Report 4.1

Logical Design

Le Nhu Chu Hiep

August 10, 2020

1. Introduction

This is a report in the hivilake project document about revision of the hivilake architecture. After meeting with my supervisor - Dr. Tran Giang Son, there are several problem with the naming, the granularity of the architect diagram. Therefore, this diagram was born to fix them.

2. Objective

This report intents to rename some sub-system, refactor misunderstand diagram and provide more detail diagram. This report can be consider as a appendix for report 4.

3. Methodoglogy

In this section, we provide the usecase diagram for usecase list in report 4. And the refactor diagram for the logical diagram will be mentioned. Moreover, the layout architecture will be added for a better vision about system.

3.1 Usercase Diagram

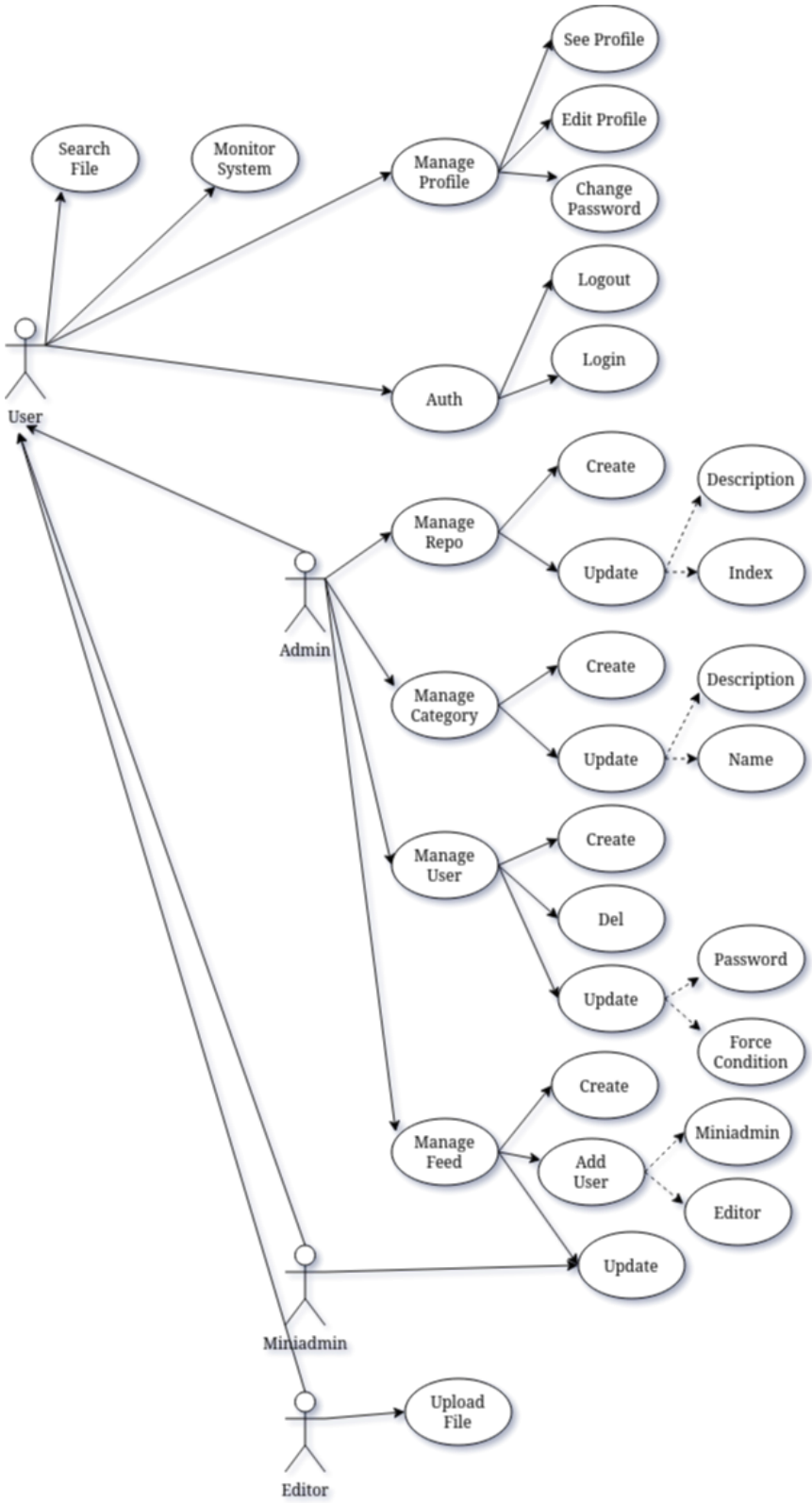

Figure 1: Usecase Diagram

In this usecase diagram, there are 4 main user which is user (general user) and its sub-users include Admin (who should have control permission in whole system), Miniadmin (who can control a feed) and finally, Editor is the user can upload file to datalake.

Besides that, the system has 4 big general usecase:

1. Auth: allow user login and logout.
2. Manage Profile: allow use interact with their own profile.
3. Monitor System: allow use monitor the healthy of each repository.
4. Search File: allow use search, see detail and download file from lake.

And the specifics user has 5 more big usecase:

1. Manage Repo: allow use interact with the repository in system.
2. Manage Category: allow admin create and update category.
3. Manage User: allow admin control user in system.
4. Manage Feed: allow user create and control the feed in system.
5. Upload File: use to upload data into lake.

NOTES This usecase diagram is built for a better hivilake design only. For a real implement system and terminology definition is job of developer Nguyen Viet Dung, so the better specification and terminology will be not include in this report.

3.2 Layout Diagram

This diagram is added into document for clarifying the structure of the system.

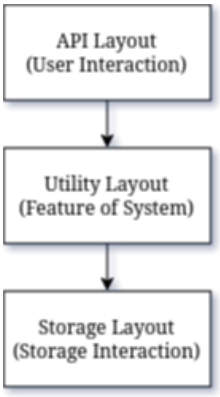

Figure 2: Layout Diagram

The hivilake splits into 3 main blocks:

- The API block is the highest abstraction. It enables higher user to use utility of system through the friendly api (Rest API for now). These API of system is connected into second layer of system - Utility.
- The Utility or second layer can be consider as the feature layout. All features of the hivilake can be called by API will be defined in here. They are logic behind the sence. The developer in this layer should be concentrate only on what they can provide for the user.
- The final layer is the storage layout. Since the feature layout developer should not care to much about the way to handle with the storage. So this layout provide an abstraction for the storage system. It allows the higher layout see the storage as a simple local system with a bit extension function and exception. In this way, the developer in the feature space does not need to care too much about the lower storage system. Moreover, this abstraction also allows - in some case - the feature space developer write their utility once and it will work with many different storage system.

3.3 Logical Diagram

In this revision, the logical diagram remove the unnecessary detail, and sketch out the logical block and its feature only.

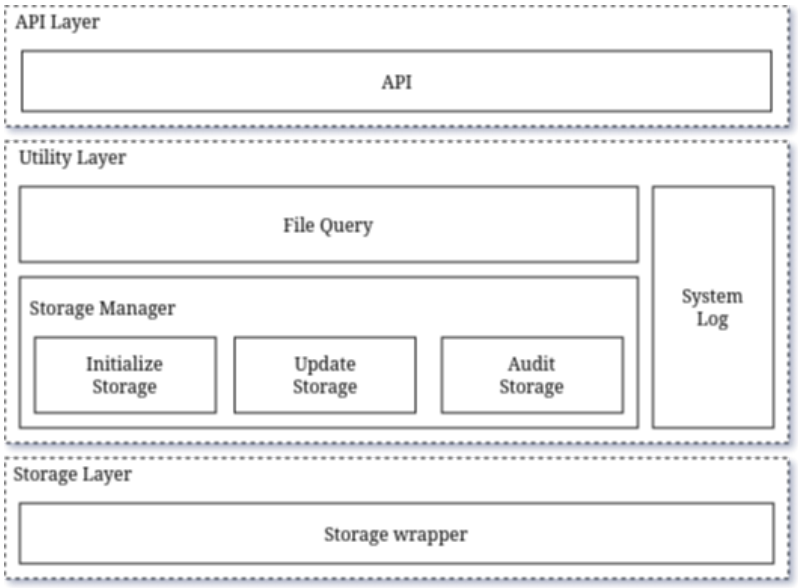

Figure 3: Logical Design Revision

In here, the API block providing the callable method allow end user to connect into utility of the system.

Then the system utility has 3 basic feature: “Storage Manager” handle the flow of data into store and verify their reliability. Then “File Query” block is used to discovery and capture user’s expected file from the storeage. Finally, the “System Log” works as a monitor system will evaluate the healthy status of storage and keep track the system activity.

The “Storage Wrapper” block has the role to simplify the storage access by hiding the complex attribute of a specific storage system and providing a easy and userful access function. Then the utility part should access into the store through “Storage Wrapper” to reduce the code complexity and improve maintainable attribute of system.