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

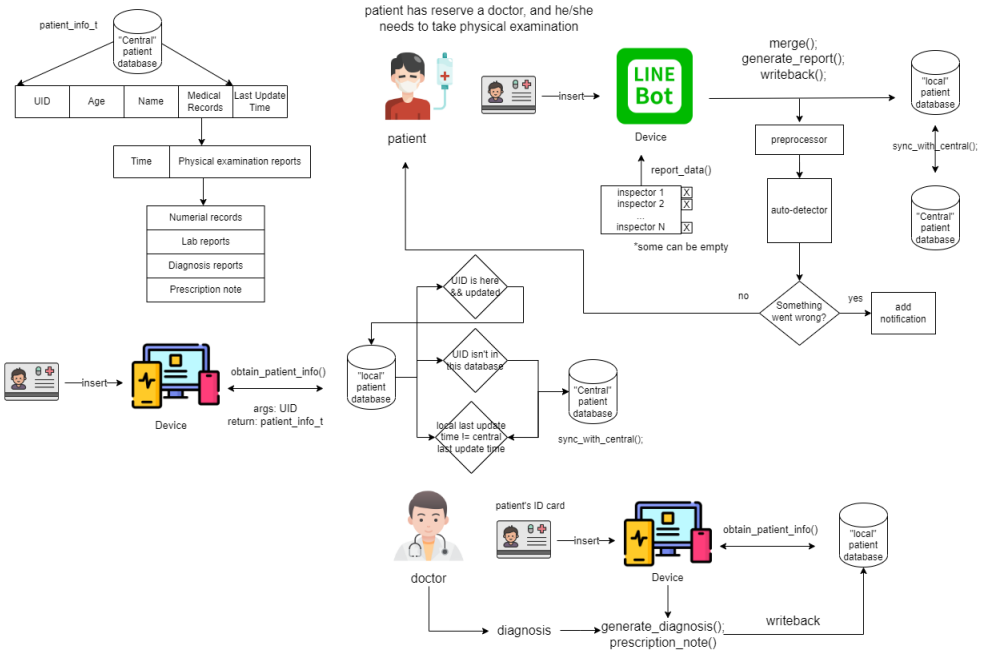

Figure 1: A simple example of outline.

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

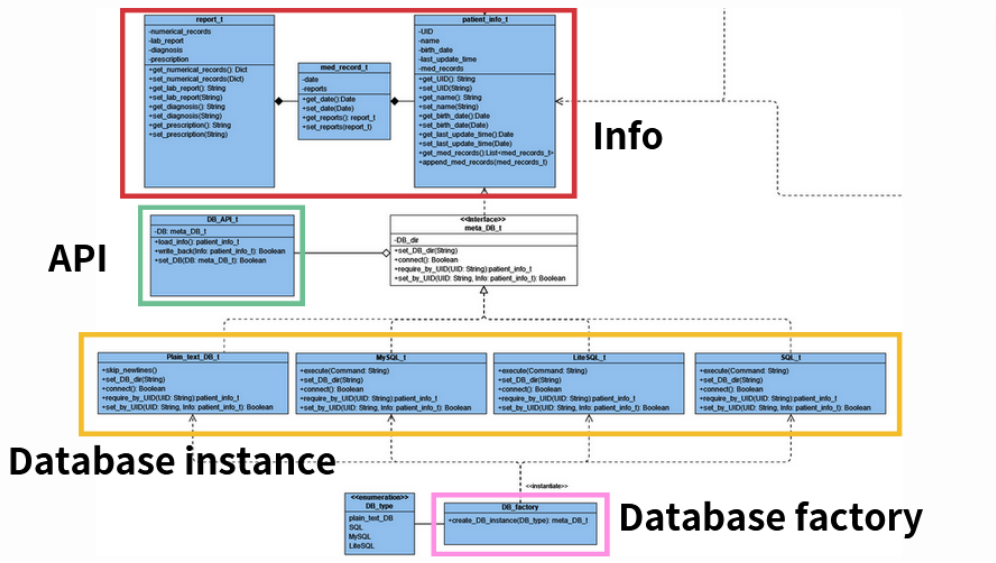

Figure 2: The UML of the whole database.

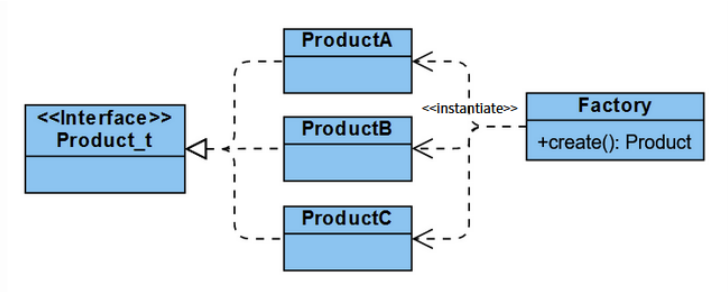

Figure 3: The UML of a simple factory.

Frontend and Medium

In the code for the patient frontend and middleware, our functionalities are as follows (refer to Figure 4):

1. Utilize Ngrok to forward external requests to the locally specified port.
2. Integrate LineBot, HTML, JS, and CSS, serving as the patient frontend to send requests such as registration, message sending, and reservations.
3. Connect to the database to retrieve detailed information based on the user's ID.
4. Interface with the Processor (Lab), sending detailed information retrieved from the database to the C++ Processor (Lab) for processing and receiving the information back.

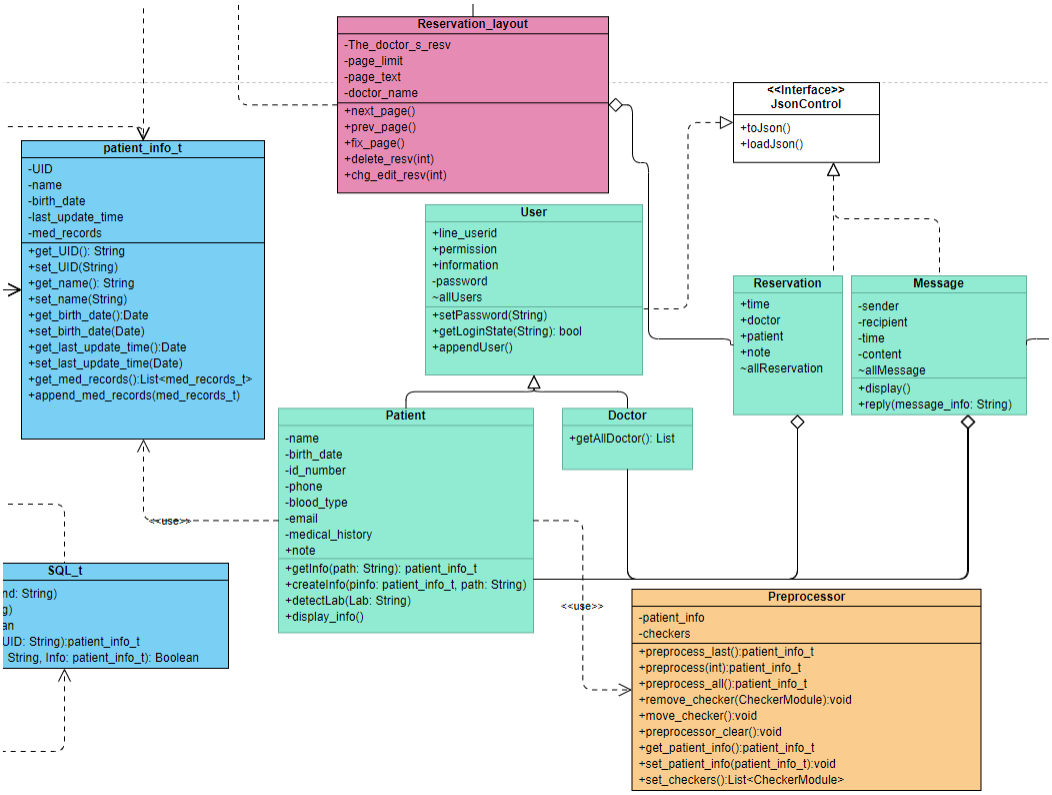

Figure 4: The UML of frontend and medium.

5. Enable the Doctor GUI to utilize the Patient Class to

   (i) obtain detailed information from the database,

   (ii) access return information from the Processor (Lab),

   (iii) retrieve the Message Class, and

   (iv) use the Reply() method to quickly respond to messages via LineBot (refer to Figure 4),

   (v) obtain detailed data from the Reservation Class,

   (vi) access the Doctor Class, and

   (vii) verify its name and password for login.

Next, we will explain each class in the frontend and medium, highlighting some special member functions and attributes:

User

- An abstract class primarily responsible for handling account information such as:

  (i) permission: Manages permissions, used to confirm the current user mode (admin or guest).

  (ii) line_id: Mainly used to record the user's Line ID, this information is automatically obtained from Line during registration and transmitted to our server.

  (iii) getLoginState(String): Takes a password as input and checks if it is the correct password.

Patient

- A child class of User, mainly deals with patient account information, where it interfaces with the patient_info_t in the database to obtain detailed data such as heart rate, blood glucose:

  (i) detectLab(String): Will send the information of patient_info_t to the Preprocessor, obtaining a detailed diagnosis such as "high heart rate," "diabetes risk," etc (refer to Figure 5).

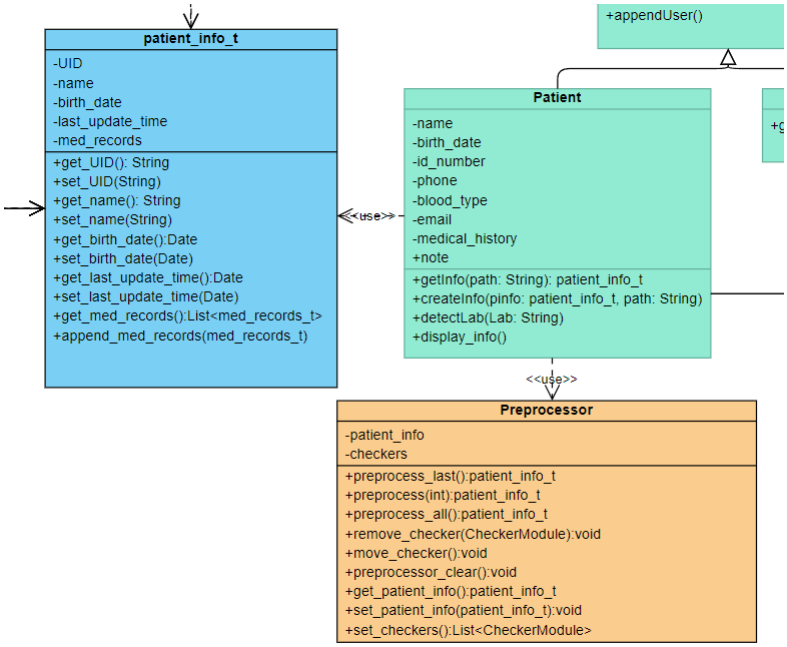

Figure 5: The UML of interaction between medium, preprocessor and database.

JsonControl

- An interface implemented by the User, Reservation, and Message classes. Its main purpose is to serialize objects into JSON files for easy storage and access:

  (i) toJson(): Stores all objects of the entire class in JSON format.

  (ii) loadJson(): Reads a JSON file and restores all objects from it into a list.

Reservation

- When a patient uses the frontend LineBot to make a reservation, a Reservation object is created. It is used and deleted by the Doctor GUI and includes Patient and Doctor objects to determine the patient and the reserved doctor (refer to Figure 6).

Message

- When a patient uses the frontend LineBot to send a message, a Message object is created. It is used, replied to, and deleted by the Doctor GUI and includes Patient and Doctor objects to determine the patient and the doctor being messaged (refer to Figure 6):

  (i) reply(): Based on its own Patient object, it uses LineBot to send a message back to the patient (because the Patient object contains the line_id attribute).

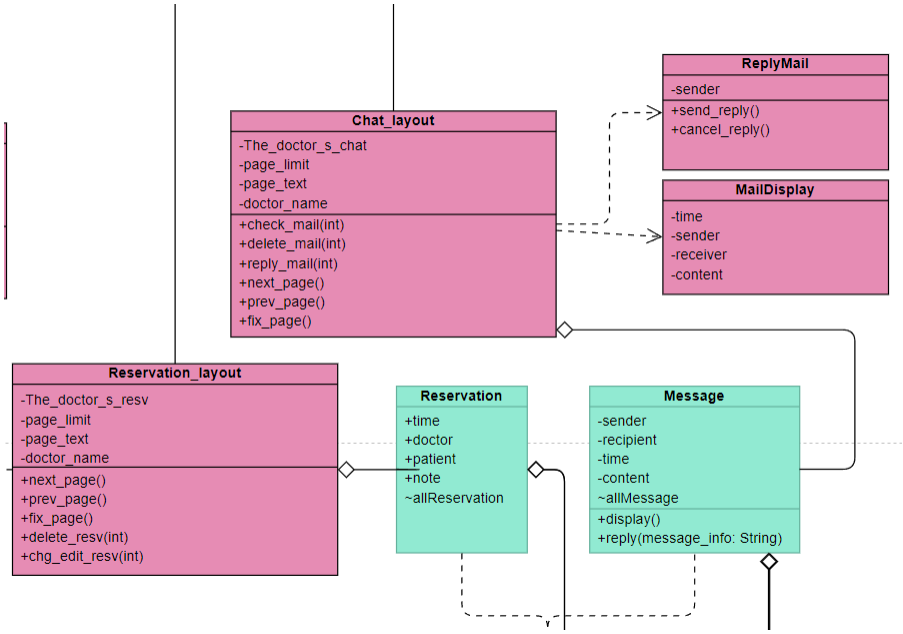

Figure 6: The UML of interaction between medium and Doctor GUI.

4 Code

5 Conclusion

6 Contribution