# Flutter Project Documentation

## Project Overview

This Flutter project is structured using the following directories and files:

1. lib/main.dart
   - Entry point of the Flutter application.
2. lib/injection.dart
   - Manages dependency injection.
3. lib/core/
   - Contains core functionalities and utilities.
   - constant/: Defines app-wide constants.
     - app_constant.dart
     - app_sizes.dart
     - app_text_style.dart
   - resources/: Manages application state resources.
     - data_state.dart
   - util/: Utility functions.
     - appUtil.dart
   - usecase/: Base use case class.
     - usecase.dart
4. lib/config/
   - Contains configuration files.
   - theme/: Defines application themes.
     - light_theme.dart
   - routes/: Manages application routes.
     - app_route.dart
5. lib/data/

- Manages data sources, repositories, models, and APIs.
- api/: API providers and requests.
  - api_provider.dart
  - news_requests.dart
- datasource/remote/: Remote data sources.
  - news_datasource.dart
- exeptions/: API exceptions.
  - api_exeption.dart
- model/: Data models.
  - news_model.dart
- repository/: Repositories for data handling.
  - news_repository.dart
  - storeToHive.dart

6. lib/domain/
   - Contains domain logic, entities, and use cases.
   - entity/: Entity models.
     - news_model_entity.dart
     - news_model_entity.g.dart
   - repository/: Repository interfaces.
     - Inews_repository.dart
   - use_case/: Use cases for domain logic.
     - get_news_use_case.dart
     - get_save_data_user_case.dart
     - save_news_use_case.dart
7. lib/presentation/

- Manages UI components, pages, and state management.
- bloc/: Business Logic Components for state management.
  - news_bloc/
    - news_bloc.dart
    - news_event.dart
    - news_state.dart
- pages/: UI pages.
  - news_screen.dart
  - single_news_screen.dart
- widgets/: Custom UI widgets.
  - news_widget.dart
  - pop_up_menu.dart

**Setting Up the Project**

1. Dependencies: Ensure all necessary dependencies are listed in pubspec.yaml.

2. Environment Configuration: Configure the environment for the project by setting up the necessary APIs and keys.

3. Run the Project: Use the following command to run the Flutter project:

```
flutter run
```

**Key Components**

- main.dart: Initializes the app and sets up the main widget tree.
- injection.dart: Configures dependency injection using a service locator.
- API Integration: Handled by files in lib/data/api/.
- State Management: Implemented using Bloc pattern in lib/presentation/bloc/.

- UI Components: Defined in lib/presentation/pages/ and lib/presentation/widgets/.

## Usage

- Home Screen: Displayed using news_screen.dart.
- News Details: Managed by single_news_screen.dart.
- State Management: Events and states are defined in news_bloc/.

## Development Practices

- Code Organization: Follow the directory structure for modularity.
- State Management: Use Bloc for managing states and business logic.
- Dependency Injection: Utilize the injection.dart for injecting dependencies.