Matriculation number: 200009834

1. Software processes and modelling

(a):

Functional requirements:

1. User logging in and logging out. After logging in and logging out, users can be divided into consultants and doctors, and they can use the system. Among them, users with the same IP address can try to log in five times. If five times fail, log in will be prohibited one hour later, and a warning email will be sent to the user's mailbox to remind the user that the account is at risk. After a user logs in, if there is no operation for more than half an hour, the user is automatically logged out.
2. The user can write, read, and rewrite data, and the user can write new data, consult previously saved data, and rewrite past data. The data will not be stored locally, but all stored in the cloud, and the data is not allowed to be extracted in any form, but only allowed to be used in the system.
3. The system uses health services to associate with other systems. The system can be associated with other medical systems to facilitate the use of users. Data exchange needs to use encryption to prevent data leakage.

Non-functional requirements:

1. Software performance requirements: The system's computer load cannot exceed 1GB RAM, the user's storage space does not exceed 2GB, and the user's CPU demand cannot exceed i5-2400 or lower.
2. Software external interface requirements: the communication protocol uses common TCP/IP, the data format of the link to the external system needs to be encrypted, and the interaction frequency depends on the user's access.
3. Software design constraints: The maintainability of the system needs to ensure that shutdown maintenance cannot exceed once a month, and each time cannot exceed 24 hours. System reliability must ensure that the system can handle high concurrency without downtime due to user requests. System security must ensure that all data is stored securely and cannot be accessed from outside, and the user's authority classification needs to be very strict.

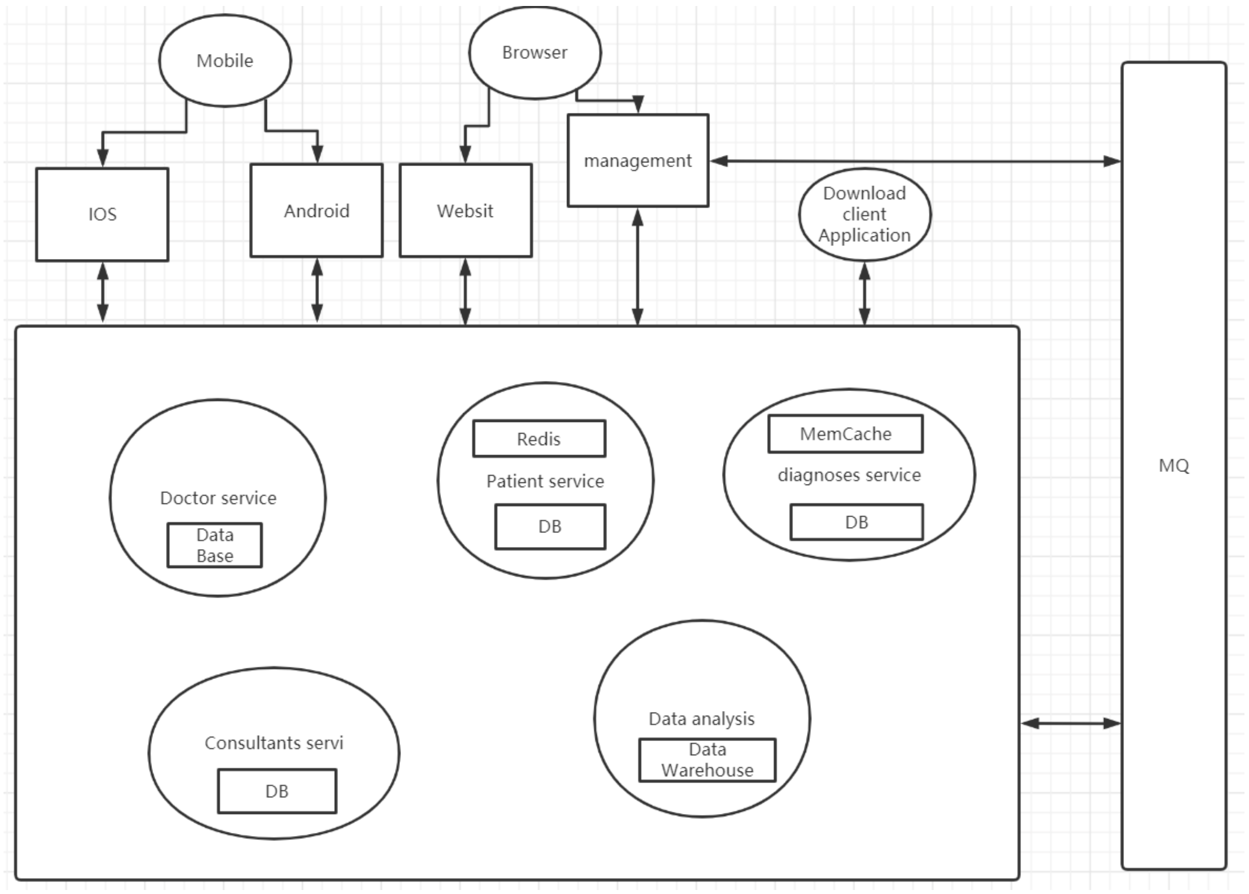
(b):

First of all, I decided to use the spiral model for the following reasons:

1. This is a large-scale project and is regulated by the government. The patient's personal information is very important, and risk assessment is required at all times.
2. The design needs to be flexible, because many requirements have to be changed along with government requirements.
3. Users need to participate in the project from time to time to ensure that the project does not deviate from the correct direction and the controllability of the project.
4. The user can always keep abreast of the latest information about the project, so that he or she can effectively interact with the management.

(c):

Since it is processing patient information, I consider the huge throughput required, so I choose a microservice architecture. The microservice architecture splits the application into multiple core functions. Each function is called a service and can be built and deployed separately, which means that the services will not affect each other when they work (and fail).



1. Technology heterogeneity
   The internal development technology of different services can be inconsistent. This system can use java to develop doctor services and Golang to develop consulting services. Choose the most suitable technology for different services. Different parts of the system can also use different storage technologies. For example, you can choose Redis storage for patient services and MySQL storage for diagnoses services.
2. Isolation
   The unavailability of one service will not cause another service to be paralyzed, because each service is an independent and autonomous system.
3. Scalability
   The system can only expand and upgrade those services that affect performance, avoiding overall modification under a single structure. For example, if the patient record needs to be upgraded, only the code of the patient service needs to be modified.
4. Simplify deployment
   In the microservice architecture, the deployment of each service is independent. If a problem really occurs, it only affects a single service, it can be quickly rolled back to solve the problem.
5. Easy to optimize
   The amount of code of a single service in the microservice architecture is not very large, so when the system needs to refactor or optimize this part of the service, the difficulty will be much less, because the smaller the amount of code, the more controllable the impact of code changes.

(d):

I think the most important thing in this software design process is security. Therefore, I chose the microservice framework on the previous question, because the data can be distributed independently, reducing the risk of data leakage, because all data in this project is sensitive data. modelling notations cannot be used for security. But the main aspects that need to be paid attention to are prevention of software piracy, software reverse engineering, authorized encryption, and illegal tampering, etc.

2. Software evolution and reuse

(a):

Software development, Software evolution, Software servicing

(b):

Software development: The manager proposes to develop a software that is easy to manage the club, and the software development begins.

Software evolution: The manager wants to add some new functions, such as membership.

Software servicing: When the program has a bug, or the program runs slowly.

(c):

1. Because it is a club, the budget is very limited. So, it's basically impossible to find more people to help development.
2. Since the development time is too long, many libraries may have been updated or abolished, so evolution is very difficult.
3. Since it was developed by two people, the documentation is not complete, which will cause a lot of trouble for follow-up updates. There are still many problems in the format of the data.

(d):

1. The size of the risk of developing new software. For this, in fact, the risk of developing new software is very small, and it is originally a program written by two people. Lack of documentation, there is not much maintenance value. There are not enough experts to add new features.
2. In order to reduce the budget, for this, if you want to increase the size of the club, then the new system is very necessary, because of the increase in the number of people and increase the efficiency of the staff. The cost of data transfer here is not very high.
3. The quality of the code may not be particularly high. The complexity of the code. In many cases, the legibility of the code is low and the complexity is too high, and it is no longer suitable for continued use. At this time, there is no value to add new functions. Even the original developers who add new features may not understand the code.

(e):

1. If the readability of the code is high enough and the complexity of the code is not too high, then it is possible to directly use the original system.
2. The reuse of function, for example, the component responsible for the function of race recording can be directly used on the new system, so there is no need to develop a new one.
3. The design pattern of the software is very good, so it can be used directly without rebuilding a new design pattern.
4. The framework of the program is very good; developer can use the original framework

directly.

(f):

First of all, we have a new system and build a product line based on this system. First, extract the core components and manage the members of the club. Configurable application components are some that may be replaced. I don't think there are any here. Finally, Specialised application components are organise coaching sessions and races.

Based on the above product line, I think the game planning system and the athlete training recording system can use this production line.

3. Software quality

(a):

1. Availability
   Customers can send life information by the watch at any time. The camera can transmit information at critical moments. The customer's watch will remind he or she of appointments and medication time. Customers can send emergency or daily contact via watch or mobile phone.
2. Reliability
   The life information of customers can be accurately received by medical staff. The camera can accurately transmit images, the appointment and medication time transmitted by the system are accurate, and the emergency contact or daily contact link is accurate. Some accidents can be accurately captured by the system.
3. Safety
   Multiple devices supporting customers, as well as a large number of devices for medical staff, require the stable operation of the server without downtime.
4. Security
   The personal life information of customers must be stored securely and cannot be captured by outsiders.
5. Resilience
   When the server is attacked, the backup server can be quickly started and repaired in a short period of time, because an emergency may occur during the downtime.

(b):

1. The user may forget to wear the watch, so that the user’s vital information cannot be monitored
2. The customer should not initiate daily contact, because the customer is the elderly, and the

elderly are easy to forget.

(c):

1. When the watch cannot receive user information, the watch will always remind that if it has not received the user's life information for more than fifteen minutes, the watch will send the information directly to the medical staff.

2. Establish a prompt window on the client of the medical staff to remind the medical staff to take the initiative to contact the customer.

(d):

1. User testing:
The vast majority of developers should not be elderly, so in many cases they may be incomplete, so it is necessary to involve users and make suggestions that only users can use.

2. Performance testing
Because of the need to avoid the risk of server downtime, developers cannot predict whether an elderly person is in danger during the downtime.

3. Use case testing
It is necessary to make sure that all user cases can be implemented, because these cases are very important and are related to whether the life information of the elderly can be observed in a timely manner, and the physical condition of the elderly can be regularly checked to prevent risks.