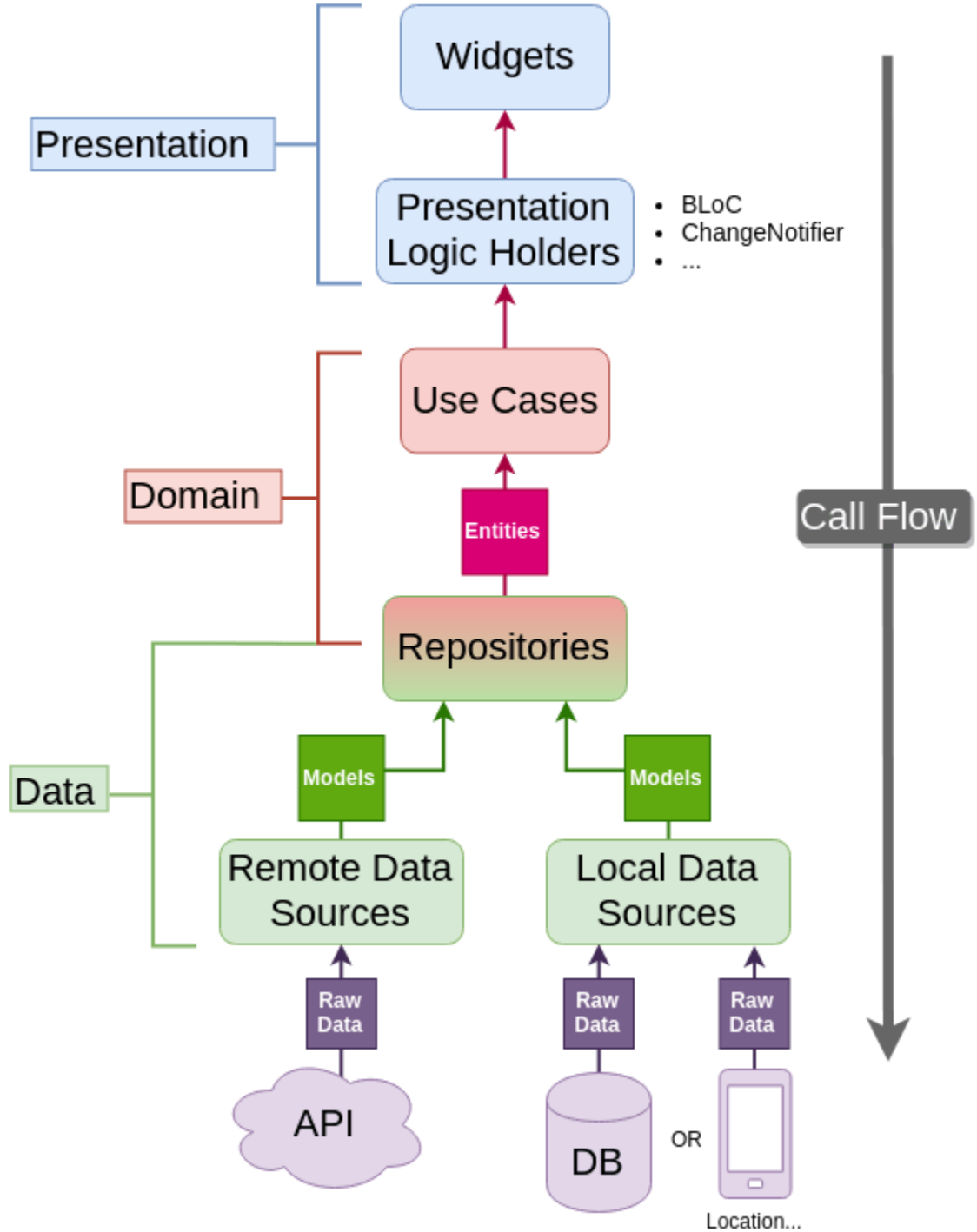

Batch_Student / Login
Save
Share
POST
localhost:3000/api/v1/auth/login
Send
Params
Authorization
Headers (9)
Body
Scripts
Settings
Cookie
Query Params
Key
Value
Description
Bulk Edit
Key
Value
Description
Body
Cookies (1)
Headers (21)
Test Results
200 OK
65 ms
1.37 KB
Save Response
JSON
Preview
Visualize
{
"success": true,
"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJpZCI6IjY3OTdhMjYyMjQ3ZDE1MzM5YTdhZDJiYiIsImlhdCI6MTczODA1NzQ2MiwiZXhwIjoxNzQwNjQ5NDYyfQ.
NjiCKaM41M4q8Yw1AebpVgsSP-6bLkVvspSidwwETIk"
}

Step 1 : Auth Remote Data Source

```dart
@override
Future<String> loginStudent(String username, String password) async {
  try {
    Response response = await _dio.post(
      ApiEndpoints.login,
      data: {
        "username": username,
        "password": password,
      },
    );

    if (response.statusCode == 200) {
      final str = response.data['token'];
      return str;
    } else {
      throw Exception(response.statusMessage);
    }
  } on DioException catch (e) {
    throw Exception(e);
  } catch (e) {
    throw Exception(e);
  }
}
```

Step 2 : Auth Remote Repository

```dart
@override
Future<Either<Failure, String>> loginStudent(
    String username, String password) async {
  try {
    final token =
        await _authRemoteDataSource.loginStudent(username, password);
    return Right(token);
  } catch (e) {
    return Left(ApiFailure(message: e.toString()));
  }
}
```


Widgets
Presentation
Presentation Logic Holders
BLoC
ChangeNotifier
...
Use Cases
Domain
Entities
Repositories
Call Flow
Data
Models
Models
Remote Data Sources
Local Data Sources
Raw Data
Raw Data
Raw Data
API
DB
OR
Location...

Step 3 : GetIt

```dart
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

```dart
abstract class Failure extends Equatable {
  final String message;

  const Failure({required this.message});

  @override
  List<Object> get props => [message];
}

class LocalDatabaseFailure extends Failure {…

class ApiFailure extends Failure {…

class SharedPrefsFailure extends Failure {
  const SharedPrefsFailure({
    required super.message,
  });
}
```

Step 2 : Token Shared Preferences

```dart
class TokenSharedPrefs {
  final SharedPreferences _sharedPreferences;

  TokenSharedPrefs(this._sharedPreferences);

  Future<Either<Failure, void>> saveToken(String token) async {
    try {
      await _sharedPreferences.setString('token', token);
      return Right(null);
    } catch (e) {
      return Left(SharedPrefsFailure(message: e.toString()));
    }
  }

  Future<Either<Failure, String>> getToken() async {
    try {
      final token = _sharedPreferences.getString('token');
      return Right(token ?? '');
    } catch (e) {
      return Left(SharedPrefsFailure(message: e.toString()));
    }
  }
}
```

Step 3 : Login Use Case

```dart
class LoginUseCase implements UsecaseWithParams<String, LoginParams> {
  final IAuthRepository repository;
  final TokenSharedPrefs tokenSharedPrefs;

  LoginUseCase(this.repository, this.tokenSharedPrefs);

  @override
  Future<Either<Failure, String>> call(LoginParams params) {
    // Save token in Shared Preferences
    return repository
        .loginStudent(params.username, params.password)
        .then((value) {
      return value.fold(
        (failure) => Left(failure),
        (token) {
          tokenSharedPrefs.saveToken(token);
          tokenSharedPrefs.getToken().then((value) {
            print(value);
          });
          return Right(token);
        },
      );
    });
  }
}
```

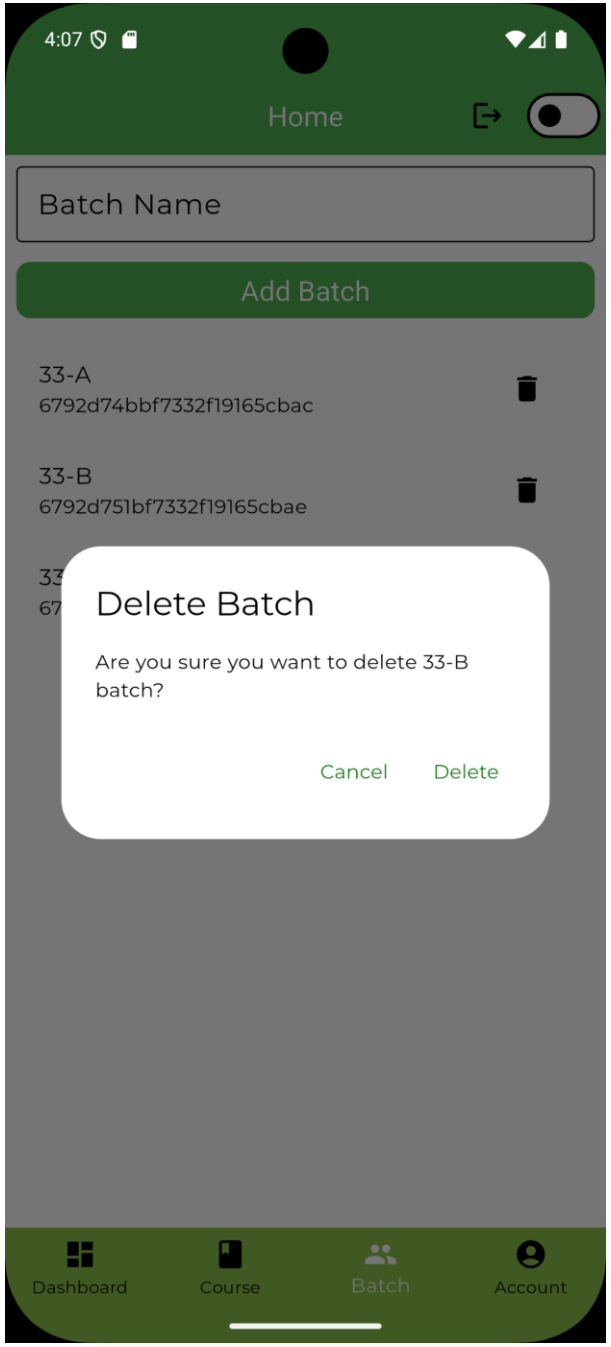
Step 4 : GetIt

```dart
Future<void> initDependencies() async {
  // First initialize hive service
  await _initHiveService();
  await _initApiService();
  await _initSharedPreferences();
  await _initBatchDependencies();
  await _initCourseDependencies();
  await _initHomeDependencies();
  await _initRegisterDependencies();
  await _initLoginDependencies();

  await _initSplashScreenDependencies();
}

Future<void> _initSharedPreferences() async {
  final sharedPreferences = await SharedPreferences.getInstance();
  getIt.registerLazySingleton<SharedPreferences>(() => sharedPreferences);
}
```

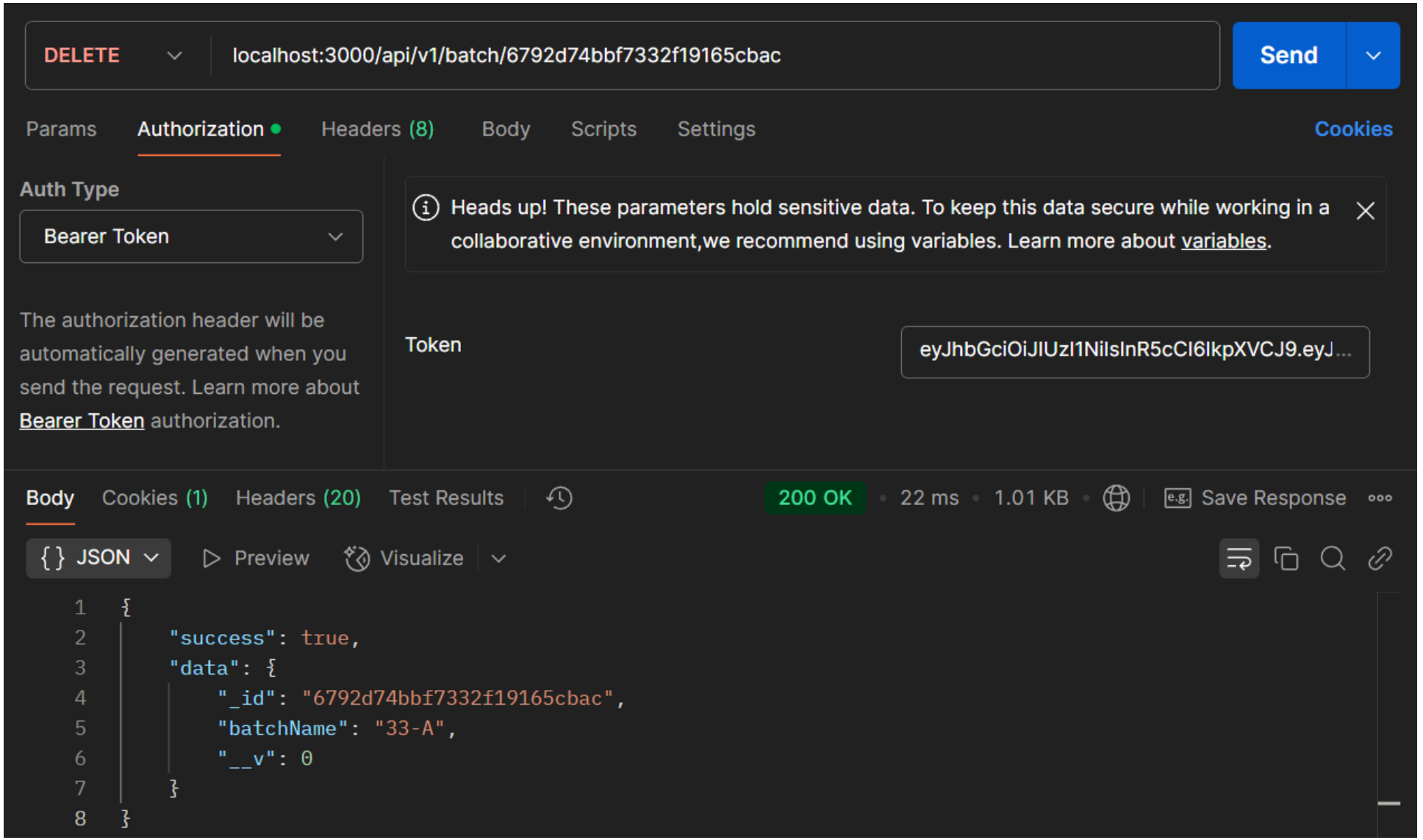
Step 4 : getIt

```dart
_initLoginDependencies() async {
  // ============================ Token Shared Preferences ============================
  getIt.registerLazySingleton<TokenSharedPrefs>(
    () => TokenSharedPrefs(getIt<SharedPreferences>()),
  );

  // ============================ Usecases ============================
  getIt.registerLazySingleton<LoginUseCase>(
    () => LoginUseCase(
      getIt<AuthRemoteRepository>(),
      getIt<TokenSharedPrefs>(),
    ),
  );

  getIt.registerFactory<LoginBloc>(
    () => LoginBloc(
      registerBloc: getIt<RegisterBloc>(),
      homeCubit: getIt<HomeCubit>(),
      loginUseCase: getIt<LoginUseCase>(),
    ),
  );
}
```

Delete Batch



Postman

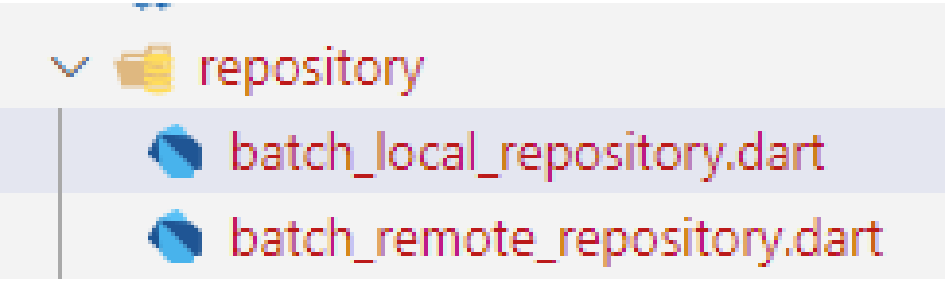

DELETE
localhost:3000/api/v1/batch/6792d74bbf7332f19165cbac
Send
Params
Authorization
Headers (8)
Body
Scripts
Settings
Cookies
Auth Type
Bearer Token
Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment,we recommend using variables. Learn more about variables.
The authorization header will be automatically generated when you send the request. Learn more about Bearer Token authorization.
Token
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ...
Body
Cookies (1)
Headers (20)
Test Results
200 OK
22 ms
1.01 KB
Save Response
JSON
Preview
Visualize
{
"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

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

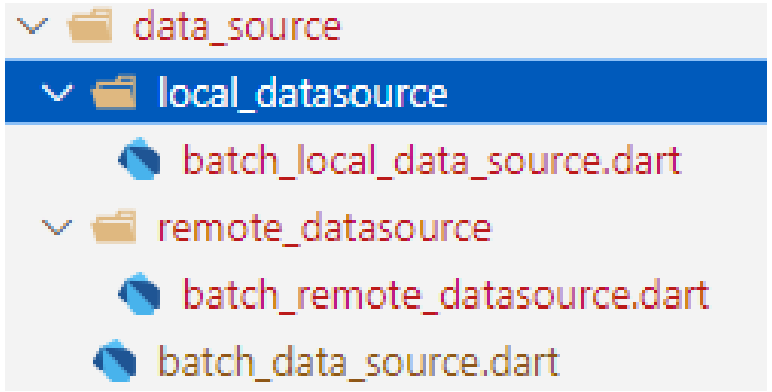


Step 3 : Batch Delete UseCase

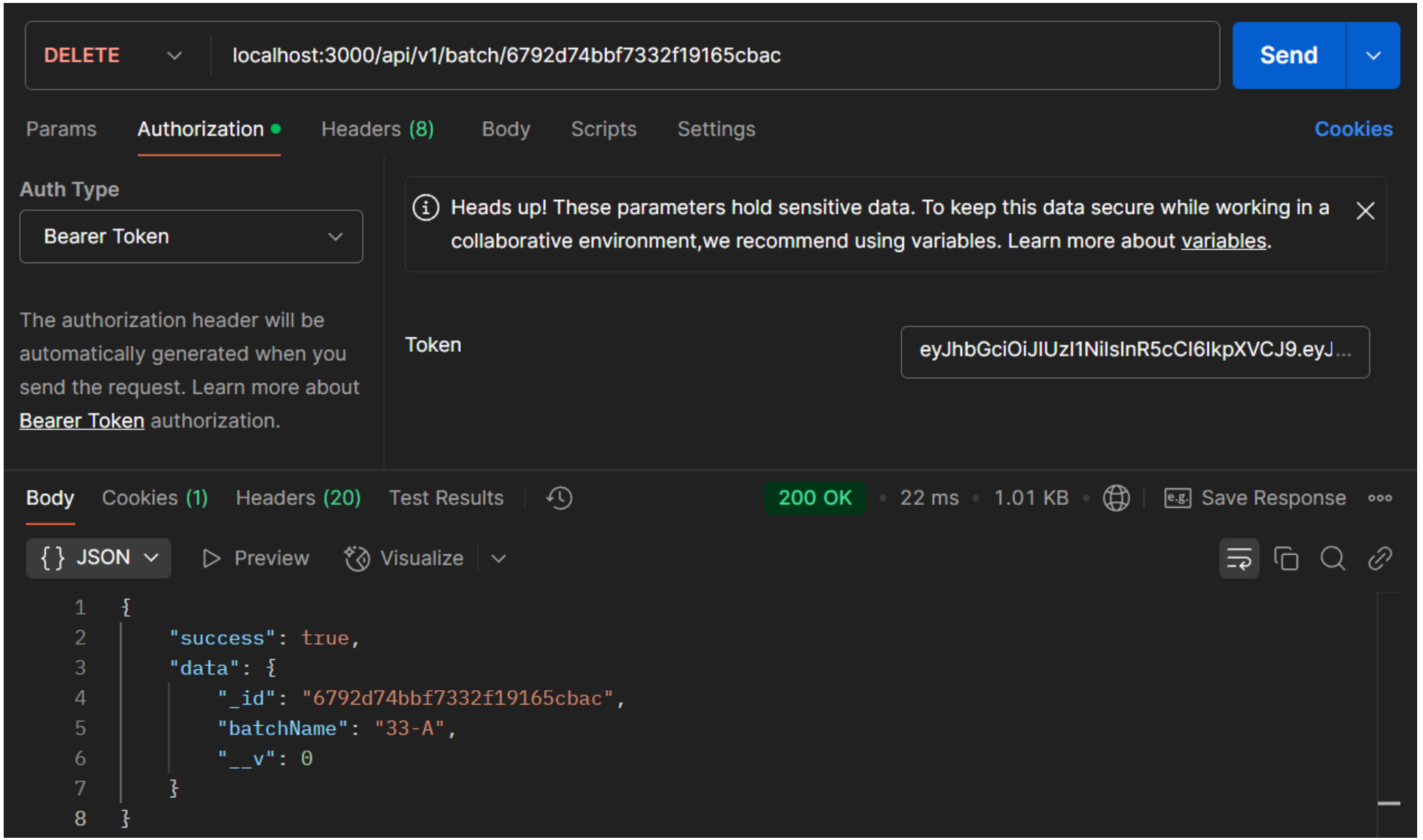
```dart
class DeleteBatchUsecase implements UsecaseWithParams<void, DeleteBatchParams> {
  final IBatchRepository batchRepository;
  final TokenSharedPrefs tokenSharedPrefs;

  DeleteBatchUsecase(
      {required this.batchRepository, required this.tokenSharedPrefs});

  @override
  Future<Either<Failure, void>> call(DeleteBatchParams params) async {
    // Get token from Shared Preferences and send it to the server
    final token = await tokenSharedPrefs.getToken();
    return token.fold((l) {
      return Left(l);
    }, (r) async {
      return await batchRepository.deleteBatch(params.batchId, r);
    });
  }
}
```

Batch Data Source

```dart
abstract interface class IBatchDataSource {
  Future<List<BatchEntity>> getBatches();
  Future<void> createBatch(BatchEntity batch);
  Future<void> deleteBatch(String id, String? token);
}
```



Postman


DELETE
localhost:3000/api/v1/batch/6792d74bbf7332f19165cbac
Send
Params
Authorization
Headers (8)
Body
Scripts
Settings
Cookies
Auth Type
Bearer Token
Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment,we recommend using variables. Learn more about variables.
The authorization header will be automatically generated when you send the request. Learn more about Bearer Token authorization.
Token
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ...
Body
Cookies (1)
Headers (20)
Test Results
200 OK
22 ms
1.01 KB
Save Response
JSON
Preview
Visualize
{
"success": true,
"data": {
"_id": "6792d74bbf7332f19165cbac",
"batchName": "33-A",
"__v": 0
}
}

Step 4: Batch Remote Data Source

```dart
@override
Future<void> deleteBatch(String id, String? token) async {
  try {
    var response = await _dio.delete(
      ApiEndpoints.deleteBatch + id,
      options: Options(
        headers: {
          'Authorization': 'Bearer $token',
        },
      ),
    );

    if (response.statusCode == 200) {
      return;
    } else {
      throw Exception(response.statusMessage);
    }
  } on DioException catch (e) {
    throw Exception(e);
  } catch (e) {
    throw Exception(e);
  }
}
```

Step 5: Batch Remote Repository

```dart
@override
Future<Either<Failure, void>> deleteBatch(String id, String? token) async {
  try {
    remoteDataSource.deleteBatch(id, token);
    return Right(null);
  } catch (e) {
    return Left(
      ApiFailure(
        message: e.toString(),
      ), // ApiFailure
    ); // Left
  }
}
```

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

```dart
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 }
```