Final Report for Cooking Project

Sep team

August 2020

Contents

1 Introduction

1.1 Preface

About 10 years ago, recipes could be found in magazines, or on television shows. Then we jot them down by copying them, making general notes, or buying books about recipes at the bookstore.

With the growth of the internet recently, we can easily search for recipes that are shared online, or cooking communities willing to share them online.
However, with the development of mobile technology, the need to manage separate recipes anytime, anywhere with personal phones, even without the network, has increased, users always want to store recipes. and can be found easily.
Or you can store the recipe photos, then write down the steps to create a recipe for yourself right on your phone instead of having to write them down in notes like before. When you love the recipes, users need to be able to search the recipes online, and then import them to their phone to easily store them.
That's why the team decided to create an application to manage these recipes, which would be a store for every recipe or the word "Master-Chief" or "Jan-Can-Cook" just with a small phone.

Instructors : Professor Fabien Baldacci.
Team Member :

1. **Tran Huy Phuc**
2. **Nguyen Chi Cong**
3. **Pham Hoang Huy**

1.2 App and download

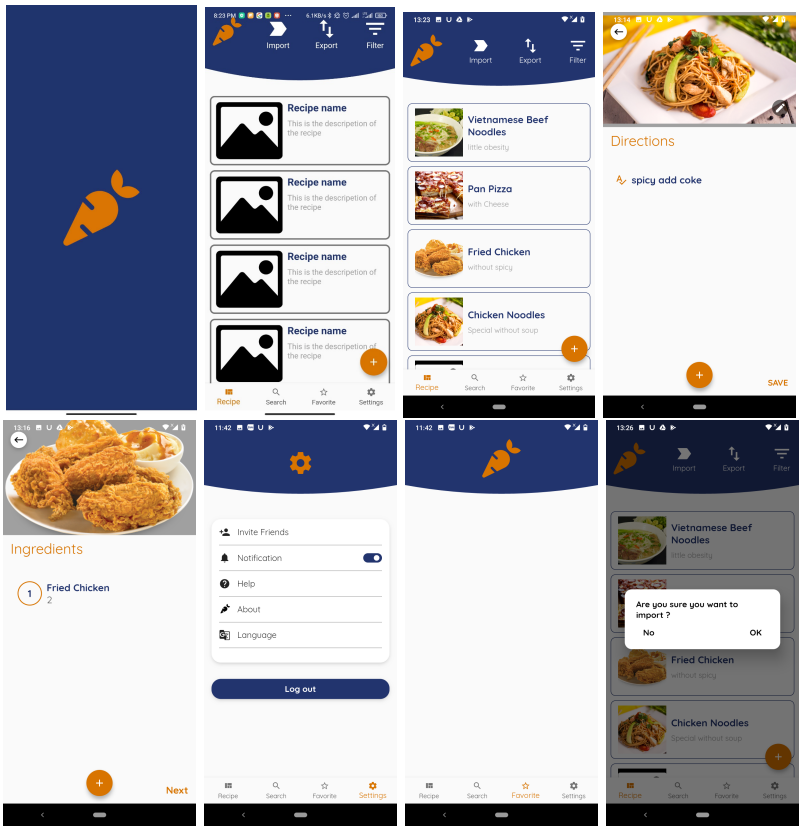


Figure 1: Cooking Book App User Interface

Download App and install

1. **apk** for android Devices. under location /final/Cookbook.apk file .
2. **api** for IOS devices.

1.3 Application Related Source

YouTube Videos : Application Demo
GitHub : Git Url
View Report online via overleaf : https://www.overleaf.com/read/kmhwshsryfjz

2 Project Details

2.1 Project Requirement

The goal of this project is to have an app on the phone to handle our own recipes. It should be able to replace our handwritten recipe book while using smartphone abilities to add smart features.

2.2 EXISTS ANALYSIS

Analyze from similar applications and traditional recipe books that people use a lot today for business analysis, then give a comparison table below.

- Recipe book : Recipe books are easy to find at bookstores, like Magnolia Table Joanna Gaines.
- Recipe website : Website of recipes, we can search, and get the recipe on this website. ex : All Recipes.
- Recipe App : But the recipe manager app, we can use these apps on our phones, and don't need internet. ex : My cook book manager .
- Recipe online app. : Application for management, social network for sharing recipes. ex : Tasty .

	Recipe Book (Magnolia Table)	Recipe Website (All Recipes)	Recipe App (My cook book manager)	Recipe Online Ap (Tasty)
My own recipe	No	No	Yes	Yes
Search recipe	No	yes	yes	yes
easy to use	no	no, need account	yes	yes
cost	costly	free	free	costly
internet	No	Require	no	Require
Mobile Use	No	Yes, Require Internet	Yes	Yes
Favorite	No	no, need account	Yes	Yes
Import, Export	No	no	yes	no

Table 1.Compare kind of existing cook book

2.3 Software Development

2.3.1 REQUIREMENT ANALYSIS

***Use-case diagram**

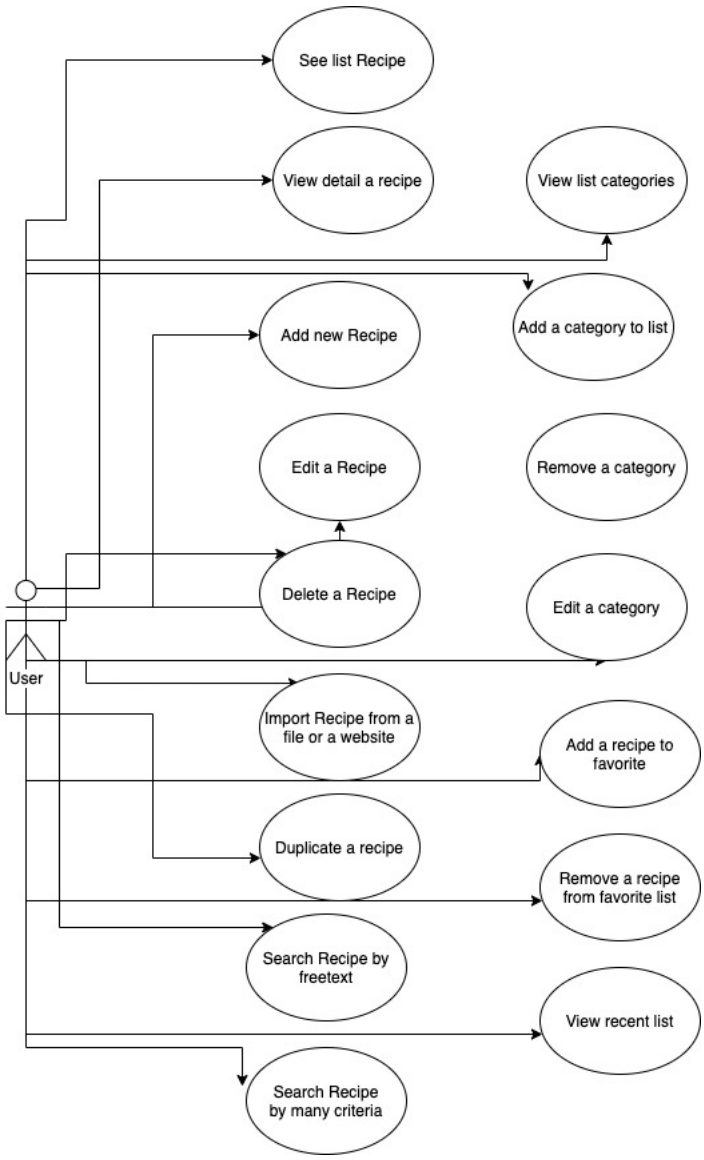


Figure 2: Use-case diagram

***Mock-up**

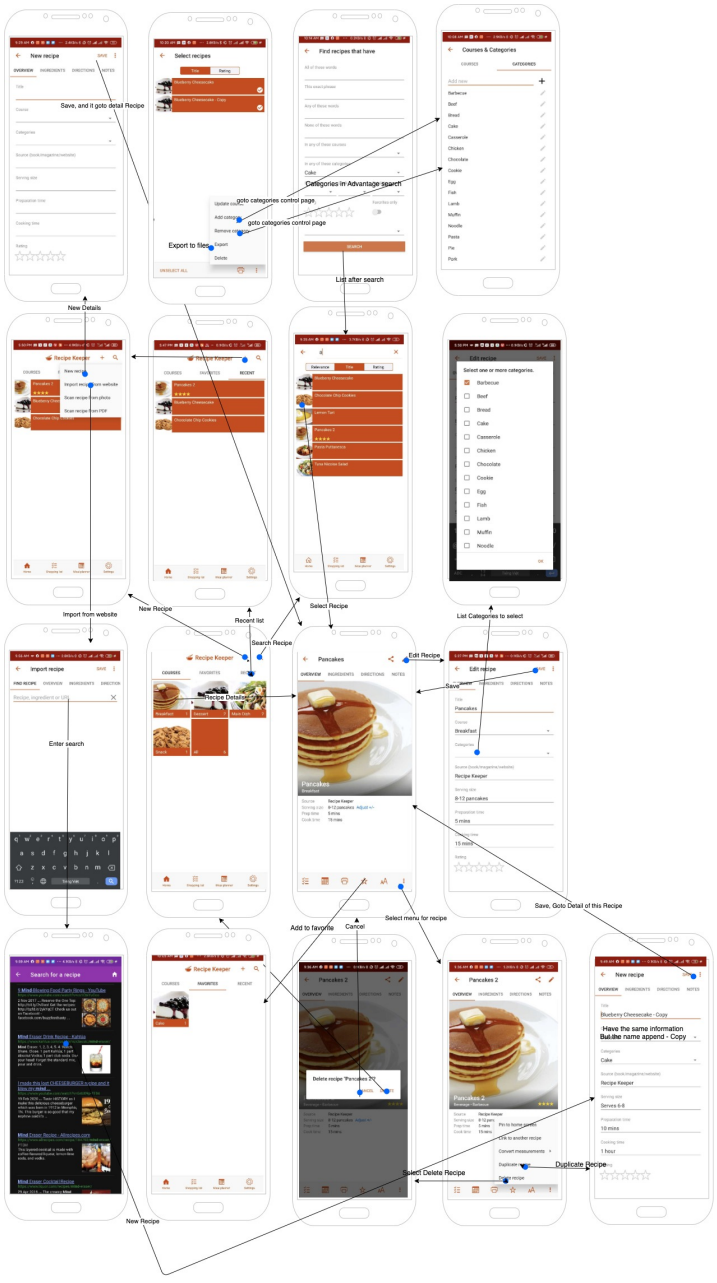


Figure 3: Mock-up UI and flow of app

* **Functional Requirement** :

1. **Recipe list** :User can view their recipe list on homepage. Each item on the list shows recipe basic information (name, image, short description).
2. **Sort recipes** : user can sort their recipes in different orders(Ex:name, created date), default order is name.
3. **Create new recipe** : user has the ability to create new recipe from the app.
4. **View recipe details** : user can click on any recipe on the recipe list to view its details.
5. **Edit recipe**: user is able to edit existing recipe and save updated details to database.
6. **Delete recipe**: user can delete an existing recipe. A popup dialog should be shown to user to confirm the deletion. After deleted, the recipe will be removed our of local database and disappears on recipe list.
7. **Duplicate recipe**: user can use this function to clone an existing recipe to new one quickly.
8. **Import recipes**: user is able to use the app to import one or more recipes from the file that was exported before. User can import recipes from the internet too.
9. **Export recipes**: from the app, user can export one or more recipes to a file. This file can be shared to other users or re-imported to the app later.
10. **Search recipe(s)**: user can search recipe(s) from local database or remote database.
11. **Category list**: user can view list of categories. Click on a category to view all the recipes belong to that category.
12. **Create new category**: from the app, user can create new category (with name and description).
13. **Edit category**: user is able to edit the information of category.
14. **Delete category**: user is able to delete existing categories. A popup dialog should be shown to user to confirm the deletion. After deleted, the deleted category will no longer be shown in category list.
15. **Favorite recipes**: user can add any recipe to favorite list, so that user can find these recipes faster.
16. **Remove recipes out of favorite list**: user is able to remove recipes out of the favorite list. A popup dialog should be shown to user to confirm the removal.

17. **Push notification**: user can receive notifications (if they allow)

18. **Resume Application**: When they're an action with higher priority (like phone call), the application should pause, and resume all progress when the action is finished.

*** Non-Functional Requirement** :

1. **Performance/Storage** : Storage of application if acceptance (¡ 100Mb)

2. **Performance/Response** : All local response action must ¡ 1s.

3. **Performance/Memory** : Memory of application if acceptance for an manage application (¡ 100Mb Memory when running).

4. **Responsiveness**: with any "unexpected event" from user, like a phone call, or user press home button, the application still works fine, and resumes after the event has completed.

5. **Portability** : User can export all user information data from a device and move it into another devices or share it to anywhere. With install file (or download it from store), user can re-import all export data.

6. **Fiability** : When there's an error, or wrong action from user, it will display user friendly message (like images must less than 2Mb) instead of a system message (like buffer is overload).

7. **Scalibilit** : The app is designed and developed to be able to handle at least thousands of recipes.

8. **Usability** : there're no "hidden function" that very complex to use or complex to find.

9. **Accessibility** : All functions need to have less than 5 steps, normal should be 3 steps.

10. **Reliability** : All the function need to work probably. If there is any wrong action from user, the app should be inform error to user. For any search recipe outside the internet (use 3rd party API), make sure return value must be correct and does not break any function.

11. **Screen Adaption**: Support the most popular mobile screen size from 4.0 inch to 7.0 inch. Able to render its layouts on different screen sizes. Along with automatic adjustment of font size and image rendering.

12. **Modifiable** : The application can be changed easily when 3rd party API has been stop or update structure.

13. **Security**: There're no user data, or any sensitive data store by application, or publish into internet.

14. **Maintainability** : Application written by Flutter (widely supported by community and easy to learn). There're document and comment code for any function/class. There're no function/class more than 2000 lines code.

15. **Compatibility** : support the most used platforms today on android platform and ios : Minimum supported Android version: the application is able to run on Android 5.0 and up. Minimum supported IOS version: the application is able to run on iOS 10 and up.

16. **Localized** : The app supports English but also has the ability to support other languages easily.

17. **Version control ability** : All code and documents store with control version on GitHub, that we can manage the version of all code/documents

2.3.2 ARCHITECTURE

***Overall architecture**

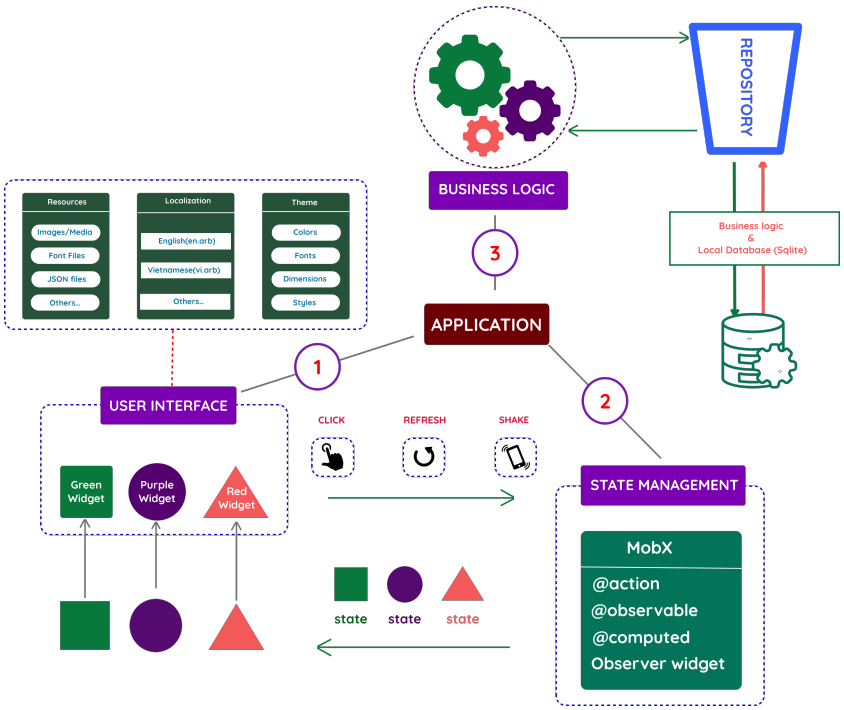


Figure 4: The overall architecture of the application

***Components** :

1. **User Interface** : A place to display information to users, and to handle events on the user's side - interactive with smart phone devices, included UI (Display User Interface), Resource (Images, json file store, v.v..), Localization (Resource file for translation data store in smart devices) Themes (Color, Front, Dimensions, Styles , v.v...).

2. **Business Logic** : Business logic layer and business layer, to process business information from users, or to interact with the database. Included Repository - business logic process components for recipe and related.

3. **State Management**

2.3.3 IMPLEMENTATION CHOICES

:

***Programming language** :
Dart is the programming language we will use.

"Dart is a client-optimized[8] programming language for apps on multiple platforms. It is developed by Google and is used to build mobile, desktop, server, and web applications.[9]
Dart is an object-oriented, class-based, garbage-collected language with C-style syntax.[10] Dart can compile to either native code or JavaScript. It supports interfaces, mixing, abstract classes, refiled generics, and type inference.[11]"*
By Wikipedia.

Dart is very powerful when combined with Flutter in developing cross-platform mobile applications, and that why team decides to use the combo Dart and Flutter to develop the mobile Application. Dart and Flutter has the growth rate is very fast in recent years.

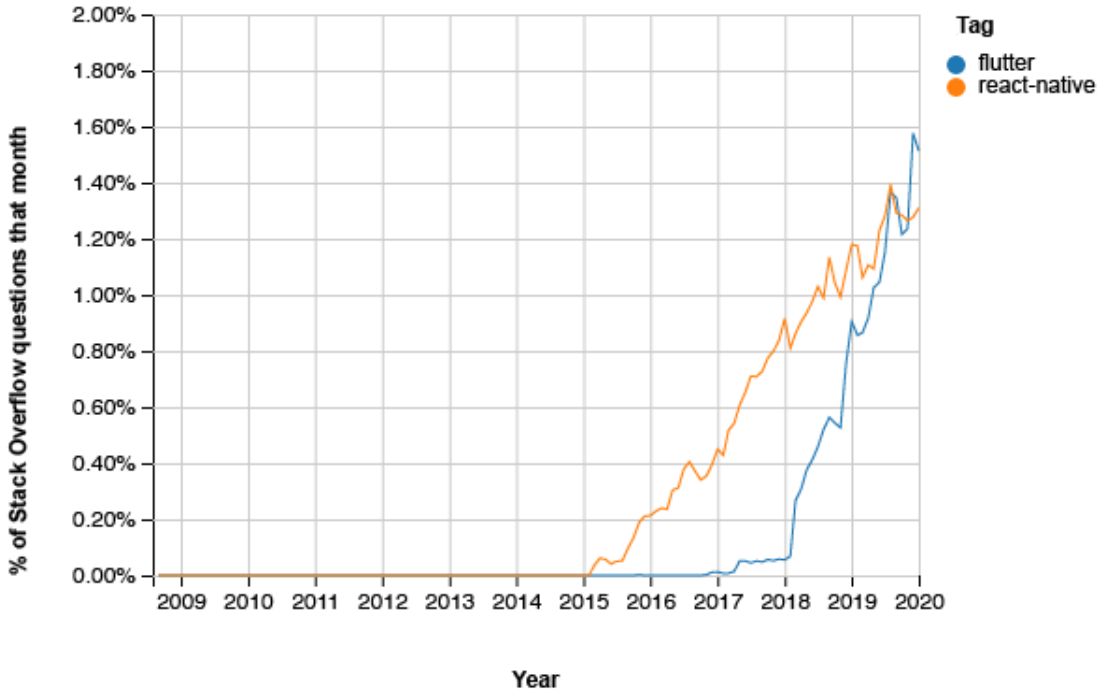


Figure 5: Stack Overflow Trends — Comparison of Flutter vs. React Native

*Source : Stackoverflow Insight

***Framework**:

Front-end (Mobile app)
We only consider cross-platform frameworks: React Native or Flutter. The reasons behind this decision are:

1. After analyzing the requirements (both non-functional and functional requirements), we concluded that the application does not require any specific native function that cross-platform framework can not do.

2. We don't have enough time and resources to build and maintain 2 separate code bases (one for Android and one for iOS)

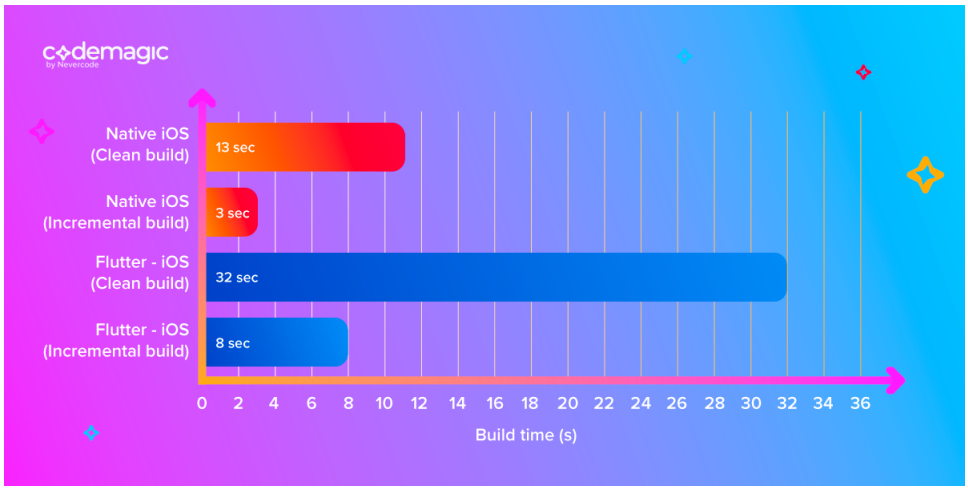


Figure 6: Flutter vs native application comparison time

*Source : Codemagic

After we finished our research (visit their official website and documentation, read many articles (listed in references 4a. at the end of this document)), we ended up to choose Flutter for this project, because:

1. Flutter has better performance(even better native in a few cases).

2. Flutter has better documentation.

3. Flutter is being developed and maintained by Google officially, while React native is not supported officially by Facebook anymore.

4. Flutter will support web and PC(and probably other platforms in the future).

References

1. https://www.thedroidsonroids.com/blog/flutter-vs-react-native-what-to-choose-in-2020
2. https://nevercode.io/blog/flutter-vs-react-native-a-developers-perspective/
3. https://medium.com/@adhithiravi/react-native-vs-flutter-what-are-the-differences-b6dc892f0d34
4. https://medium.com/@moqod-development/flutter-vs-react-native-for-cross-platform-development-821b44138b4a
5. https://medium.com/swlh/flutter-vs-native-vs-react-native-examining-performance-31338f081980
6. **Flutter documentation:** https://flutter.dev/docs

2.3.4 LISIBITLY

: in cooking app, we included :

1. : Analyzer for Dart to check rule of code, or comment, v.v...

2. : Dartdoc : To generate document of Dart programming language code. Docs api at location : /final/source$_c$*ode*/*doc*/*api*/ ∗ *Location*

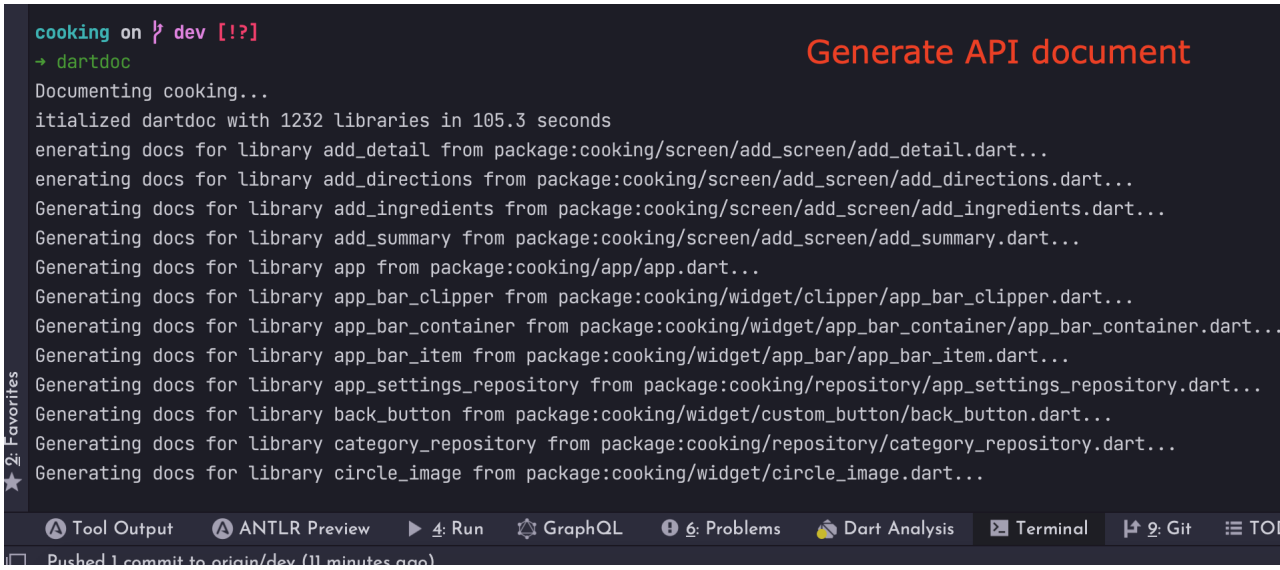


Figure 7: Generate Document

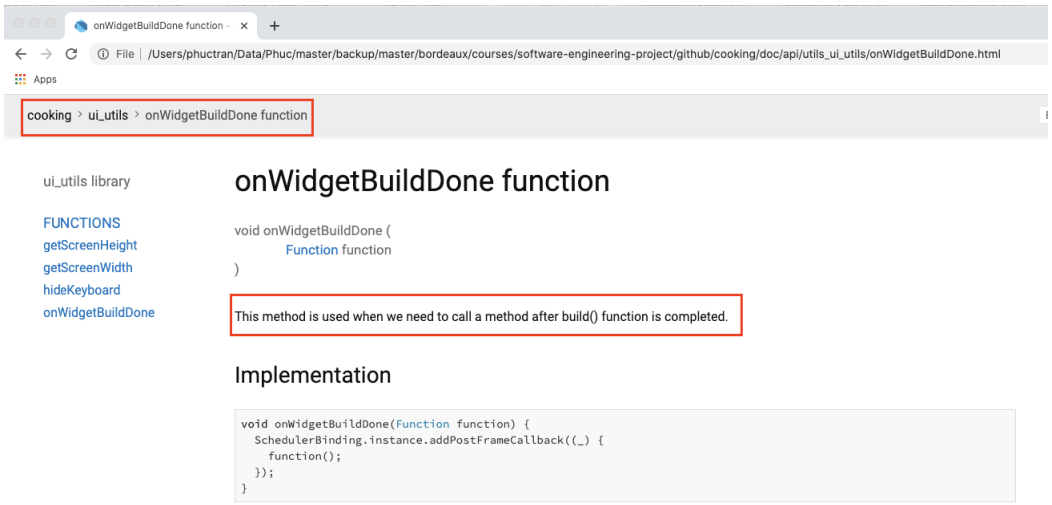


Figure 8: After generate, it should document api

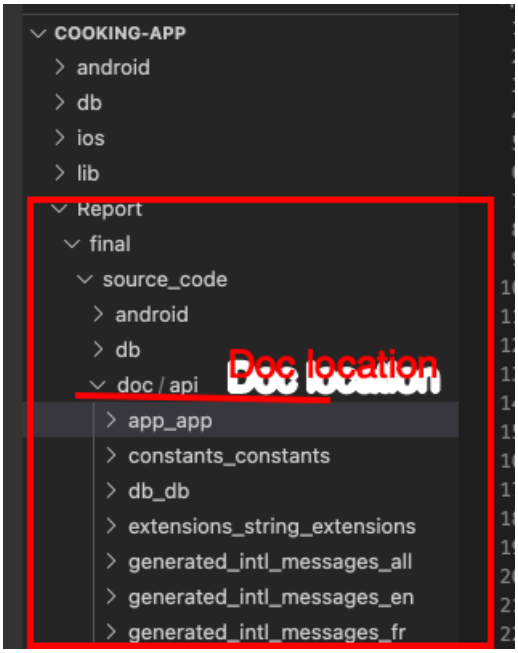


Figure 9: Doc Location Under /final/source$_c$ode/doc/api/$*$

2.3.5 ROBUSTNESS

: in cooking app, we included :

1. Unit-test : Unit test for repository - business layer of recipe and recipe categories, when every time we have new code, we can re-run the whole unit code to make sure the system still working fine with Mock data.

2. Log : We use fire-base to log error/exception .

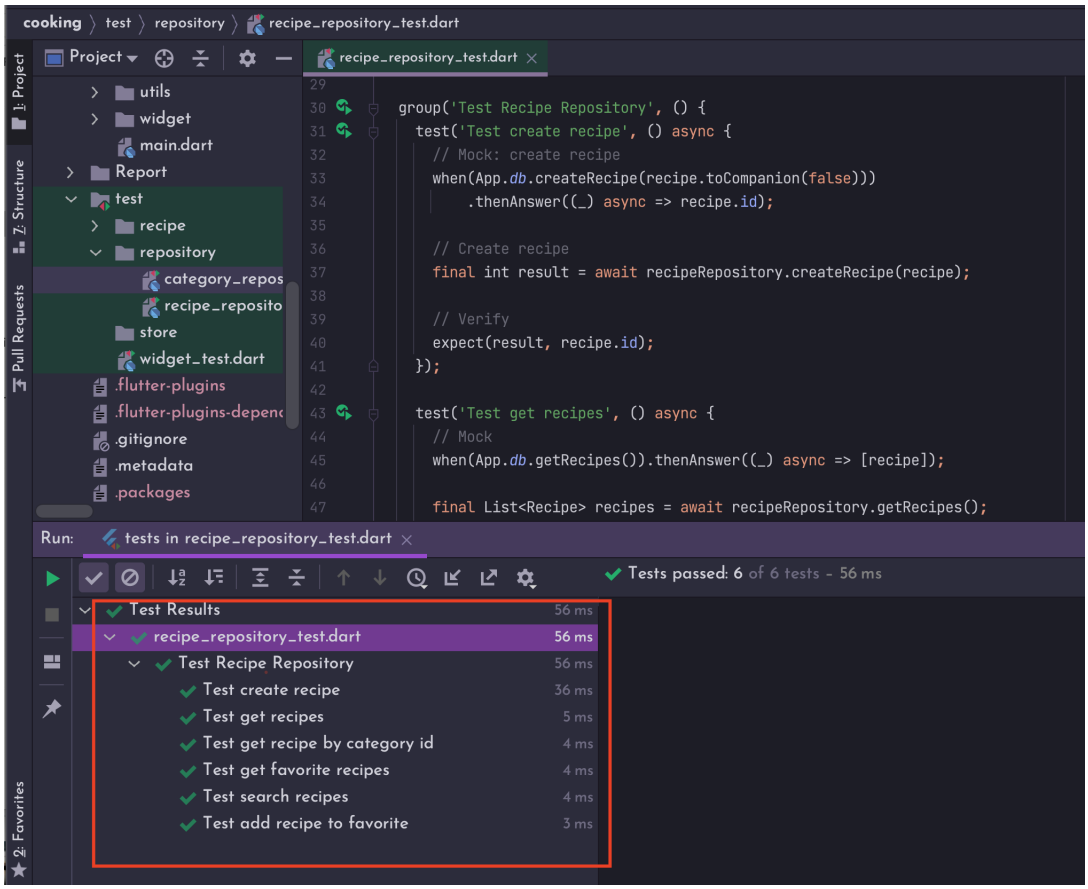


Figure 10: Unit-test of repository (all passed)

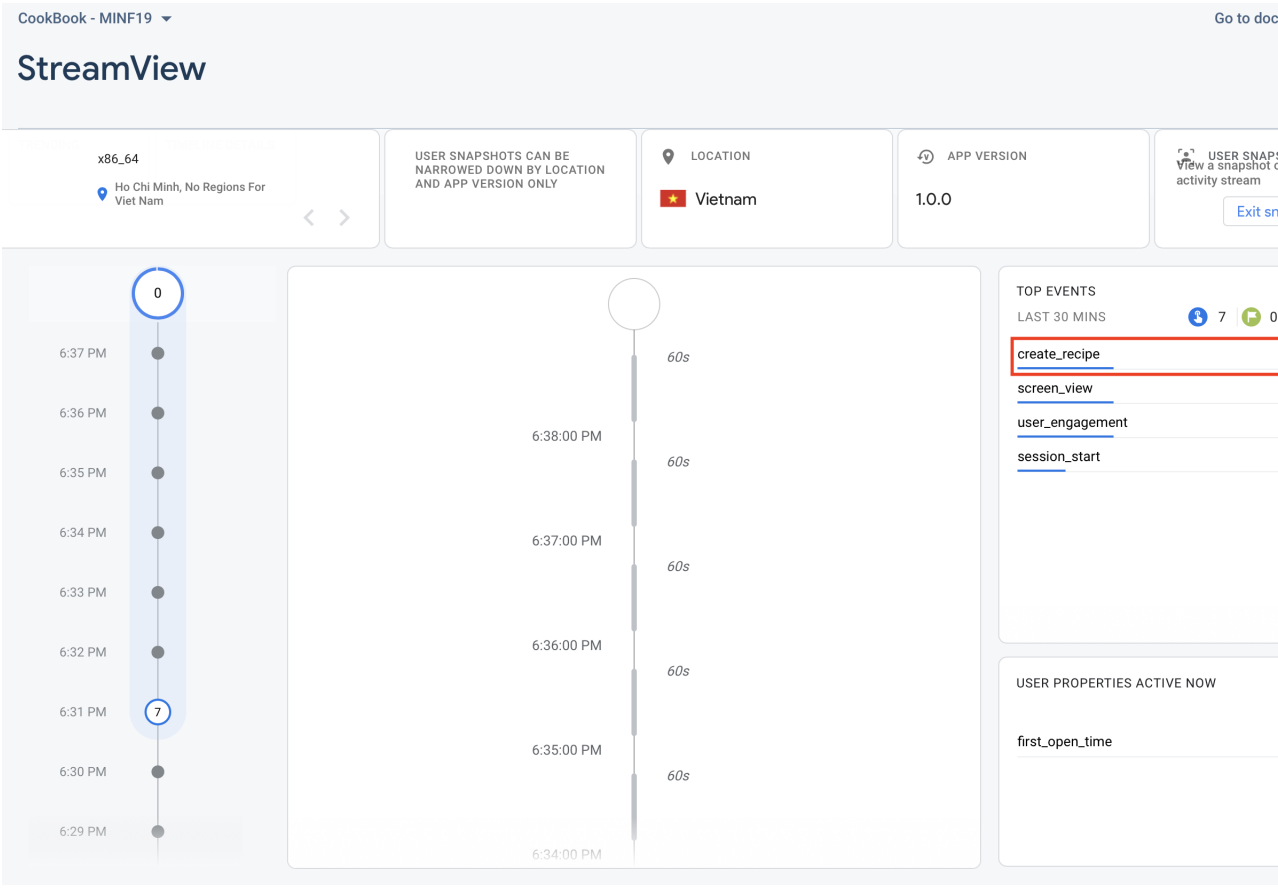


Figure 11: Fire-base for logging

```
        text: S.of(context).share,
        onPressed: () {
          FirebaseAnalytics()
              .logEvent(name: Constants.ANALYTICS_SHARE);
          Share.share(S.of(context).msgShare,
              subject: S.of(context).shareSubject);
        }),
```

```
onPressed: () {
  FirebaseAnalytics()
      .logEvent(name: Constants.ANALYTICS_EVENT_CREATE_RECIPE);
  navigateTo(AddDetail());   Tran, Yesterday • Upload latest code
},
```

Figure 12: Setup Fire-base

2.3.6 BLACK BOX TEST

Acceptance test:
For project, we design different functions such as: Import, Export, Filter, Recipe, Search, Favorite, and Setting.

-**Test case Add New Recipe** :
-Description : Test function when user add new recipe.
++ Step 1 : Open the app
++ Step 2 : Go to Recipe Tab
++ Step 3 : Click (+) Button in
++ Step 4 : Fill-in information of new recipe
++ Step 5 : Click Save Button
-Expected result : The list recent recipe should show this new recipe have just created.
-Actual Result : As expected
-Pass/Fail : Passed
-Comment :
-Result :

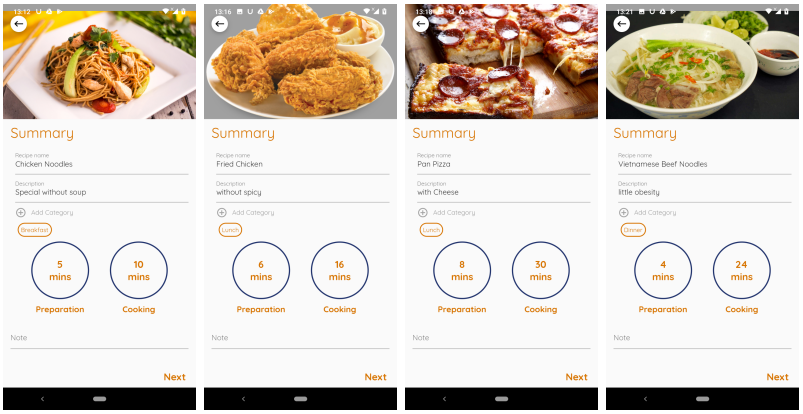


Figure 13: Result for create new recipe success
For this step, a user can add Fired Noodles, Fried Chicken, Pan Pizza, and Vietnamese Beef Noodles. This activity is created to add the recipe include ingredient and direction steps.This stage will help the user can add photo of food and write Recipe name, ,Description, Add Category, Preparation or Cooking.After full filling, the next button will move to Ingredients and Direction when the user selects.

-**Test case search recipe** :
-Description : Test function when user search a recipe .
++ Step 1 : Open the app
++ Step 2 : Go to Search tab
++ Step 3 : Fill keyword into search box
++ Step 4 : Click search button
-Expected result : The list recipe should show correct with keyword filled-in .
-Actual Result : As expected
-Pass/Fail : Passed
-Comment :
-Result :

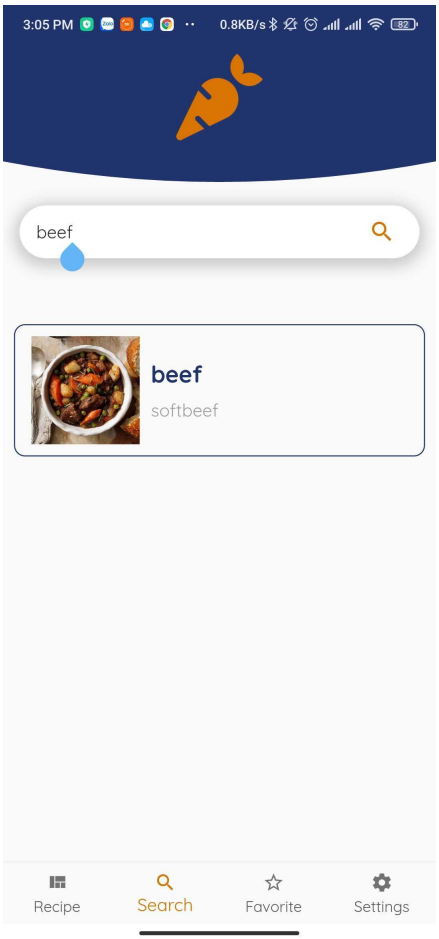
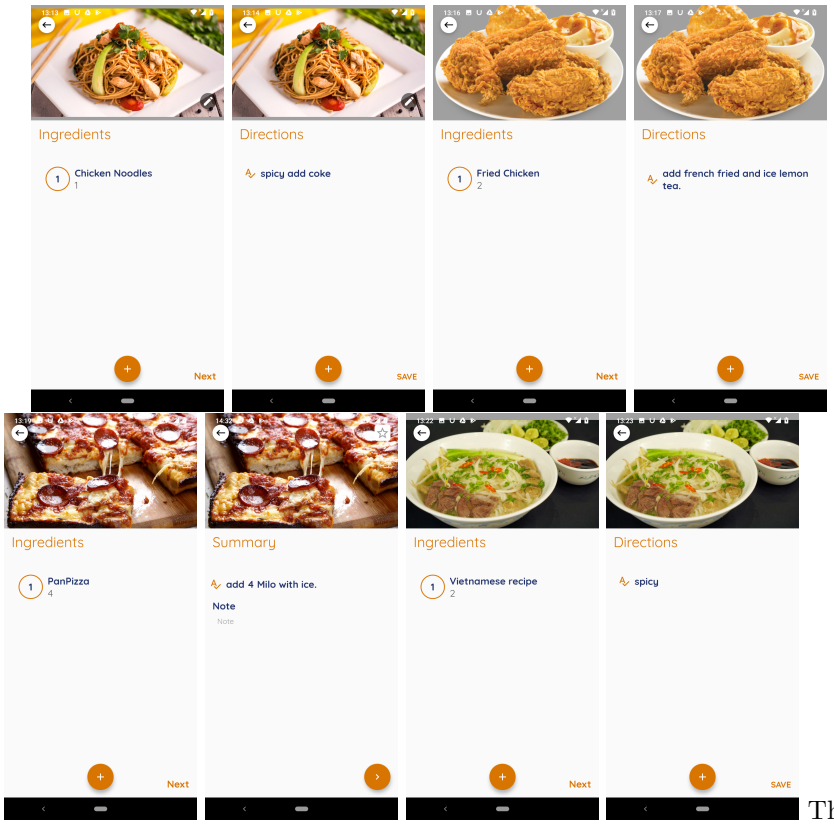


Figure 14: Result for search 'beef' keyword

-Test case add category for recipe :
-Description : Test function when user view a recipe.
++ Step 1 : Open the app
++ Step 2 : Go to Recipe Tab
++ Step 3 : Click (+) Button in
++ Step 4 : Fill-in information of new recipe
++ Step 5 : Click add categories Button
++ Step 6 : Select Category
++ Step 7 : Click Save Button
-Expected result : Category must show as same as when user create recipe.
-Actual Result : As expected
-Pass/Fail : Passed
-Comment : -Result:



The categories include Fired Noodles, Fired Chicken, Pan Pizza, and Vietnamese Beef Food.Besides the user can be able to add drinks. In addition, the user can view the time for Preparation or Cooking.

Figure 15: Result for create new recipe success

-Test case add recipe to favorite :
-Description : Test function when user add a recipe to their list of favorite .
++ Step 1 : Open the app
++ Step 2 : Go to Recipe Tab
++ Step 3 : Select a recipe to view detail
++ Step 4 : Click star Button in the top of right corner to add an item to favorite list
++ Step 5 : Go back into main screen
++ Step 6 : Go to tab Favorite
++ Step 7 : Check if favorite recipe just added into the list
-Expected result : When added into favorite list, and open the tab Favorite, it should appear the icon just added. .
-Actual Result : As expected
-Pass/Fail : Passed
-Comment : -Result:

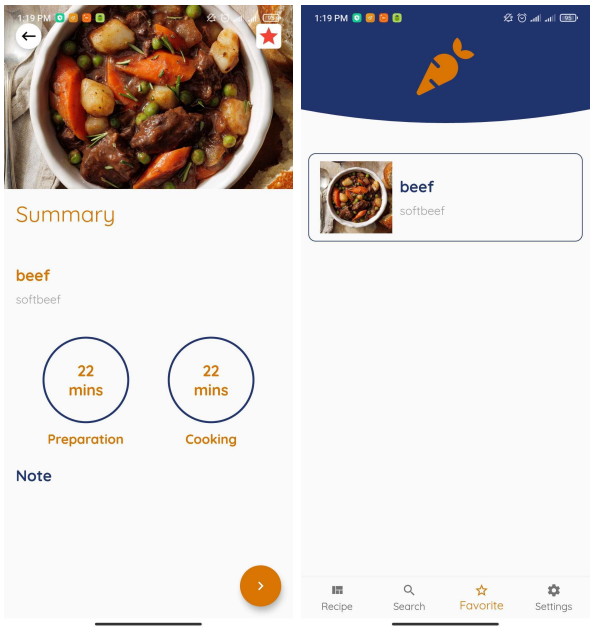


Figure 16: test for add to favorite

-**Test case remove recipe to favorite** :
-Description : Test function when user remove a recipe to their list of favorite .
++ Step 1 : Open the app
++ Step 2 : Go to Recipe Tab
++ Step 3 : Select a recipe to view detail
++ Step 4 : Un-check star Button in the top of right corner to add an item to favorite list
++ Step 5 : Go back into main screen
++ Step 6 : Go to tab Favorite
++ Step 7 : Check if favorite recipe just remove into the list
-Expected result : When remove a recipe into favorite list, and open the tab Favorite, it should not appear the recipe in favorite list. If the list empty, it should show message as this favorite list is empty. -Actual Result : As expected
-Pass/Fail : Passed
-Comment : -Result:

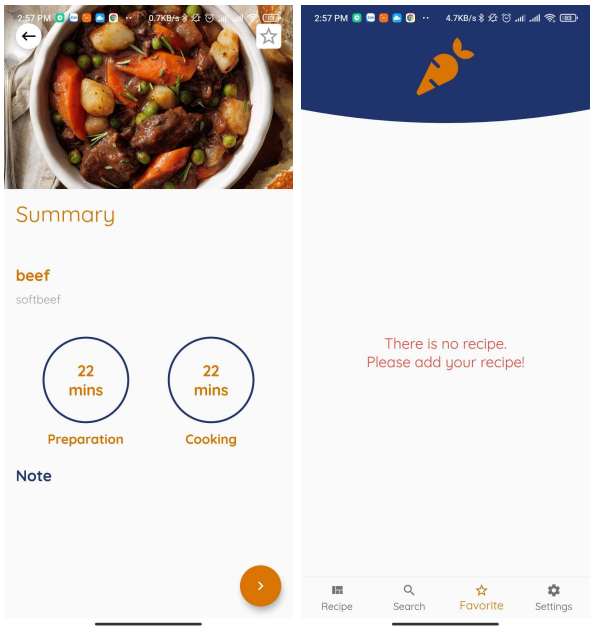


Figure 17: test for remove to favorite

-**Test go-to setting screens** :
-Description : Test function when user goto setting screen.
++ Step 1 : Open the app
++ Step 2 : Go to Settings tab
-Expected result : It should show setting pages include 4 buttons : Share, Help, Language, About button.
-Actual Result : As expected
-Pass/Fail : Passed
-Comment :
-Result :

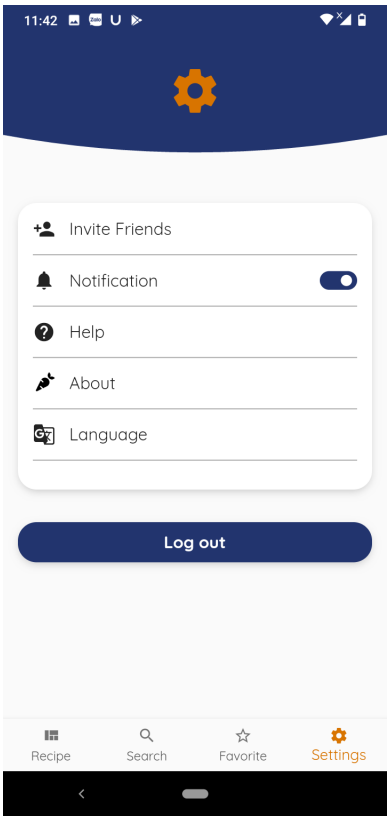


Figure 18: Result for go-to setting tabs
For this function, it will display the layout with some functions like Share, Help, Language, and About.

2.3.7 WHITE BOX TEST

Unit test:
Built-in test package of Flutter (https://flutter.dev/docs/cookbook/testing/unit/introduction)
"Mock": use mockito (https://flutter.dev/docs/cookbook/testing/unit/mocking)
CI (continuous integration): Github action
When we merge code from any branch to "master" branch, pre-define script on Github action will be triggered to run all the tests. If the tests are passed, it continues to build .apk file and stores it on Github (we probably send this apk to Slack channel if we need). State management in Flutter: We use mobX (https://mobx.netlify.com) because of its simplicity.

Two main business logic repository need to test is :
Unit test for recipe repository:

1. Test create recipe **(Passed)**
2. Test get recipes **(Passed)**
3. Test get recipe by category id **(Passed)**
4. Test get favorite recipes **(Passed)**
5. Test search recipes **(Passed)**
6. Test add recipe to favorite **(Passed)**

Unit test for categories repository:

1. Test create category **(Passed)**
2. Test update category **(Passed)**
3. Test get categories **(Passed)**

Unit test for categories : Unit test for recipe

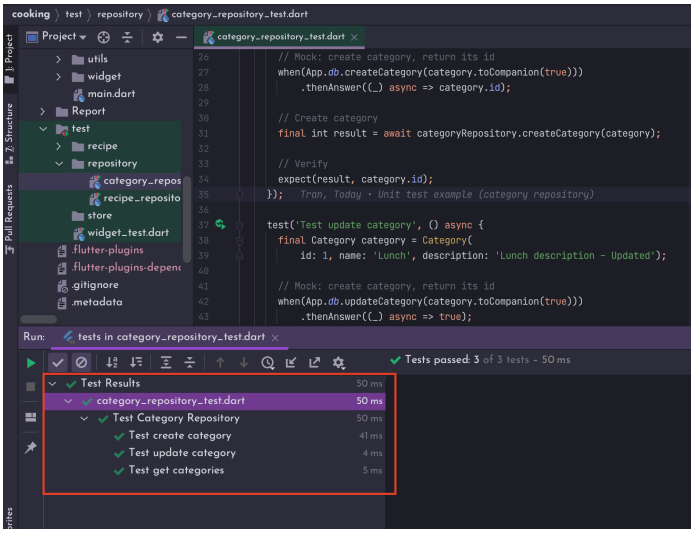


Figure 19: Unit test for categories

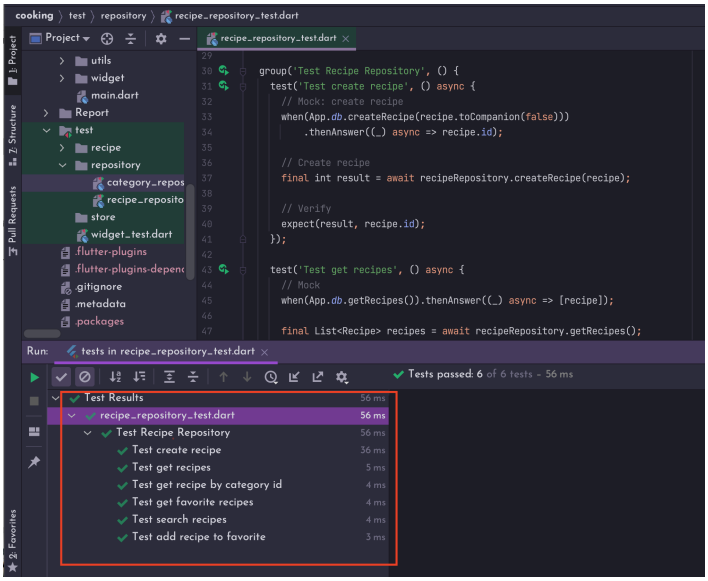


Figure 20: Unit test for recipe

2.3.8 EXPERIMENTS, SOME TESTS, SOME FIRST RELEASE

For plan of development of team include 3 phases :

1. **Phase 1 : 30 Jun 2020** : Initialize application with flutter code. As we need verify that team can work with new framework and technology, that we need try to new an empty app, and prepare all enviroments and tools for next step. On end of Jun, we have an empty app.

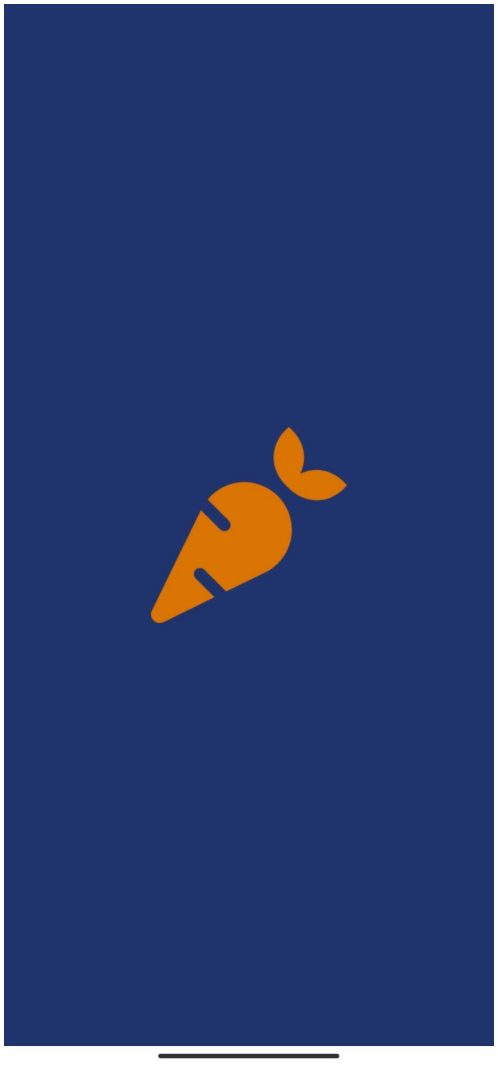


Figure 21: Version 0.1 for app - Just display fashscreen

2. **Phase 2 : 30 Jul 2020** : Initialize main function and architecture, so we need a plan on end of Jul 2020, all main architecture need to be implemented, all main screen need to initialize, (home page, search recipe page, edit/add new recipe). After that time, we have an application install file that can install in android with that install file all function are working, but test may not cover all function.

Figure 22: Version 1 for app

3. **Phase 3 : 30 Aug 2020** : Final for the rest function and test all function to make sure we can release it. On this phase, all Unit test need passed.

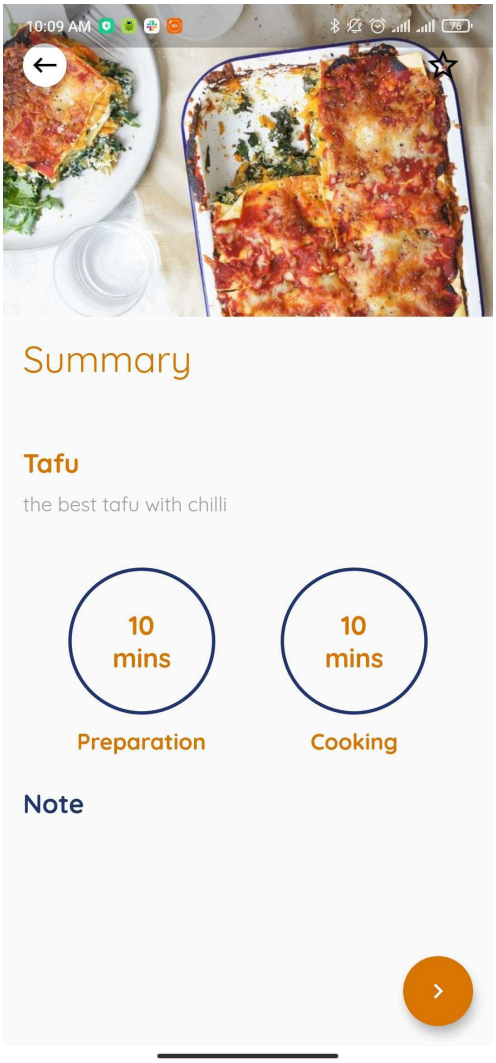


Figure 23: Version 2 for app

2.3.9 RATIONALIZED DEVELOPMENT

1. **IDE** : Intellij

2. **Version control software** : git , Git Repository : https://github.com/thphuc/cooking-app . All member has their owns branch in local, but all working in dev branch.

3. **Builders** : Intellij build-in builder for flutter to build android and ios program.

4. **Debuggers** : Flutter DevTools - DevTools is a suite of performance and debugging tools for Dart and Flutter. It's currently in beta release, but is under active development. **What we can do with DevTools** :
   -Inspect the UI layout and state of a Flutter app.
   -Diagnose UI jank performance issues in a Flutter app.
   -CPU profiling for a Flutter or Dart app.
   -Network profiling for a Flutter app.
   -Source-level debugging of a Flutter or Dart app.
   -Debug memory issues in a Flutter or Dart command-line app.
   -View general log and diagnostics information about a running Flutter or Dart command-line app.
   -Analyze code and app size
   -And many more ...

5. **Documentation** : Dartdoc - The **dartdoc** command creates API reference documentation from Dart source code.

6. **Tests** : Flutter support it's own testing library include 3 levels :
   -Unit test : handy for verifying the behavior of a single function, method, or class. The **test** package provides the core framework for writing unit tests.
   -Integration test : test how individual pieces work together as a whole (Unit and Widget), or capture the performance of an application running on a real device. These tasks are performed with integration tests. The **flutter-driver** package provides the core framework for writing Integration tests
   -Widget test: handy for verifying the behavior of widget class . The **flutter-test** package provides the core framework for writing Widget tests.
   On this project, we most focus on Unit Test only. It's supported from Flutter framework.

7. **Performance estimation and memory analysis** : Flutter DevTools - DevTools is a suite of performance and debugging tools for Dart and Flutter. It's currently in beta release, but is under active development.

2.4 Scientific approach

2.4.1 justify facts and choices (references, alternatives, tests, experiments)

All the report included explain the solution, and reference .

2.4.2 being rigorous during verification of the work done (tests, experiments)

All function include Unit Test to make sure the program working well.

2.4.3 knowing context and domain of application (bibliography, existing analysis)

Please refer Requirement Analysis for more Details.

2.4.4 write a documentation and confrontation of ideas

Included in the report.

3 Conclusion

After developing a software product, the team realized that it not only involves writing software, but in that, in order to ensure the product quality, it takes a lot of work, and the team's support, from source code management, to writing documentation, or writing unit-tests to secure software products. After the process, everyone on the team gained more experience in the process of producing a big software product.

4 Appendix