Final Documentation

iCURE

Xinyi Wang, Wenxin Feng, Wujie Duan, Zhuoer Wang

1. Description of our Project

iCure is an Electronic Health Records software designed to be applicable in different scenarios. Features include electronic records of medical treatments, appointment scheduling, medical forum, patient-doctor private chat rooms and other supporting features all work together to provide both care for patients and efficiency for doctors.

The goal of iCure is to provide services utilizing the power of the data to create a user-friendly environment for patients and doctors. By saying this, we implemented functions for patients and doctors respectively during medical treatment processes. Moreover, iCure also serves as a platform for interactions between patients and doctors, which includes forums and chat rooms for medical consulting. For patients, iCure focuses on enabling them to have medical care resources not just when they're at the hospital, but whenever they have concerns for their health. For medical care providers, the electronic records help them to keep track of the medical history of each patient they treat, which will make the whole treatment process more efficient. And as a software-driven by data, iCure is able to provide tailored service for each user, which can refine the traditional medical treatment processes significantly.

The GitHub link for our project is: https://github.com/Cheryl008/iCure

2. Software Process

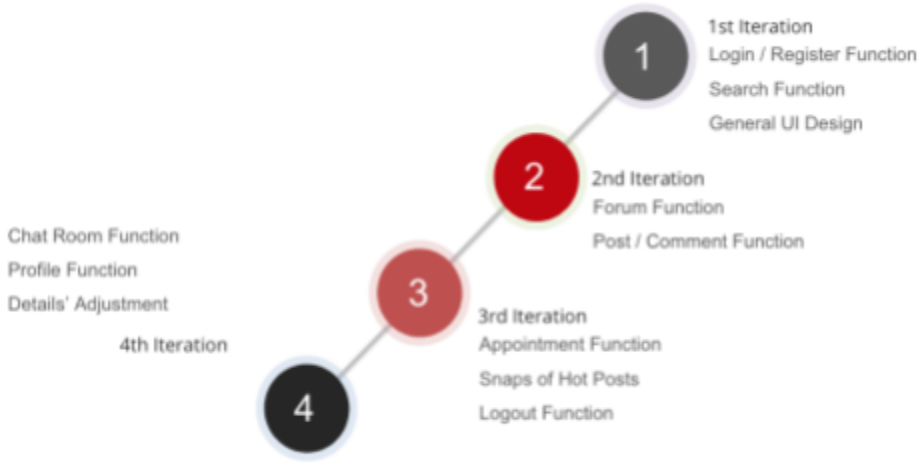

Figure 1 - Four Iterations

The software process we use is `eXtreme Programming` (XP). The reasons we choose agile method are that our team size is small and the requirements may rapidly change.

We had four iterations and each iteration was organized as a two-week cycle (Figure 1). In the first iteration we finished login/register function, search function and general UI design. In the second iteration, we implemented forum and post/comment function. In the third iteration, we completed appointment function, snaps of hot posts and logout function. In the final iteration, we finished chat room, profile and made refactoring and adjustments.

Before starting to implement, we did project planning which clarified the priority of each task and identified the tasks of each iteration based on the priority and the estimated workload. We also did project scheduling, which enabled us to finish each iteration on time.

During the implementation, we divided our team into different roles. Wujie and Zhuoer worked on the back-end implementation, while Xinyi and Wenxin worked on the front-end implementation. During the programming process, we used the strategy of pair programming to secure the quality of the codes. Moreover, this strategy helped each of us to get familiar with the code base of our project. Furthermore, we agreed to release codes frequently with GitHub as version control such that everyone could catch the progress and know what others were working on. Finally, we maintained frequent communication during the whole process which was essential for us to achieve all our plans.

After each iteration, we spent great effort testing each function to make sure that our web application worked smoothly in that iteration. The test-driven development ensured the high quality of our system.

3. Requirements and Specifications

We will introduce all use cases of our project in this section.

UC1 Flow of Events for the LogIn Use Case

1.1 Preconditions:

All visitors can login with their existing accounts as doctors or patients.

1.2 Main Flow:

When the user enters our webpage, there will be a button for login. By clicking the Login button, the user will be directed to a login page, where he/she can login with an existing account by choosing login type and providing the email address and password. If the email and corresponding password are correct, the user will successfully login as a patient or doctor and be directed back to the homepage. After login, more functions, such as making appointments and making posts, are unlocked and the user can also check and edit his/her own profile.

1.3 Subflows:

[S1] If the user hasn't created an account, he/she can click the button for creating a new account on the login page to register. The user can choose to register as a doctor or patient, and then will

be directed to the corresponding register form. By filling and submitting the registration form with all the required information, the user will successfully create a new account and be directed back to the homepage. Finally, the user can login with his/her newly created account.

1.4 Alternative Flows:

[E1] If the user provides invalid email address or password, there will be an error page indicating that the login information is invalid.

UC2 Flow of Events for the Search Use Case

2.1 Preconditions:

All types of users can use the search bar on the front page.

2.2 Main Flow:

The search bar is placed in the center of the front page. Before typing in the search content, the user needs to select the category he/she would like to search for, which includes (Doctor) Name, Hospital, Department and Position. By clicking the search button, the search results will be displayed in another page. The search results are sorted according to their rating and the doctors with higher rating are displayed on top of the page. The user can then pick any doctor to set up an appointment with.

2.3 Subflows:

[S1] If the user would like to make an appointment with a doctor from the search results, he/she can click the Make an Appointment button to enter the appointment making page, which is included in the appointment user case.

2.4 Alternative Flows:

[E1] If there is no result meeting the search criteria, the result page will not display anything.

UC3 Flow of Events for the Forum Use Case

3.1 Preconditions:

All types of users, including visitors, patients, doctors, can use the forum.

3.2 Main Flow:

To access the forum page, the user just needs to click the button on the toolbar of each page, including the front page. After clicking, the user will be led to a page where all posts are listed with the order of their hit numbers. The user can either pick a post or use the search bar to search with a specific title or an author name. The posts displayed will alter based on the result. Then, the user can click to enter the chosen post's page.

Also, the user can click one of the hottest posts displayed on the front page to enter the chosen post's page.

3.3 Subflows:

[S1] After entering the post page, the content of the post will be displayed. Below the content, there is the comment section. One can leave comments and see the comments that have already

been made. Also, the user can search for a specific author name to see the comments left by the specific person.
[S2] The user can make a post by clicking the Make Post button on forum pages. In the composing page, one can enter a title and content in the word editor. By clicking submit button, the post will be submitted. The website will lead the user back to post display page, where the new post will be visible.

3.4 Alternative Flows:

[E1] If there is no result that meets the searching criteria, the page will display "no search result".
[E1] If any error occurs, users will be directed to the same error page where the error message on the page will change according to the type of error. Errors include make post error and make comments error.

UC4 Flow of Events for the Make Appointment Use Case

4.1 Preconditions:

When logged in as patients, users can make appointments with specific doctors.

4.2 Main Flow:

After the patient has logged in, he/she can enter the search result page by two ways. First, he/she can enter search result page after using the search bar. Second, on the Homepage, there is a drop down to enter appointment page and after entering appointment page, there is a Make Appointment button to enter search result page. Then the patient can use time slot on the calendar. Then after entering title and chief complaint of this appointment, he/she can make an appointment with the chosen doctor.

4.3 Subflows:

[S1]The patient can return back to the search result page to choose another doctor or return back to his/her appointment page to see the newly made appointment from the making appointment page.

UC5 Flow of Events for the Manage Appointment Use Case

5.1 Preconditions:

When logged in as patients or doctors, users can manage their appointments.

5.2 Main Flow:

After the patient has logged in, there will be a dropdown where he/she can click to enter the available appointment page. All the upcoming and history appointments one has will be displayed. When clicking into the appointments, detailed information will be shown.

5.3 Subflows:

[S1] When a doctor click into an upcoming appointment, he/she will be able to write diagnosis for this appointment. Patients' health data will also show in the diagnosis page. After the doctor

complete diagnosis and submit it, the appointment will automatically become history appointment.
[S2] When a patient click into a history appointment, he/she will be able to rate this appointment and write comments. Each appointment can only be rated once by the patient.

UC6 Flow of Events for the Chat Use Case

6.1 Preconditions:

The patient has to have an upcoming or history appointment with the doctor before sending messages to the doctor.

6.2 Main Flow:

After the patient has logged in, there will be a dropdown where he/she can click to enter the available chat page. In the available chat page, all the doctors with whom the patient has upcoming appointments will be displayed, where the patient can click and enter a chat room with the selected doctor. The doctor can likewise find all his/her patients of the upcoming or history appointments in the available chat page entered by a dropdown.

6.3 Subflows:

[S1] When the user clicks to enter the chat room, the most recent several messages will be displayed in the content area in the webpage since the messages are displayed according to time. The newest message is displayed at the top of the content area while the older messages can only be seen when the user scrolls down. Below the content area, there is an input text box where the user can type in the message he/she wants to send. When the user clicks the Send button, the message in the input box will be sent.

6.4 Alternative Flows:

[E1] The doctor or the patient cannot chat with those who do not appear on their Message list.

UC7 Flow of Events for the Profile Use Case

7.1 Preconditions:

The user must be logged in to view his/her profile.

7.2 Main Flow:

By clicking the Profile button, the user will see all his/her information displayed in the profile page.

7.3 Subflows:

[S1]If the user would like to make any edit, he/she can click the Edit button and all information will be changed into input box. Note that information like name, id and type cannot be changed. After filling out the new information, the user can click Submit to confirm the edit.

UC8 Flow of Events for the LogOut Use Case

8.1 Preconditions:

All logged-in users, including patients and doctors, can logout.

8.2 Main Flow:

After logging in, there will be an icon on the right corner of the nav bar. There will be a dropdown where the user can click logout. After logging out, the user will be directed to the homepage as a visitor.

Figure 2 shows the relationships between use cases.

Figure 2 - Relationships between Use Cases

4. Architecture and Design

The type of architecture we use is Client-Server Style. Our product is a web application, so Client-Server is the best structure for us. From server to client to database, we implemented a full-stack JavaScript architecture. We used JavaScript with Bootstrap for front-end, ExpressJS with Node.js for back-end, and MongoDB for database.

The basic process is that first the client makes a request and the request is processed by client-side JavaScript. Then the request is sent to Node.js/ExpressJS server and the server sends the request to MongoDB. After retrieving data from database, the data are sent back to server and then to front-end and displayed. Figure 3 shows the design of database. Figure 4 shows the relationship between front-end pages and their main functions.

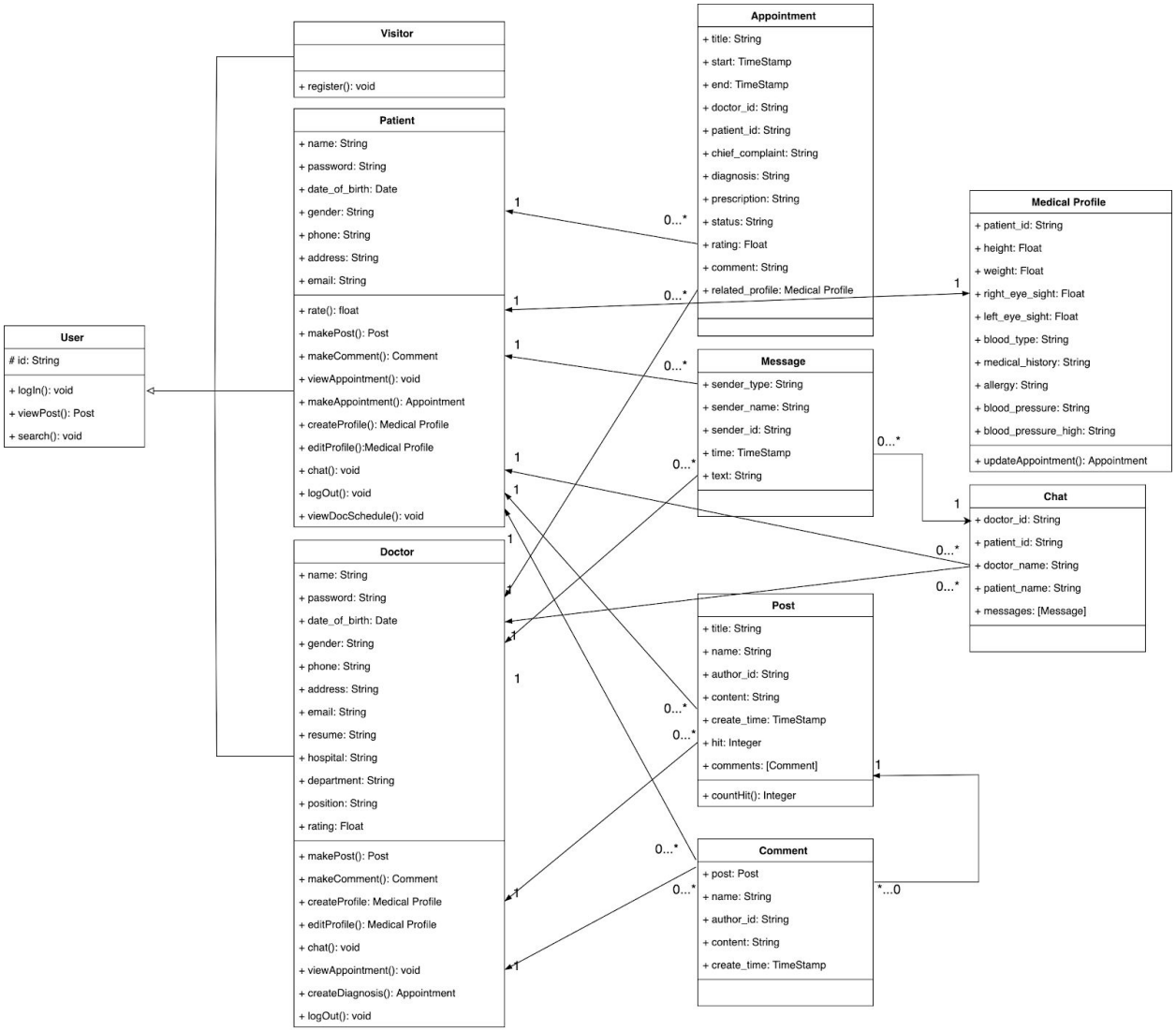

Figure 3 - Design of Database

We use Node.js as our server side platform. It has three features: asynchronous and event driven, very fast, and single threaded but highly scalable, which is suitable for building our light-weight and fast-responding website. We use ExpressJS as our server framework. It is minimal, providing us the absolutely required tools to build our app, and flexible, with numerous modules available on npm for express, which can be directly plugged into express. Therefore, it is very suitable to have some extensions on our website, such as FullCalendar or Quill. MongoDB is very fast, supports iterative development, and has flexible data model, which is proper for storing our various data. In all, the flexibility and efficiency of the full-stack JavaScript development suits our agile methodology well.

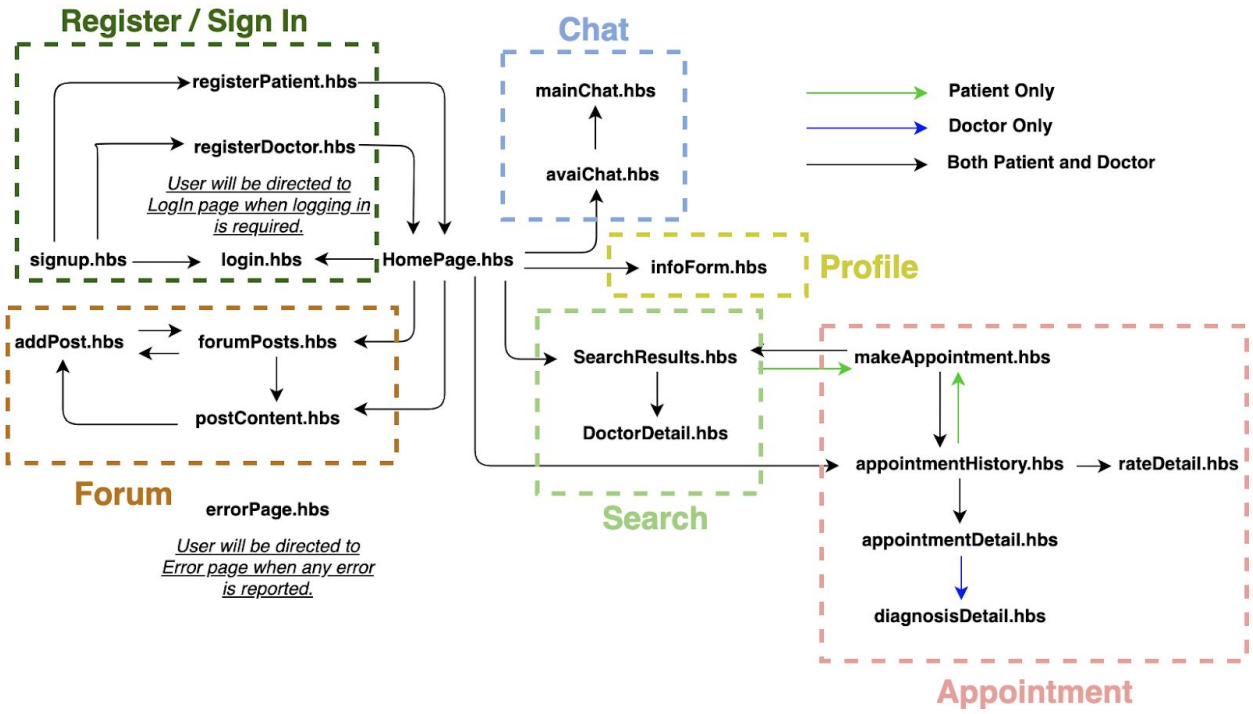

Figure 4 - Relationships between Front-end Pages

5. Reflections and Lessons Learned

Xinyi Wang: Looking back to our initial plan, I find that we actually successfully implemented most of the functions we designed. All the functions with high priorities were achieved. Although there are tons of refinements that can be made to this web application, the process of developing this prototype has already enabled me to learn the whole development process of a software. From the requirements elicitation to the software architecture, from the implementation to the testing, I experienced all stages of software development in practice thanks to this project. Our working process was according to `Agile`, which is very rewarding learning coding

techniques from my teammates. I learned JavaScript better and picked up more techniques to do front-end implementation, which would be all very useful in my future studies.

Wujie Duan: In this project, I learned how to cooperate with other team members to build a complete system. In my previous projects, I was usually in charge of both front-end and back-end so that I tended to not strictly follow coding standards. However, this time in order to integrate my back-end code with others' front-end code, I gradually learned how to write readable, reusable, refactored code. Moreover, I learned how to write and design automated tests with Selenium, which brought me closer to becoming a well-rounded software engineer. This project also allows me to recapitulate what I have learned about back-end JavaScript development and I am very proud that we have tried the trending full-stack JavaScript techniques (MongoDB, Node.js, ExpressJS and front-end JS framework).

Wenxin Feng: I have learned a lot from this project and gained experience of participating in a full cycle of software development. My understanding of this field has gradually accumulated through this process. The best thing about this project is that I can try what I have learned recently out into practice. For example, I have designed code structure based on design patterns and reviewed my codes using cyclomatic complexity. I have also used Bootstrap to design UI by learning each component from documentations. My team members also helped me a lot during the process. We made great teamwork with each other and the whole development process went on smoothly. I have mastered tools such as Github, Selenium and MongoDB and other languages important in software engineering, which I believe will support me in my future study.

Zhuoer Wang: During the process of software development, in the technical aspect, I get more familiar with the back-end development in JavaScript with Node.js and Express. Also, I learned how to complete interactions between front-end and back-end, as well as between back-end and databases(MongoDB). More importantly, it's my first time exploring and integrating external API (fullCalendar) into the web application and has achieved success. From the aspect of teamwork, I realize the importance of efficient communication and plan. On the one hand, even our teamwork works smoothly and have a relatively clear plan, we still met some difficulties in version control, for example, merging conflicts reported by git. On the other hand, with active communication and knowledge sharing, I also learned much about the front-end and testing from my teammates. Last but not the least, as a software engineer of an EHR system, I realized that I should not only pay attention to the technical problems, but also the rationality of the whole program and the user experiences.