# I. App introduction

This is a basic sample app that is written following the Clean Architecture.

This app allows the user to log in to learn and practice English vocabulary. The vocabularies could be update server-side by using Firebase Real-time database.

**Source code**: <https://github.com/Tan2NT/flutter_english_hero>

**App features**

|  |  |  |
| --- | --- | --- |
| **Feature** |  |  |
| **Authentication** |  |  |
| **Study English vocabulary** |  |  |
| **Test English Vocabulary** |  |  |
| Error handling |  |  |

## II. Technical apply

|  |  |
| --- | --- |
| Technical | Material |
| Clean Architecture | Separate the application into layers:  - **core**: define some common utilities that could be apply for all project, common be used by other layer.  - **data**: represents the data-layer of the application. The Data module, which is a part of the outermost layer, is responsible for data retrieval. This can be in the form of API calls to a server, a local database, or even both.  - **domain**: encapsulates and implements all of the use cases of the system. These use cases orchestrate the flow of data to and from the entities, and direct those entities to use their enterprise wide business rules to achieve the goals of the use case.​  - **presentation**: The Presenter will have members that are functions, which are optionally set by the viewModel and will be called if set upon the Usecase sending back data to the view for display, completing, or error.  - **ui**: the layer content the views of our application. It receive data from the viewModel then displays on the screen to the user. |
| Network request | <https://pub.dev/packages/http> |
| Error handling | Handle API remote error from the base view, and if we want to handle more specific to a view, we could also override the handleException method to archive that. |
| Local database | <https://pub.dev/packages/sqflite> |
| Remote database | Using the Real time database of Firebase |
| Authentication | Using the Firebase Authentication with email/password. User can create a user and login with it. |
| Dependency injection | <https://pub.dev/packages/get_it> |
| Localization | <https://pub.dev/packages/easy_localization> |
| Test | Widget test: <https://docs.flutter.dev/cookbook/testing/widget/introduction>  API test with Mockito: <https://pub.dev/packages/mockito/install> |