E-Healthcare Management System

Cheng-Han Hsieh, 謝承翰
B103040012

Shih Yu Sun, 孫世諭
B103040001

Casper Liu, 劉世文
B093040051

Tina Tsou, 鄒宜庭
B096060032

Chia-Yen Huang, 黃嘉彥
B103040051

Ting-Hao Hsu, 許廷豪
B103040008

December 24, 2023

1 Outline

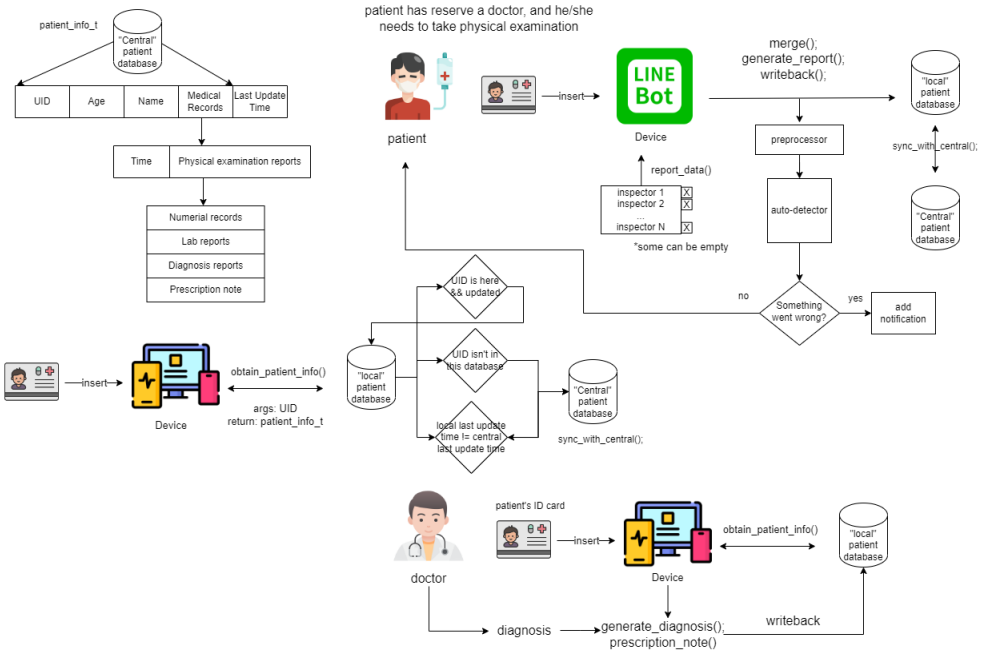

Figure 1: A simple example of outline.

2 Features

With this E-healthcare management system, the hospital/clinic can easily sync the information of patients with other health systems, manage the information of patients. For patients, the patient can do most of the things online, for example, make a doctor appointment, look up the medical records and prescriptions, and obtain the physical reports at home. Even more, the system use machine learning to detect the abnormal data in the reports, notify the patient to prevent the disease becoming worse. To conclusion, the major features of the system can be summarized as followings:

- Automatically sync the information of patients between different health systems by using local and central database.

- Facilitate the accessing of medical records and prescriptions, for both doctors and patients.
- Introduce the automatic disease detecters by leveraging machine learning and big data.

3 Methodology

Database

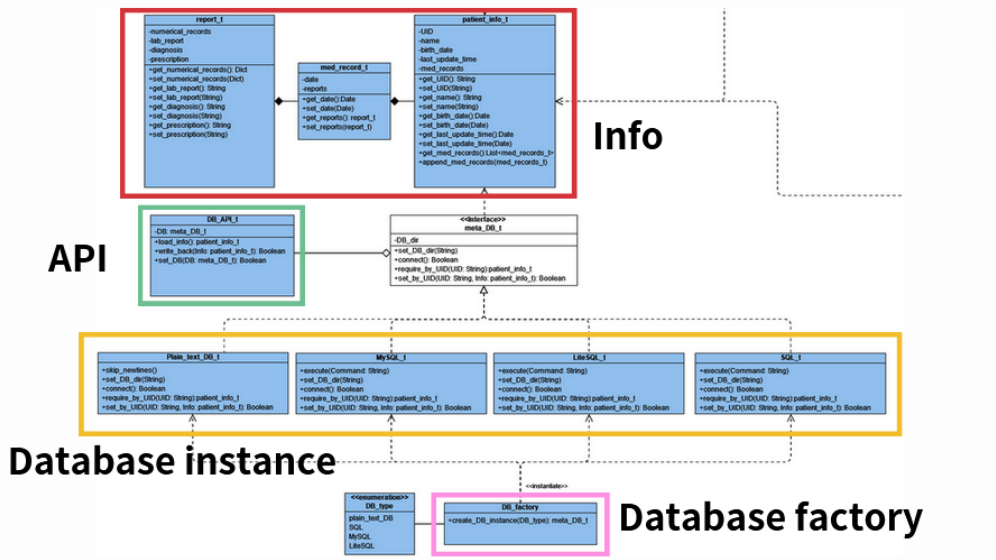

Figure 2: The UML of the whole database.

Figure 2 shows the UML of the database. It can be divided into four parts, the class definition of patients' information, the database API, the class definition of database, and the database factory.

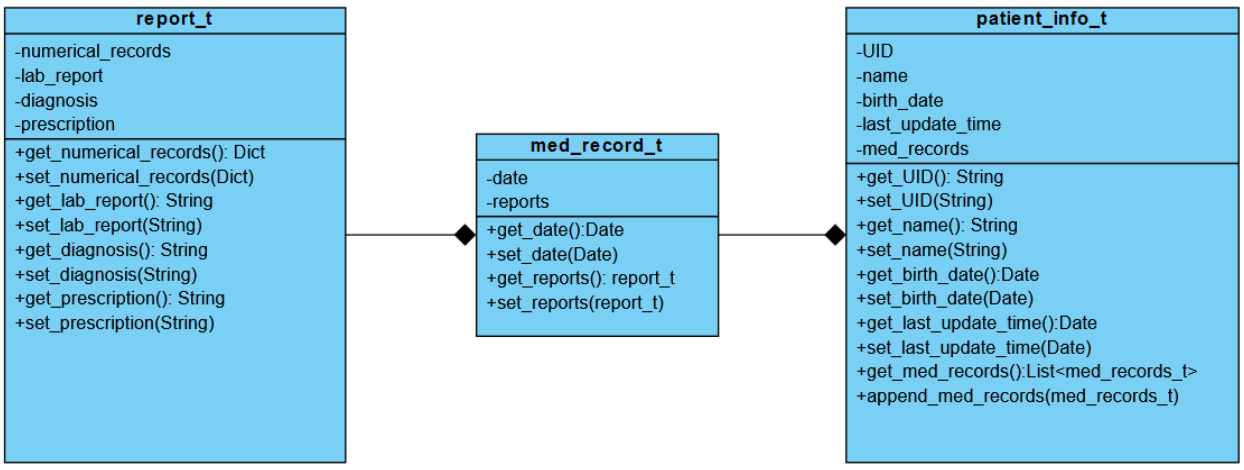

Figure 3: The UML of the class definition of patient information.

Figure 3 shows the class definition of the patients' information. The class, patient_info_t, contains the necessary information of a patient, for example, UID, name, birth date, and the list of medical records. And in the class, med_record_t, is date and reports, like diagnosis, prescriptions, and physical data and reports.

In this part, it provides a definition of patients' information for the whole database. Later, the database will depend on the class defined in this part.

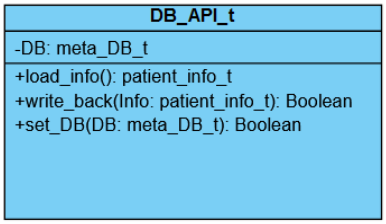

Figure 4: The UML of the class definition of database API.

Figure 4 is the definition of the API of the database. It provide some simple and secure methods to access the database, for example, load the information of a patient, update the information, and set the database. Later, those parts of the system that need to access the database rely on this API.

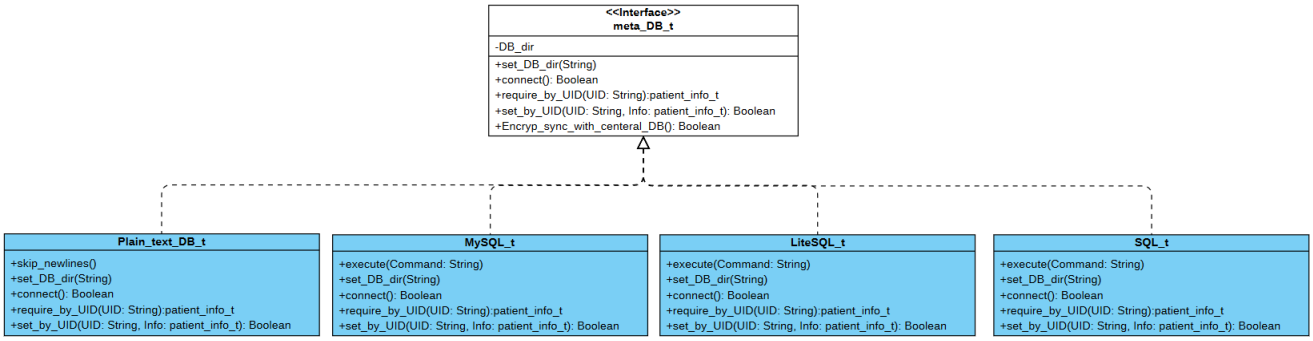

Figure 5: The UML of the class definition of database instance.

Here is the definition of the databases. First is an interface of the database, it defines some common methods of database, like connect, require, update, and sync with other database. Then are the specialized databases. Here, plain text database, SQLs are defined.

The rationales behind the need of an interface is, in the early development, we did not determine which type of database should be use. And second, when scaling the system up, the database may need to be replaced with other databases that have higher throughput and lower response time, like ScyllaDB. An interface of the database solves the problems, because it abstracts the database and unify the methods. The additional databases can be easily provided by following the interface.

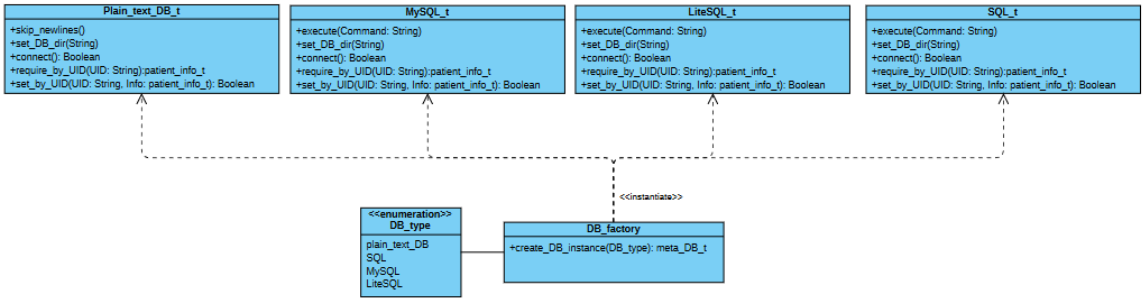

Figure 6: The UML of the factory of databases.

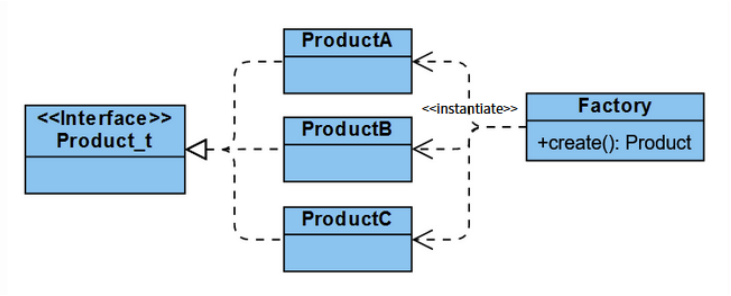

Figure 7: The UML of a simple factory.

4 Code

5 Conclusion

6 Contribution