# **Software Requirements**

Version 4, last updated by Jarvis at 2021-04-27

# **Software Requirements Specification (SRS)**

Revision History:

|  |  |  |
| --- | --- | --- |
| Date | Author | Description |
| 4-12-2021 | Jarvis Liu | ·Finish the background writing  ·Describe the system purpose briefly |
| 4-12-2021 | Lily Zhang | ·Complete the algorithm and customer’s requirement analysis  ·Finish the function requirement |
| 4-12-2021 | Egbert Ding | · Complete the Security and Flexibility module  ·Complete the Others module |
| 4-12-2021 | Bryce Shi | ·Write the performance requirement  ·Describe the data management capability |
| 4-15-2021 | Jarvis Liu | ·Introduce the intended audience  ·Elaborate the offered services  ·illustrate the concept of operations |
| 4-16-2021 | Bryce Shi | · Finish the behavioral requirements part  ·Complete the input and output requirements |
| 4-17-2021 | Egbert Ding | · Finish the quality requirements and    fundamental assumption |
| 4-17-2021 | Gabriel Carneiro | · Insert use cases |
| 4-18-2021 | Ana Borges | · Insert use cases |
| 4-18-2021 | Lily Zhang | · Revise use cases |
| 4-22-2021 | Jarvis Liu | ·RA document fixed |
| 4-27-2021 | Jarvis Liu | ·Modify all the offered services due to a  huge demand change after the final inquiry  on Thursday's class. |
| 4-27-2021 | Bryce Shi | ·Modify the System Inputs and Outputs  due to a huge demand change after the final inquiry on Thursday's class. |
| 4-27-2021 | Egbert Ding | ·Modify some use cases due to a  huge demand change after the final inquiry  on Thursday's class. |
| 4-27-2021 | Lily Zhang | ·Modify and add some use cases |
|  |  |  |

Contents

Software Requirements

Software Requirements Specification (SRS)

1. Introduction

1.1 Intended Audience and Offered Services

1.2 How to use the document

2.  Concept of Operations

2.1 System Context

2.2 System capabilities

3.  Use Cases

Case 1:  Application Server Register new User

Case 2: Application Server upload new x-ray

Case 3: Application Server Retrieve User’s Data by time

Case 4: Administrator Delete User’s Data

Case 5: User wants to unsubscribe the system

Case 6: User wants to retrieve the result of diagnosis

Case 7: Store the Analysis Result from algorithm

Case 8: Upload the Analysis Result from doctor

Case 9:Administrator add news

Case 10:Administrator delete news

Case 11:Administrator upload news

Case 12:Administrator search news by date

Case 13:Patient send a message to the doctor

Case 14:Doctor send a message to the patient

Case 15:Administrator wants to delete a message

Case 16:Doctor wants to search message with specific patient by time

Case 17:Patient wants to search message with specific doctor by time

Case 18:Doctor wants to search last n message with specific patient

Case 19:Patient wants to search last n message with specific doctor

Case 20:Administrator wants to add a reply record

Case 21:Administrator wants to delete a reply record

Case 22:Administrator wants to upload a reply record

Case 23:User wants to get a reply

4.Behavioral Requirements

4.1 System Inputs and Outputs

4.1.1 Inputs

4.1.2 Outputs

4.2 Detailed Output Behavior

5. Quality Requirements

6. Expected Subsets

7. Fundamental Assumptions

8. Expected Changes

9. Appendices

9.1 Acronyms and abbreviations

9.2 References

## **1.  Introduction**

### **1.1    Intended Audience and Offered Services**

This document is designed to provide information guiding the operation on the database, guaranteeing that all system demands are understood and fulfilled. The server team may refer to this document to render mentioned information services to the algorithm and client team.

Services for the algorithm team:

· Customer’s Data - The server team should send the x-ray film uploaded by the customer to the algorithm team by calling “insert\_record”in  Patient Record Table.

· Analysis Result - The server team should help store the data results gotten by the algorithm team into the Patient Record Table in the database, including the upload\_record\_id, user\_id, url, algorithm\_result, detect\_data, checked\_by\_doctor, doctor\_id, doctor\_result.(The reason why we need check\_by\_doctor is that the doctor may modify the prediction made by the algorithm team so we have to check whether the picture is updated or not.)

Services for all the client teams:

Sharing functions:

· Register - When a new user registers for the diagnosis, the server should store his/her information into the User Table by calling “insert\_record”. The information consists of user\_id, name, birthdate, phone\_number, email\_address, credential, register\_data, user\_password, account\_type.

· Upload Data - When the patient uploads the x-ray film,the server team should send the data(including user\_id and the picture) to the database immediately by calling “insert\_record”in  Patient Record Table.

· Search&Request - When the client is intended to retrieve User's Data, the server should send relevant data in the Use Table and Patient Record Table to him/her, which is calling “select\_record” by detect\_date.

· Unsubscribe - When the client choose to unsubscribe from this service, the server should delete all the relevant information from the User Table, Patient Record Table, Message Table by operating “delete\_record”on all the lines that matches the corresponding user\_id.

· Update Picture - When the doctor modifies the picture returned by the algorithm team, the server should change the specific picture data of the patient by calling "update\_record" in Patient Record Table.

· Edit Account - When the client wants to edit account, the server should change certain information in User Table by calling"updata\_record" based on user\_id.

Wechat only:

· Consultation - our team conserve all the Messages between patientes and doctors, including id、patient\_id(user\_id)、 doctor\_id(user\_id)、upload\_record\_id in the Message Table and when the patient/doctor is trying to find the chat record, the server team can get the required messages such as returning the latest ten messages or messages in certain time period.

Android only:

· News Record - Data including news\_id,news\_content and news\_time is stored in the database.The former news would be deleted based on "news\_time" and other news would be added into News Table by calling "insert\_record".Also,the server should send relevant news to the patients based on their demands.

Chatbot only:

· Automatic Response - When the patient comes for health inquiry, the server team should match the questions with the keyword stored in the Chatbot Table and gives automatic response by calling "select\_record" based on the keyword.

### **1.2    How to use the document**

Table of Contents:

1. Introduction – Guide the server team to  render mentioned services related to the constructed database to the algorithm and client team.

2. Concept of Operations - Descriptions about the faced problem and the solution to be implemented.

2.1 System Context - System requirements the application will require to run for diverse operating systems.

2.2 System Capabilities – Overview of all capabilities(functions)  available to the server team.

2.3 Use cases - A detailed look at each functional requirement, describing the application context both before and after an operation is taken.

3. Behavioral Requirements - How will the application interact with a user.

 3.1 Input and output requirements – A description of allowed inputs and generated outputs.

3.1.1 Input - Describes the content of input and any restrictions that will be placed on allowed input.

3.1.2 Output - Describes the content of output and the range of outputs that can be generated.

  3.2 Detailed Output Behavior - Output descriptions in prose.

4. Quality Requirements - Requirements not pertaining to the function of the application will be listed here.

5. Expected Subsets - Expected levels of functionality at checkpoints during development.

6. Fundamental Assumptions - Some specifics about input, output, or behavior upon which other requirements are founded will be listed here.

7. Expected Changes - Future features and directions the project is expected to take.

8. Appendices - Details aiding the understanding of this document

8.1 Definitions and acronyms - Any technical terms or abbreviations will be spelled out here for ease while using the document

 8.1.1 Definitions - Definitions of technical or unusual terminology.

8.1.2 Acronyms and Abbreviations - Any abbreviated terms will be expanded here.

8.2 References - any external references necessary or helpful to understanding this document will be listed here.

## **2.  Concept of Operations**

In the process of scoliosis diagnosis, it is necessary to store a large amount of patient information and pictures of the spine. The traditional manual operation method is prone to data loss, statistical errors and slow speed.

Our purpose is to create five tables in the constructed MYSQL database to store the relevant information of patients and the data results gotten by the algorithm team to realize data sharing and centralized control. And the server team is able to render mentioned services(the kernel of the operation is about insertion, deletion ,update and selection) to the algorithm and client team.

### **2.1 System Context**

System Requirements:

Support Windows/Mac OS X/Linux. The following runtime environments need to be installed and configured in advance. This system mainly uses java/python as the back-end application development tool, MYSQL as the back-end database and Windows as the system platform.

Windows:

Windows 10/8/7, Python 3.0+, Pytorch1.1.0+, OpenCV-Python

### **2.2 System capabilities**

After finishing the database construction, the server can fulfill the requirements raised by the two algorithm teams including sending the X-ray film that customer uploads and storing the analysis result into the database, and the needs of four client teams including Register, Upload Data, Search & Request , Unsubscribe, update Picture, Edit Account, Consultation, News Record, Automatic Response.

## **3. Use Cases**

|  |
| --- |
| **Use Case 1:** Application Server Registers  a new User |
| **Brief description:**  Application server requests the database for the registration of one user(patient/doctor/administrator) to the platform. |
| **Primary Actors:**  Application Server |
| **Preconditions:**  The Data Access Object service (DAO) and the DBMS are running and the Application Server is authorized to communicate with the DAO.  The new user have already offered the required information |
| **Main flow:**  1. The Application Server requests the DAO for the registration of a new user, sending the authorization token and the following information about the user that it wants to register: name, birth-date, phone\_number, email\_address, credential, register\_date, user\_password,account\_type;  2. The DAO verifies if the user haven't been recorded;  3. If the user hasn't been recorded   1. the DAO generate an ID for the user 2. the DAO registers the new user in the User Table, returning the confirmation of registration and the new user identification in the platform;   4. Else  a.  the DAO returns a error message, with a unauthorized error; |
| **Post conditions:**  The new user(patient/doctor/administrator) is registered in DBMS. |
| **Alternative flows:**  None. |

|  |
| --- |
| **Use Case 2:** Application Server uploads a new x-ray film |
| **Brief description:**  Application server stores a new X-ray film in the database. |
| **Primary Actors:**  Application Server |
| **Preconditions:**  The Data Access Object service (DAO) and the DBMS are running and the Application Server is authorized to communicate with the DAO.  The owner of the X-ray has registered an account. |
| **Main flow:**  1. The Application Server receive the photo uploaded by the user and generate its URL.  2. The DAO verifies if the Application Server is authorized;  3. If the Application Server is Authorized   1. The DAO create a record a record for the X-ray(It can be uploaded by a doctor or patient),generate an ID for the upload\_record 2. b.b.store its URL and related information(owner, detect\_date) in the Patient Record table 3. return the confirmation of the x-ray store in the database;   4. Else  a.  the DAO returns a error message, with a unauthorized error; |
| **Post conditions:**  1. The x-ray and related information are registered in DBMS. |
| **Alternative flows:**  None. |

|  |
| --- |
| **Use Case 3:** Application Server Retrieves User’s Data by time |
| **Brief description:**  Application server requests results for a user in a determined period of time. |
| **Primary Actors:**  Application Server |
| **Preconditions:**  The Data Access Object service (DAO) and the DBMS are running and the Application Server is authorized to communicate with the DAO.  The user(doctor/administrator/patient) wants to retrieve User’s Data |
| **Main flow:**  1. The Application Server requests the DAO for the results of a patient, sending the follow information: user’s note identification, start date and end date of the period of request;  2. The DAO verifies if the Application Server is authorized;  3. If the Application Server is Authorized  a. the DAO responses with the images of x-ray for the patient and the prediction about the x-rays uploaded in the period sent by the Application Server existent in database;  4. Else  a.  the DAO returns an error message, with a unauthorized error; |
| **Post conditions:**  1. The x-rays and predictions are returned to the Application Server. |
| **Alternative flows:**  None. |

|  |
| --- |
| **Use Case 4:** Application Server Deletes user’s specific data |
| **Brief description:**  Application Server deletes relevant records according to the provided record\_id |
| **Primary Actors:**  Application Server |
| **Preconditions:**  The Data Access Object service (DAO) and the DBMS are running and the Application Server is authorized to communicate with the DAO. |
| **Main flow:**  1. The Application Server requests the DAO for deleting specific record about one specified user, sending the authorization token and the patient note identification;  2. The DAO verifies if the Application Server is authorized;  3. If the Application Server is Authorized  a. the DAO deletes the record related to the patient;  4. Else  a.  the DAO returns an error message, with an unauthorized error; |
| **Post conditions:**  1. the specific data related to the specified user is deleted from the database. |
| **Alternative flows:**  None. |

|  |
| --- |
| **Use Case 5:** User wants to unsubscribe the system |
| **Brief description:**  Application Server deletes user’s account, deleting all the data related to specified patient from the DBMS. |
| **Primary actors:**  User |
| **Second actors:**  Application Server |
| **Preconditions:**  Relevant data has been obtained from hospitals or other channels.  The server is available with all the data stored in the database. |
| **Main flow:**  1. The Application Server requests the DAO for deleting specific user, sending the authorization token and the patient note identification;  2. The DAO verifies if the Application Server is authorized;  3. If the Application Server is Authorized   1. the DAO deletes all the record related to the patient in the Patient Record Table 2. the DAO deletes the user record according to the provided user id in the User Table   4. Else  a.  the DAO returns an error message, with an unauthorized error; |
| **Post conditions:**  1. all the data related to the specified user is deleted from the database. |
| **Alternative flows:**  None. |

|  |
| --- |
| **Use Case 6:** User wants to retrieve the result of diagnosis |
| **Brief description:**  The diagnosis result is returned to the user according to the id of the X-ray. |
| **Primary actors:**  User(patient/doctor/administrator) |
| **Second actors:**  Application Server |
| **Preconditions:**   1. the customer must be registered on the platform. 2. the Application Server is Authorized |
| **Main flow:**  1.Find the relevant records in the Patient Record Table according to the provided ID  2.If the Application Server find records meets the criteria  a.The diagnosis result is returned to the user according to the user's requirement  3.Else  a.  the DAO returns an error message, with an unfounded error; |
| **Post conditions:**  1. The user gets the result for the specific X-ray |
| **Alternative flows:**  None. |

|  |
| --- |
| **Use Case 7:** Store the Analysis Result from the algorithm |
| **Brief description:**  The new result predicted by the algorithm is stored in the database. |
| **Primary actors:**  Algorithm. |
| **Second actors:**  Application Server. |
| **Preconditions:**  The algorithm works well with the input data |
| **Main flow:**  1.The use case begins when the algorithm do the prediction.  2.If the prediction was executed successfully       a.The algorithm sends a successfully message and sends the result to the server in order to store it in the database.  b.According to the id of the X-ray, the predicted result (including picture and data) is stored in the corresponding position of the Patient Record table  3.Else        a.The algorithm returns an error message. |
| **Post conditions:**  1.The prediction is stored in the database, which is available for consulting. |
| **Alternative flows:**  None. |

|  |
| --- |
| **Use Case 8:** Upload the Analysis Result from doctor |
| **Brief description:**  Update the doctor's modified pictures and data results to the corresponding position according to the record\_id. |
| **Primary actors:**  Doctor |
| **Second actors:**  Application Server |
| **Preconditions:**  the result of the algorithm has been saved in specific customer’s record. |
| **Main flow:**  1.Doctors get the images and results predicted by the algorithm, and make judgments and corrections  2.If the result predicted by the algorithm needs to be changed        a.Doctors upload revised images and data  b.According to the id of the X-ray, the revised result (including picture and data) is updated in the corresponding position of the Patient Record table  c.Identify that the doctor has modified the record, and record the ID of the doctor  3.Else        a.The original result will not be modified |
| **Post conditions:**  1.The revised result is uploaded in the database, which is available for consulting. |
| **Alternative flows:**  None. |

|  |
| --- |
| **Use Case 9:** Administrator adds news |
| **Brief description:**  The administrator adds the new news content of today to the database |
| **Primary actors:**  Administrator |
| **Second actors:**  Application Server |
| **Preconditions:**  The Data Access Object service (DAO) and the DBMS are running and the Application Server is authorized to communicate with the DAO.  The administrator has registered an account. |
| **Main flow:**  1. The Application Server requests the DAO for the addition of a piece of news, sending the related information and the content of the news  2. The DAO verifies if the news haven't been recorded;  3. If the user haven't been recorded  a.  the DAO generate an ID for the news  b. the DAO add a new record in the News Table(Including news ID, news date and news content), returning the confirmation of addition;  4. Else  a.  the DAO returns a error message, with an unauthorized error; |
| **Post conditions:**  1.The news is stored in the database, which is available for acquisition |
| **Alternative flows:**  None. |

|  |
| --- |
| **Use Case 10:** Administrator deletes news |
| **Brief description:**  Application Server deletes relevant news according to the provided news title |
| **Primary actors:**  Administrator |
| **Second actors:**  Application Server |
| **Preconditions:**  The Data Access Object service (DAO) and the DBMS are running and the Application Server is authorized to communicate with the DAO.  The administrator has registered an account. |
| **Main flow:**  1. The Application Server requests the DAO for the deletion of a piece of news, sending the related news title  2. The DAO verifies if the news has been recorded;  3. If the user has been recorded  a.  the DAO delete the record  b. return the confirmation of deletion;  4. Else  a.  the DAO returns a error message, with an unauthorized error; |
| **Post conditions:**  1.The news is deleted from the database, which is unavailable for acquisition |
| **Alternative flows:**  None. |

|  |
| --- |
| **Use Case 11:** Administrator uploads news |
| **Brief description:**  Application Server uploads relevant records according to the provided record id |
| **Primary actors:**  Administrator |
| **Second actors:**  Application Server |
| **Preconditions:**  The Data Access Object service (DAO) and the DBMS are running and the Application Server is authorized to communicate with the DAO.  The administrator has registered an account. |
| **Main flow:**  1. The Application Server requests the DAO for the revision of a piece of news, sending the related information and the content of the news  2. The DAO verifies if the news has been recorded;  3. If the user has been recorded  a.  the DAO refresh the news  b. return the confirmation of update;  4. Else  a.  the DAO returns a error message, with an unauthorized error; |
| **Post conditions:**  1.The prediction is updated in the database, which is available for consulting. |
| **Alternative flows:**  None. |

|  |
| --- |
| **Use Case 12:** Administrator searches news by date |
| **Brief description:**  Search all eligible news by date and form a list to return |
| **Primary actors:**  Administrator |
| **Second actors:**  Application Server |
| **Preconditions:**  The Data Access Object service (DAO) and the DBMS are running and the Application Server is authorized to communicate with the DAO.  The administrator has registered an account. |
| **Main flow:**  1. The Application Server requests the DAO for the search of pieces of news, sending the related date  2. The DAO verifies if there is news of the day  3. If there is news of the day  a.  the DAO generate an list for all the news of the date  b. returning the list and the confirmation of research;  4. Else  a.  the DAO returns a error message, with an unfound error; |
| **Post conditions:**  Administrators can get all news contents for the corresponding date |
| **Alternative flows:**  None. |

|  |
| --- |
| **Use Case 13:** Patient sends a message to the doctor |
| **Brief description:**  The patient communicates with the doctor. |
| **Primary actors:**  Customer. |
| **Second actors:**  Application Server. |
| **Preconditions:**  The patient sends a message. |
| **Main flow:**  1.The use case begins when the customer send a message.  2.The server sends the information of this message.  3.The information of the message including itself will be saved in the database.  4.Update the Unread attribute of all the messages the customer received before. |
| **Post conditions:**  The message will be saved in the message table. |
| **Alternative flows:**  None. |

|  |
| --- |
| **Use Case 14:** Doctor sends a message to the patient |
| **Brief description:**  The doctor communicates with the patient. |
| **Primary actors:**  Customer. |
| **Second actors:**  Application Sever. |
| **Preconditions:**  The doctor sends a message. |
| **Main flow:**  1.The use case begins when the customer send a message.  2.The server sends the information of this message.  3.The information of the message including itself will be saved in the database.  4.Update the Unread attribute of all the messages the customer received before. |
| **Post conditions:**  The message will be saved in the message table. |
| **Alternative flows:**  None. |

|  |
| --- |
| **Use Case 15:** Administrator wants to delete a message |
| **Brief description:**  The administrator deletes a message. |
| **Primary actors:**  Administrator |
| **Second actors:**  Application Sever. |
| **Preconditions:**  Administrator appoints a message. |
| **Main flow:**  1.The administrator appoints the message to be deleted.  2.The server sends the id of the message.  3.Delete the message according the id. |
| **Post conditions:**  The message will be saved deleted. |
| **Alternative flows:**  None. |

|  |
| --- |
| **Use Case 16:** Doctor wants to search message with specific patient by time |
| **Brief description:**  Doctor gets messages with specific patient by time. |
| **Primary actors:**  Doctor |
| **Second actors:**  Application server |
| **Preconditions:**  The doctor appoints the patient. |
| **Main flow:**  1.The doctor appoints the messages with the patient to search.  2.The server sends the patient id.  3.Query in the message table according to the patient id.  4.Return the qualified message by time to the server. |
| **Post conditions:**  The server gets the qualified message. |
| **Alternative flows:**  None. |

|  |
| --- |
| **Use Case 17:** Patient wants to search message with specific doctor by time |
| **Brief description:**  Patient gets messages with specific doctor by time. |
| **Primary actors:**  Patient |
| **Second actors:**  Application server |
| **Preconditions:**  The patient appoints the doctor. |
| **Main flow:**  1.The patient appoints the messages of the doctor to search.  2.The server send the doctor id.  3.Query in the message table according to the doctor id.  4.Return the qualified message by time to the server. |
| **Post conditions:**  The server gets the qualified message. |
| **Alternative flows:**  None. |

|  |
| --- |
| **Use Case 18:** Doctor wants to search last n message with specific patient |
| **Brief description:**  Doctor searches the last n message. |
| **Primary actors:**  Doctor |
| **Second actors:**  Application server |
| **Preconditions:**  The patient appoints the patient. |
| **Main flow:**  1.The doctor appoints the messages of the patient to search.  2.The server sends the patient id.  3.Query in the message table according to the patient id.  4.Return the last n qualified message to the server. |
| **Post conditions:**  The server gets the qualified message. |
| **Alternative flows:**  None. |

|  |
| --- |
| **Use Case 19:** Patient wants to search last n message with specific doctor |
| **Brief description:**  Patient searches the last n message. |
| **Primary actors:**  Patient |
| **Second actors:**  Application server |
| **Preconditions:**  The patient appoints the doctor. |
| **Main flow:**  1.The patient appoints the messages of the doctor to search.  2.The server sends the doctor id.  3.Query in the message table according to the doctor id.  4.Return the last n qualified message to the server. |
| **Post conditions:**  The server gets the qualified message. |
| **Alternative flows:**  None. |

|  |
| --- |
| **Use Case 20:** Administrator wants to add a reply record |
| **Brief description:**  Administrator adds a record. |
| **Primary actors:**  Administrator |
| **Second actors:**  Application server |
| **Preconditions:**  The administrator provides a suitable reply record. |
| **Main flow:**  1.The administrator provides a reply record.  2.The server sends the reply record.  3.Add a record in the patient record table. |
| **Post conditions:**  A new record is added. |
| **Alternative flows:**  None. |

|  |
| --- |
| **Use Case 21:** Administrator wants to delete a reply record |
| **Brief description:**  Administrator deletes a record. |
| **Primary actors:**  Administrator |
| **Second actors:**  Application server |
| **Preconditions:**  The administrator appoints a record to delete. |
| **Main flow:**  1.The administrator appoints a record.  2.The server sends the record id.  3.Delete the record in the patient record table according to the record id. |
| **Post conditions:**  A reply record is deleted. |
| **Alternative flows:**  None. |

|  |
| --- |
| **Use Case 22:** Administrator wants to upload a reply record |
| **Brief description:**  Administrator uploads a record. |
| **Primary actors:**  Administrator |
| **Second actors:**  Application server |
| **Preconditions:**  The administrator provides a suitable record. |
| **Main flow:**  1.The administrator uploads a reply record.  2.The server sends the record.  3.Upload the record in the patient record table. |
| **Post conditions:**  A reply record is uploaded. |
| **Alternative flows:**  None. |

|  |
| --- |
| **Use Case 23:** User wants to get a reply. |
| **Brief description:**  User wants to get a reply. |
| **Primary actors:**  User |
| **Second actors:**  Application server |
| **Preconditions:**  The user provides a reply. |
| **Main flow:**  1.The user provides a record to get.  2.The server sends the record id.  3.Query the record in the patient record table according to the id.  4.Return the qualified reply. |
| **Post conditions:**  The server gets the qualified reply. |
| **Alternative flows:**  None. |

## **\*4. Behavioral Requirements**

### **4.1 System Inputs and Outputs**

#### **4.1.1 Inputs**

          The database inputs come from the users(inputs from patients and doctors are dealt by the server ). The users(admin) can create, read, update, delete data in the database. The inputs include the train and test data , the data inputted by doctor/patient ,the result returned from the algorithm.

Inputs When Client Use The Database:

Register: When a new user registers for the diagnosis

      \*General: At least two inputs will be required for a new user. One of those two inputs must be a User\_id. There is no length limit on any of the fields below.

     \*User\_ID:\*User\_ID: The user\_id is the primary key of user table. This is the identity of user, which means every user has a unique user\_id;

 \* Name: A user can have a first name, or last name, or both. The first and last names will be separate fields. The system will accept any characters in the first and last name fields.

      \* Gender: Gender should be male or female .

      \* Birth-date: Birth-date that is accepted as valid by the system must be digits. The only acceptable inputs for a valid input is numerical characters ranging 0-9. If characters is not a digit, it will raise an error.

\*Phone Number: Phone number that is accepted as valid by the system must be digits. The only acceptable inputs for a valid input is numerical characters ranging 0-9. If characters is not a digit, it will raise an error.

\* Email Address: The email address must obey the email address format strictly.

\*Register Date: The form is as same as birthdate.

\*User\_Password: Could be any characters except Enter and Space key.

\*Account Type: Patient，Doctor or Admin.

Upload Data:

        \*X-ray Image: The X-ray image will be saved as the form of url.Server input the url of the X-ray image ,the url will be insert to the user's table.

\*Upload\_Record\_ID: Differentiate the different upload, should be digits.

\* Algorithm\_Result: it will be the Cobb Angle.

\*Detect Date: The form is as same as birth-date.

\*Checked by Doctor: Should be Yes or No.

\* Doctor\_Result: it will be the Cobb Angle.

\*Doctor ID: The id of the Doctor (user).

\*Search&Request: Client can retrieve User's Data. Server will input a command that can call the database search function.

\*Delete: The server will input a command that can call the database delete function.

\*Update: The server will input a command that can call the database update function.

Wechat Only:

\*Consultation\_ID: The id of this consultation.

\*Patient\_ID: user id.

\*Doctor\_ID: user id.

\*Upload\_Record\_ID: ID of the X\_ray image that the user uploaded.

\*Create\_Time: Include date and time. The form of data is as same as birthdate, the time also should be digits.

UnRead : Yes or No.

Android Only:

News Record:

\*News\_ID: The id of the news, which should be digits.

\*News\_Content: The content of the news ,which shoud be text.

\*News\_Time: The update time of this news, and the form is as same as create time.

Chatbot Only:

\*Keyword: Text.

\*Answer: Text.

       Patient names will not be subject to restrictions of length or character type. The  name is not be required to match the filename to be saved or open from.

Search will not be restricted as far as input is concerned.

#### **4.1.2 Outputs**

      The outputs are sent to the server and finally sent to the users or algorithm through the server.

     Output includes name, gender, birth-date, result, X-ray images ...,almost every input can be output, so we don't repeat them in the following illustration. Doctor/Admin can view the patients’ information and the patients only can view their own data. After the users input the X-ray images the system will output the result from algorithm or doctor, and the result will be saved in the database.

      Create/Update Information:

      \* If the system receives incorrect/invalid input from the user at the create and update, the system will warn the user that their input is invalid and the information will not been insert to the database. The user will need to let the system know when the input has been changed by re-clicking the confirm button.

      Viewing User data:

      \* Doctor/Admin can read the patients information in the database and the patient only can read their own information(now and past). If the data required does not exist ,the system will send  ‘can’t find’ to the user. The user can input again or exit.

      Delete:

      \* Only the Admin can perform the delete operation .The user will be warned if they want to delete the data. After the data be deleted ,system will return a ‘Delete Success’. If not, it will return a error.

       \*X-ray image: The X-ray image that be inputted. The output will be the url of the image.

### **4.2 Detailed Output Behavior**

      The output varies from user to user and operation to operation. The data gotten from the algorithm and doctor will be saved in different tables. The user get the result more reliable. If there is a result in the doctor’s form, the output will be the doctor’s diagnosis, or the system will return the algorithm’s result.

      The user create ,delete and update operation will bring a tip, create/delete/update success or fall. The read operation will show the information in the database, user can view the data from the past to the present so that they can know the whether illness get better or worse.

The output to the algorithm will be sent through the server, and the server will transform the data to the form that meets the algorithm's need.

## **5 Quality Requirements**

The application must be competitive with similar applications in regards to performance, security ,confidentiality, flexibility and so on.

Performance: Responsiveness to user input

      \* Standard actions that manipulate address books or their contents should not exceed 500ms execution time.

      \* Searching on address books is a possible exception to the above standard, as search performance will depend on the size of the address book.

Security and Confidentiality:Protect the database and the accounts

     \* The key application server and database server are separated to prevent users from directly operating the database server and ensure the database security.

     \* In the software system, the data protection and business operation permission are completed by identifying the user's identity and authority. Compared with the user password, if it is the same, the system will assign the user's operation authority to the user, and then the user will operate the system according to the assigned authority. To prevent the reveal of the user password, in the process of transmission and the password record field in the database should not use plain text transmission and storage, should use effective mainstream technology for the plain text password before the password is transmitted, encrypt the transmission data with the encryption algorithm described in the encryption part, and transmit it to the system after encryption. The system compares the encrypted password data submitted by the user with the encrypted password saved by the user, and the subsequent operation will be carried out if they are consistent.

     \* In this application, Different businesses are handled by different people, and the data that can be accessed by different operators are different. In order to satisfy this requirement, when the user logs in, the system obtains the user's permission after the user's authentication, and displays the corresponding function menu according to the user's permission. When the user browses the data after reading, writing and deleting, the system judges the user's access rights to the data and determines whether the operation is allowed.

Flexibility: Function flexibility and Operation flexibility

     \* The database is designed for more requirements in the future. To meet these probable functions, the database need to adopt a extensible and modifiable method to develop. When some new requirements adding to the system, we don't have to rebuild the whole database, but just to add a few interfaces and instances to realize the functions.

  \* The database system should be compatible with different kinds of operation modes, ranging from common user to large-scale interface call. Different operation modes should be run properly in this database system, and the system is enable to extend more available operation modes in the future.

## **6. Expected Subsets**

L0:

-Basic GUI with the ability to send data to and recall data from the back-end storage

-Ability to add address to address book

-Ability to view an entry in an address book

L1:

-Address book capable of: storing complete entries, sorting, editing entries, and saving

-Complete GUI for access to address book

-Ability to import and export address books

-Open recent

## **7. Fundamental Assumptions**

In order to make the application run properly, there are some advanced issues to take into consideration.

Environment support: the database system should run properly in prevalent operation systems and software environments, such as Linux, Windows 7++, IOS, Android.

Interface Principle: the database system should provide some interfaces for convenient large amounts of data analysis. The interface principles should be designed appropriately.

## **8. Expected Changes**

Features to Add:

      Import and Exporting Sets of Addresses

      User Defined Fields

      Links to Social Media

      Deletion of Many Contacts at Once

      Importing a Photo for Each Contact

      View Contacts By Photo

      Search Contacts By Photo

      Voice-To-Text Searching

      Ability to "Turn Off" Yellow Highlighting for Invalid Fields

Future Platforms:

      Write Front End For Android

      Port Application to iOS

## **9.  Appendices**

### **9.1 Acronyms and abbreviations**

|  |  |
| --- | --- |
| **Acronym or**  **Abbreviation** | **Definitions** |
| DBMS | Database Management System |
| DAO | Data Access Object |

### **9.2    References**

\*《Beginning Database Design》

《Database design requirements analysis》

《RA\_HealthySpine》

《RA\_EPRO》