Web Based Online Dynamic Electrocardiogram System

by

He Ma
Bachelor of Engineering, Shanghai Maritime University, 2013

A Report Submitted in Partial Fulfillment
of the Requirements for the Degree of

MASTER OF ENGINEERING

in the Department of Electrical and Computer Engineering

Supervisory Committee

Web Based Online Dynamic Electrocardiogram System

by

He Ma
Bachelor of Engineering, Shanghai Maritime University, 2013

Supervisory Committee

Dr. Tao Lu, Department of Electrical and Computer Engineering
Supervisor

Dr. Lin Cai, Department of Electrical and Computer Engineering
Departmental Member

Abstract

Supervisory Committee

Dr. Tao Lu, Department of Electrical and Computer Engineering
Supervisor

Dr. Lin Cai, Department of Electrical and Computer Engineering
Departmental Member

Dynamic electrocardiography has an important value for people with heart disease to help detect irregular arrhythmia, while few portable products of dynamic electrocardiography are commercially available. To solve this problem, this project, through the use of the CakePHP and RequireJS frameworks, develops a comprehensive web application for a wireless dynamic electrocardiogram system. The application uses several methods to improve the data loading performance, including gzip compression, multi-layer caching, and garbage collection. Authorization and input fields escaping were also implemented to safeguard sensitive information. By configuring a load balancer and scaling the server and the database, the application is highly scalable. Styling consistency of web page components was introduced to optimize user experience of the application. To enhance application's page rank on search engines, Search Engine Optimization (SEO) strategies were introduced. Furthermore, implementation of the PHPUnit test is also provided to ensure software quality.

Table of Contents

List of Tables

List of Figures

Acknowledgments

I would like to thank:

Dr. Tao Lu and **Dr. Xiaodai Dong** for providing me this opportunity and valuable constructive suggestions during the study of DCG web application.

Weizheng Li, Lan Xu, Yuejiao Hui and other team members who participate in the DCG project, for your generous help and support during collaboration in the research process of the project.

He Ma

Feb 5, 2016

Chapter 1 Introduction

1.1 The Problem

Heart disease is one of the main contribution factors to mortality rate. In Canada, “more than 1.4 million people have heart disease and approximately 33,600 of which die from heart diseases every year” [1]. The reason that heart disease contributes such a high mortality rate is that the heart attack occurs at random and the time window for rescue is short. On the other hand, study has shown that many heart diseases in early stages have already produced the abnormal electrocardiogram (ECG) [2]. Because an ECG can accurately record the electrical activities of heart, the minor difference in an ECG may imply a type of cardiac disease. To better understand how the ECG can reflect heart conditions, the principle of recording the electrical activities is described in the next paragraph.

1.2 Electrocardiogram

An electrocardiogram is the visual output that an electrocardiograph produces. An electrocardiogram is the temporal trace of the **potential difference** between different parts of myocardial cells, which are the muscle cells that compose the cardiac muscle in the heart. The myocardial cell membrane is semi-permeable. In resting state, positive cations align outside the membrane, and negative anions align inside. Consequently, the potential outside membrane is higher than that of inside. This state is also called the polarization state. The process of a cycle of the heart beat against electrocardiography is described as the following stages:

1 In the resting state, since cells in all parts of the heart are in the polarization state, there is no potential difference among them. Consequently, the curve of ECG displays a straight line.

2 When the heart starts beating, myocardial cells receive stimulation from the sympathetic nerves, which are part of the autonomic nervous system, resulting the change of cells' membrane permeability. Consequently, a large number of cations will flood into membrane within a short period of time. The potential inside membrane changes from negative to positive. This process is called depolarization, and the depolarization of myocardial cells in the whole heart is recorded as ECG. This potential curve is called depolarization wave [3] which contains P wave and QRS wave as shown in Fig. 1.

3 When depolarization finishes, cell membrane expels cations, causing potential inside membrane changes back to negative to that outside membrane, this potential curve is called repolarization wave, as called T wave shown in Fig. 1.

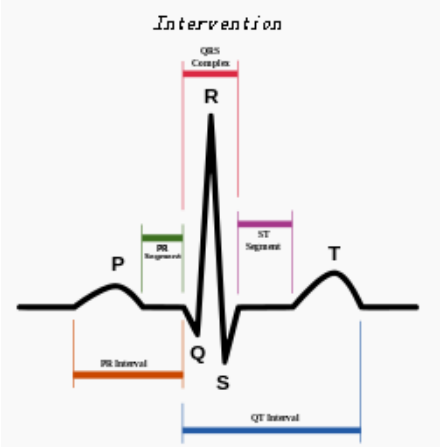


Figure 1. A Normal Sinus Rhythm in Electrocardiography Reprint From [3]

4 When all myocardial cells finish repolarization and switch back to polarization state, there will be no potential difference between the different parts of myocardial cells, and the potential curve goes back to an equipotential line.

In practice, an ECG is the record of continuous potential changes of multiple heart beat cycles described above. Fig. 2 shows an example of an electrocardiogram containing the heart electrical activities in three channels. The electrical activities are recorded by placing pairs of electrodes on different parts of human body and then the temporal change of the voltage difference among them are recorded continuously [3]; each pair of electrodes detects electrical activities in one dimension which will be recorded via one channel in an ECG.

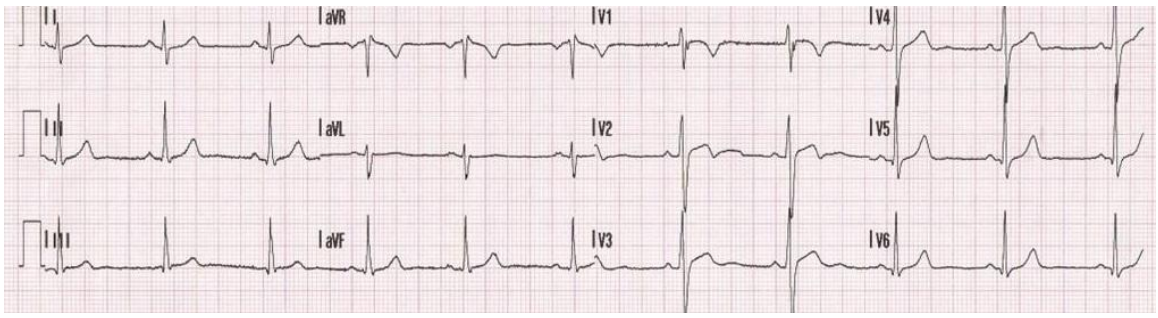


Figure 2. An Example of Electrocardiograms Reprint From [3]

Since its invention by Alex. Muirhead [4], the electrocardiograph technology has been very accurate in detecting heart diseases and widely used in clinical practice. Physicians can judge patients' heart conditions by observing the differences of values of P waves, QRS waves, and T waves in channels. However, regular ECGs still cannot be used to effectively diagnose some arrhythmia types of heart diseases, because the abnormal ECG signals happened in these types of diseases are often transient and uncertain. Consequently, another new type of the ECG, the dynamic electrocardiography, becomes extremely useful to patients with these diseases to save their lives.

Dynamic electrocardiography (DCG) is a method that can record the electrical activities of heart in the long-term period no matter whether the tested person is in the walking or sleeping state. The main difference between DCG and traditional electrocardiogram is that DCG is a long-term recording of ECG which can extend to 24 hours, containing up to one hundred thousand electrical signals. DCG can immensely increase the detectable rate of non-persistent arrhythmia, especially for transient arrhythmia and transiently ischemic myocardium. To be specific, DCG has the advantages in the following applications:

1. DCG can be used by patients with cardiac arrhythmias to detect arrhythmia that is life threatening so that they can get timely treatment, such as detecting pairs or ventricular tachycardia for premature beat patients.
2. DCG is commonly used to detect arrhythmias caused by various types of cardiovascular diseases, including myocardial infarction, cardiomyopathy, myocarditis, etc.
3. DCG can be used to the research on patients with syncope to find cardiogenic syncope cases so that these patients can get treatment in time.

For the patients mentioned above, DCG plays a critical role to their wellbeing. However, till now there are few mature DCG products provided in the market for patients and physicians to use. Fig. 3 is a commercially available ECG device, in which ECGs are printed on paper and the device can be used only at home. To monitor long-term DCG, a compact, portable device that can be carried and used by patients in daily life is necessary. It should store ECG data in a digital media so that physicians can observe them through the Internet afterwards.

Figure 3. An ECG Device Reprint From [5]

To implement a portable DCG system, Hu [6] introduced a portable ambulatory electrocardiogram monitoring system based on Bluetooth. However, it does have some disadvantages:

1. Although the module is portable, it does require a PC to receive DCG data through Bluetooth, which reduces the portability.
2. The Bluetooth used to wirelessly transferring data is power-consuming. This makes the device hard to last for more than 24 hours of continuous transferring DCG data if powered by batteries.
3. The device has four modules and one LCD, which occupies a large space and is not portable.
4. DCGs in the system are not easily accessible. PC has to install the software to render DCGs. There is no convenient way that physicians can directly use to observe patients' DCG through the Internet.

Therefore, to design a new comprehensive DCG system the following features have to be included:

1. It should have a convenient way to receive the data without using a PC which is too inconvenient to carry in daily life.

2. It should use a more power-efficient method to transfer data so that the device functions for a long period of time, typically more than a few hours.
3. The device should be compact enough for patients to carry and contain as few modules as possible.
4. It should have a client-side application accessible through the Internet so that physicians can timely observe patients' DCGs data.

To meet these requirements, the wireless dynamic electrocardiogram system, in collaboration with Prof. Dong's group, is under development.

1.3 Wireless Dynamic Electrocardiogram System

As introduced above, the wireless dynamic electrocardiogram system provides a convenient and affordable way for people with a history of heart disease to record long-term DCG and also a platform to present DCGs to remote physicians to help diagnose heart disease. The wireless DCG system has the following features:

1. It uses a mobile phone to receive data through Bluetooth low energy technology. The mobile phone relays data to a database server through the Internet.
2. Bluetooth Low Energy (BLE) technology is used as the data transferring protocol that has a tremendous low power consumption enabling device to remaining functioning for long term.
3. There are only two single-module chips in the device, ADS1192 and CC2540, which makes the device smaller and portable.
4. A comprehensive web application is implemented to present DCGs online, thus physicians can conveniently remotely observe DCG data.

5. The system has a database server and a database to process and store enormous DCG data.

The system structure of the wireless DCG system is shown in Fig. 4. It uses four electrodes attached to the human body to collect heart electrical activity analog signals from the body via two channels. Then, through the ADS1192 and CC2540 chips, the raw data are transferred to a mobile phone using BLE. A single-module chip with a button battery (3v and 220mAh) using BLE can work a few months or even years. In contrary, normal Bluetooth with two AAA batteries can only work for a few days. To accommodate the long-term recording of ECG data, the mobile phone will periodically send small sets of data to the database server when WIFI available. The database server receives data and saves it in a database. Web application sends HTTP requests to the database server to retrieve DCG data from it, and then present DCGs to the user.

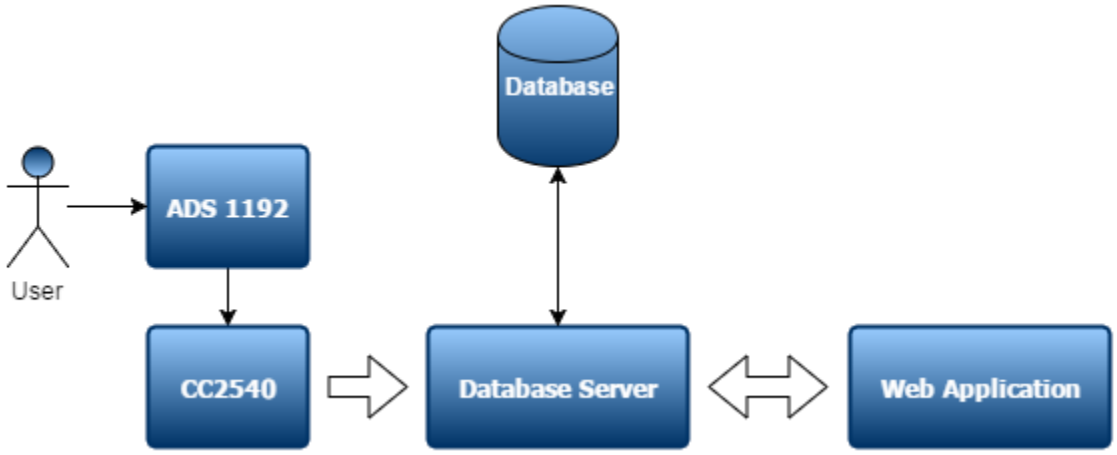


Figure 4. Dynamic ECG System

With wireless dynamic ECG (DCG) system, patients are able to record long-term DCG. Typical user scenarios are as follows:

1 A patient attaches electrodes to the body and activates wireless DCG device.

2 The patient downloads the Android app, and then connects the device to the app.

3 The patient clicks “start recording” to start recording long-term DCG.

4 A complete DCG is recorded and saved in the database.

5 Physicians log in web application and observe patients’ DCG online.

6 Physicians make comments to patients’ DCG on the application.

7 Patients view physicians comments on the web application and seek for treatment if necessary.

The last three scenarios are critical in the system, because DCGs only recorded but cannot be observed by physicians on time are less useful. The system utilizes Internet technology to implement a web application so that physicians only need to log in the web application through Internet at home or hospital to see patients’ DCGs.

1.4 The Main Purpose of the Project

The main purpose of this project is to implement the web application component of the wireless DCG system. The web application should contain the following features:

1. It should have authentication and authorization mechanisms so that physicians can access authorized DCG data of patients.
2. It should render DCGs in different channels on one web page, and users should be able to scroll and set different time interval to better observe the values in charts.
3. Physicians should be able to request access to a patient’s data. They should also be able to leave comments for patients’ DCG tests.
4. Patients should be able to see comments for their DCG tests.

To implement the web application, the system architecture among the DCG web application, the database server, and the Internet is designed as shown in Fig. 5. The DCG web application uses an interface to communicate with the database server to obtain

data. Meanwhile, it runs as a web server so that the user can access it through the Internet. This communication way is a typical client-server architecture.

Figure 5. Client-Server Architecture of the Wireless DCG System

More specifically, to realize the DCG web application in the system with this architecture, the following main sections will be introduced:

1 Design and write application program interface (API). The DCG web application uses RESTful API to communicate with the database server.

2 Implement the data interaction by using CakePHP framework, which is a back-end framework written in PHP. The web application uses CakePHP to send HTTP requests to retrieve data.

3 Design and implement the user interface with the support of JavaScript libraries, such as Bootstrap, which is used to build the layout of web applications. Another library, jQuery, is also used for events binding and data updating.

4 Verify the data loading performance and the software quality. The DCG web application uses RequireJS to modularize JavaScript files; it will also implement multi-layer caching and file compression to improve performance. PHPUnit test is introduced to minimize software failure.

5 Design database schema according to demand, and then write Structured Query Language (SQL) as a description of the schema.

6 Secure the DCG web application by preventing users from submitting malicious inputs and cross-site attacking.

1.5 Outline of the Report

The outline of the report is as follows:

Chapter 2 describes the functionalities of the DCG web application based on different web pages and user groups.

Chapter 3 is the description of the system architecture, including how the web application communicates with the database server.

Chapter 4 is a detailed discussion of the evaluation of the data loading performance. Additionally, several strategies are introduced to improve performance.

Chapter 5 explains why the DCG web application needs to implement authentication and authorization. Some popular attacks on web applications and how to prevent them are also introduced in this chapter.

Chapter 6 discusses the scalability of the application. Feasible solutions are given in this chapter to increase the scalability and robustness of the DCG web application.

Chapter 7 describes the user experience of the web application. This chapter also briefly introduces search engine optimization and page rank algorithms.

Chapter 8 discusses testing to ensure the overall quality of a software system. Testing can also reduce the potential for regression when modifying or adding features in the future, so effective ways of testing must be described.

Chapter 9 provides some recommendations of future work and draws conclusions for the report.

Chapter 2 Functionalities

The essential function of web applications is presenting content to the user. In the DCG web application, the core contents are the DCGs presented to the user. In addition to that, the application has several other types of contents such as user information, DCG test information, comments, and notes that are all involved in various user actions. Users trigger actions on web pages to manipulate data containing these contents. However, different authority strategies need to apply to different types of users. The actions and functionalities of the DCG web application can be categorized by user groups. As shown in Fig. 6, there are three user groups in the application: doctor, patient and guest. There are six main sections in the application: DCGs, records, comments, notes, connection and public. Users from different user groups will have their own access authorities to these sections.

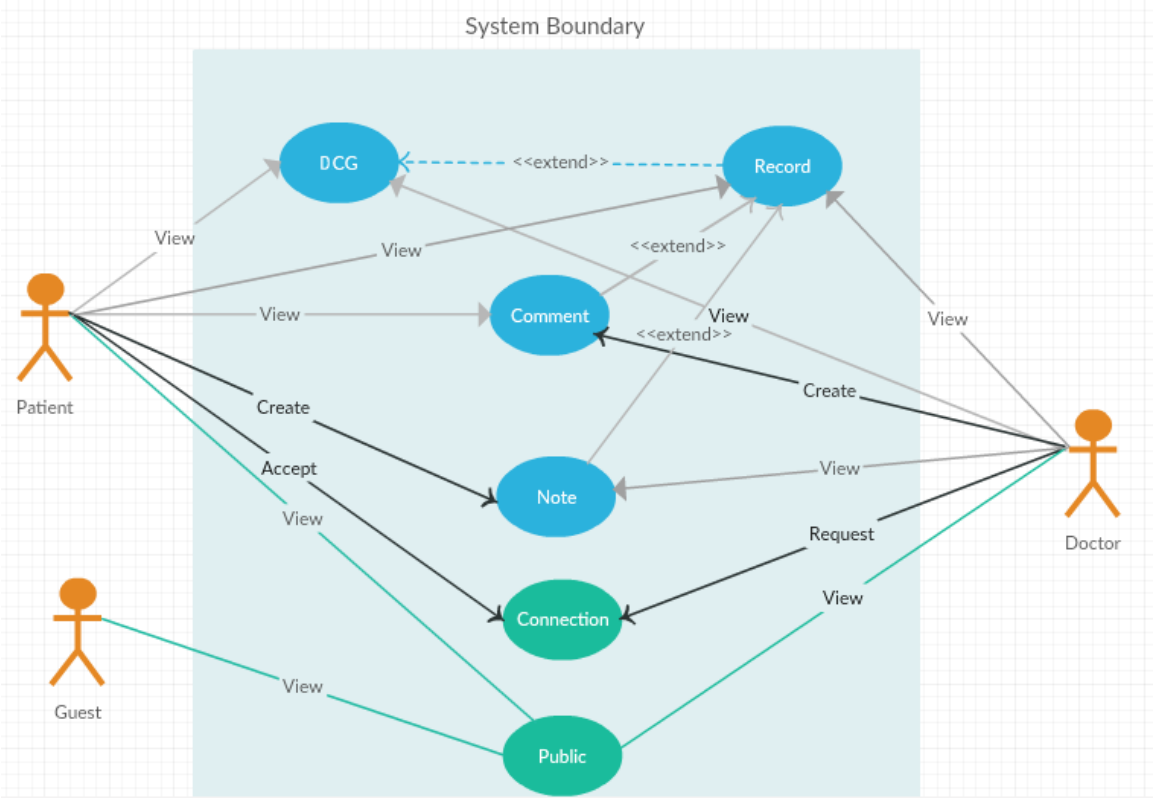


Figure 6. Use Cases in the DCG Web Application

From the user group's perspective, the guest user has the lowest level of authority. This user group can only see the public section including product introduction, news, and other non-private information. The guest user does not require a login. In contrast, the patient and doctor users require a login. After login, patient users have access to all sections so that they can see their DCG records and doctors' comments in the application. Similar to the patient user, doctor users also have access to all sections after login. Once logged in, the doctor user can see patient users in the application and establish the connections with them. Then, the doctor user can see a patient's DCG and make comments.

For action sections, these sections are also categorized by colors. The blue ones are DCG-related, and the green ones are not DCG-related. The lines between user and

section in black and bold represent HTTP post actions, which will usually change the data in the database. The lines in gray stands for HTTP get actions that retrieve data only from the database server. The lines in green are also HTTP get actions but retrieve only public information. The DCG-related sections can be viewed only by the patient user and authorized doctor users, including the DCG data, associated records, comments, and notes. In the note section, a patient user can create a note but only an authorized doctor user can view it. In the comment section, a doctor user can create a comment and the patient user can view it. In the connection section, a doctor user can send a connect request to a patient user, and then patient user can then either accept or decline the request.

More specifically, the use cases are listed in priority order for the three user groups in the following tables:

Table 1. Guest Use Cases

Priority	Use Cases
High	View Homepage View Product View Pricing Sign up
Medium	Contact Follow
Low	View Team

Table 2. Patient Use Cases

Priority	Use Cases
High	Login Edit Profile Accept/Decline Doctor Access Request View DCG View Records View Comments
Medium	View Note View Doctor Profile Change Password Search DCG
Low	Create Note Share DCG Register

Table 3. Doctor Use Cases

Priority	Use Cases
High	Login Edit Profile View Patient DCG Send Access Request View Patient Record Create Comments
Medium	View Patient Note Search DCG Search Patient Change Password
Low	Register View Patient Profile

2.1 Guest User

The DCG web application provides basic introductory information about the DCG system and hardware devices to the public. This information generally does not require

authentication or authorization. A guest users can send an HTTP request directly and access the information. This public section is divided into several web pages in the application, which will be introduced in the following paragraphs.

2.1.1 Homepage

The homepage is one of the most important pages for a web application; it is the first page a user will see when using the application. Thus, the homepage will have a high daily page view. The loading time of a homepage with a normal bandwidth should not be more than 4 seconds, and its user interface should be attractive and easy to use. Fig. 7 (background image from [7]) shows the homepage of the DCG web application. The homepage will show up when users access the web application by typing URL in browser (https://www.cardiacare.ca).

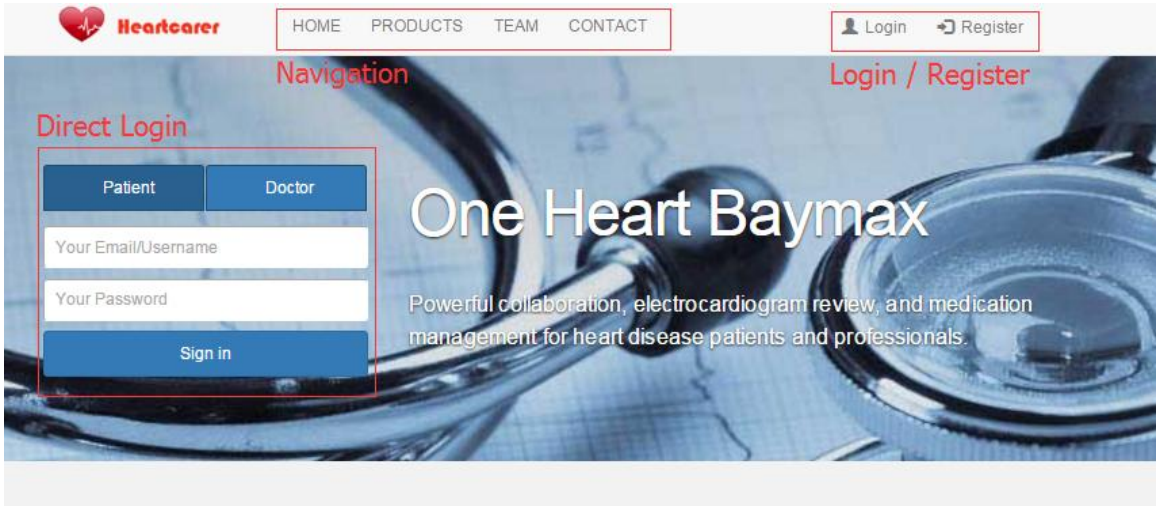


Figure 7. DCG Web Application Homepage

It can be seen that the homepage shows the theme of the application which is heart healthcare. A navigation bar is located at the top, and some links are present, including Products, Team, and Contact. Two authentication links (Log in/ Register) are on the top

right position. Below the navigation bar there is a background image and a form for direct login. On the right side, there is a title and some description of the DCG web application. The homepage gives the user a general impression of what the application is for. It also provides links and forms that are useful to all types of users.

2.1.2 Team

This section lists the team members who participated in the DCG project. It briefly introduces the team members' research areas, backgrounds, and interests.

2.1.3 Products

The Products page gives details about the DCG hardware device and how it can successfully work in the long term with a mobile app under a BLE connection.

2.1.4 Sign in and Sign up

There are two forms on Sign in page, one for a patient user to sign in, and the other for a doctor user to sign in. The forms can be switched by clicking the tab above it. A user must input a correct username and password on the corresponding form to log in. The password is hidden using stars to secure the information. When the application sends a login request to the database server, the password will also be encrypted by a secret key. Once the user successfully logs in to the DCG web application, the corresponding section with personal services will show up according to the user group. On the bottom right corner, there is a blue button that will link a user to a sign up page, which is useful if the user is a first-time user.

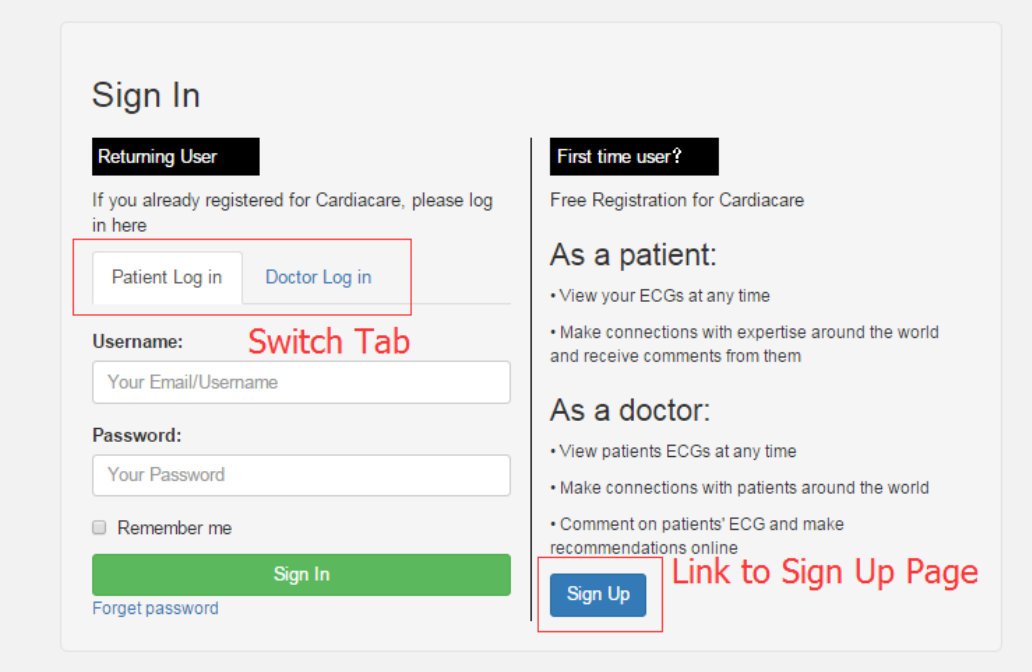


Figure 8. Sign In Page

The Sign up page is shown in Fig. 8. Signing up for the application requires more personal information from the user than signing in; it requires the user's full name, health card number, age range, email and address. All personal information will be encrypted and then transferred using HTTPS. The Sign up page will generally be used by doctor users, because based on the current design patient users will use other approaches to sign up for an account in the system when they start to use hardware device and mobile apps. For instance, when a user starts to use the device to log daily DCG data, the mobile app will require the user to sign up for an account so that the DCG data can be associated with the account. This account information is saved on the database server. The user can then directly log in to the DCG web application instead of creating a duplicate account in the system. However, because the doctor user does not use a hardware device with the application, the doctor user should sign up for an account through the web application.

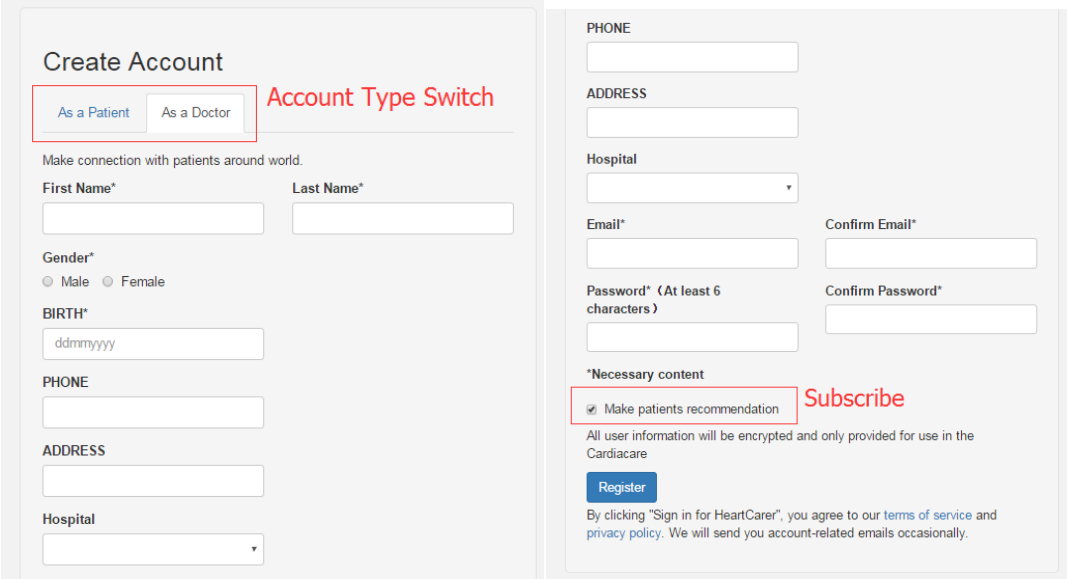


Figure 9. Sign Up Page Containing Comprehensive Information

2.1.5 Contact

The contact page contains the contact information for the research group including its location, phone number, and email address. On the left side, there is an inquiry form, with which the user can send an inquiry on any concerns about the web application or the DCG system. This form is the only one in the application that does not require authentication, so a complete input escaping is necessary.

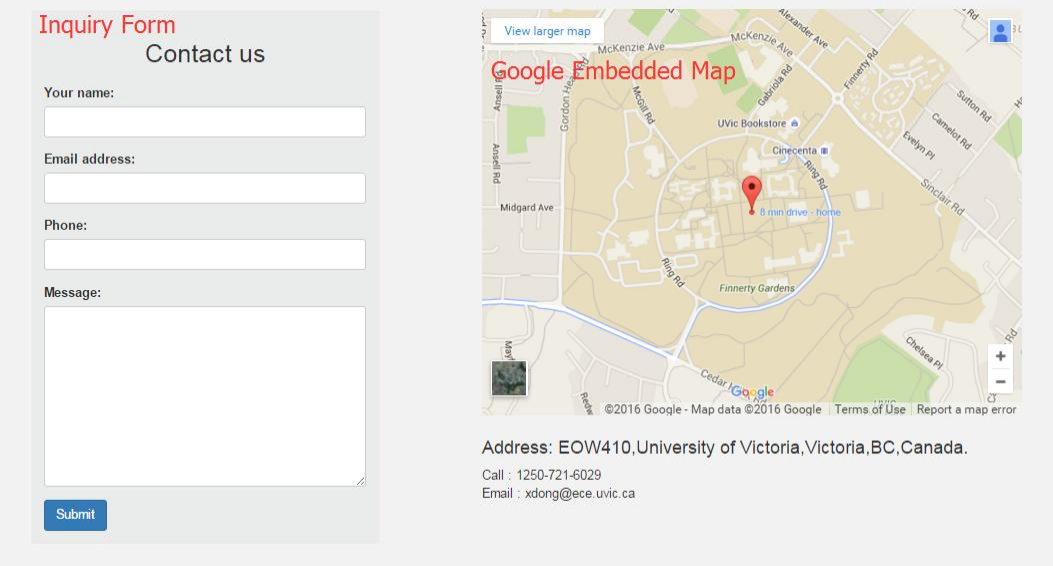


Figure 10. Contact Page

2.1.6 Social Media and Others

One way to keep a close connection to the outside world and increase the page views is to provide an official social media account on the page. The user can follow the official account and obtain the latest updates and news about the DCG application. Users may also share posts from the official account, which usually contains some links to the application. A large number of incoming links can greatly increase the page rank of the DCG web application as will be discussed in chapter 7. Social media buttons are located in the middle of the footer at the bottom of the application. The footer also provides series of useful links to help the user find information.

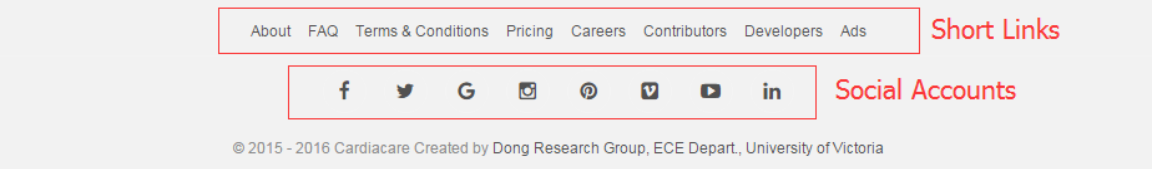


Figure 11. Footer Layout

2.1.7 FAQ

Frequently asked questions will be logged on this page to help solve users' problems. This page can be enriched as more users start to use the application.

2.2 Patient

The pages discussed above belong to the public section, which any user can access. Patient users will have access to additional personal services including the management of DCG data and other personal information. The functionalities for the patient user group are described in the following paragraphs.

2.2.1 Dashboard

The Dashboard page is the index page after login. This page shows an overview of the personal information for a patient user. For example, as shown in Fig. 12, at the top, there are four panels in different colors that show how many DCG tests, doctors, comments and notes that user has. Clicking on one of panels will redirect the user to the corresponding page. For instance, clicking the Doctors panel will redirect to the doctor list page. The user can also see the latest DCG tests in the middle of the page and the latest comments from authorized doctors by clicking on the switching tab. There is also a DCG test view history on the right side, indicating which tests have been viewed recently.

Although this page is the index page for the patient users, this page cannot use the cache because the latest tests are changing all the time. To enhance the loading speed, the web application sends a request with a parameter "limit" to the database server. This parameter ensures that the request retrieves only the latest tests and view history instead of all of the tests and history. Moreover, when the user expands one of the latest tests, the

DCG data are then asynchronously loaded and rendered. If the user does not expand the tests, it is not necessary to retrieve those data from the database server.

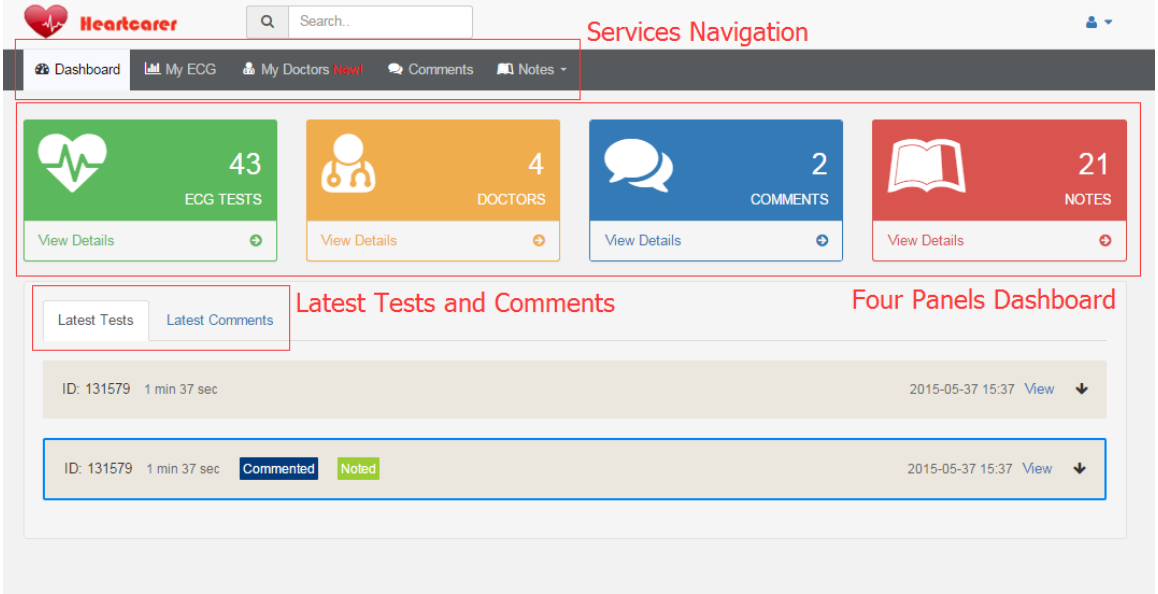


Figure 12. Patient Dashboard

2.2.2 DCG Test Center

The DCG Test Center page contains a table of DCG tests. Each row in the table represents a separate DCG test. The row contains information on the test, including its ID, the numbers of records, comments and notes, the length of the test and the date and time the test was created. An in-place search is provided at the top right corner and pagination is located at the bottom-right corner.

Above the DCG tests table, there is a search area provided for searching for a specific test. The tests can be searched by time range and whether there is comment or note associated with it.

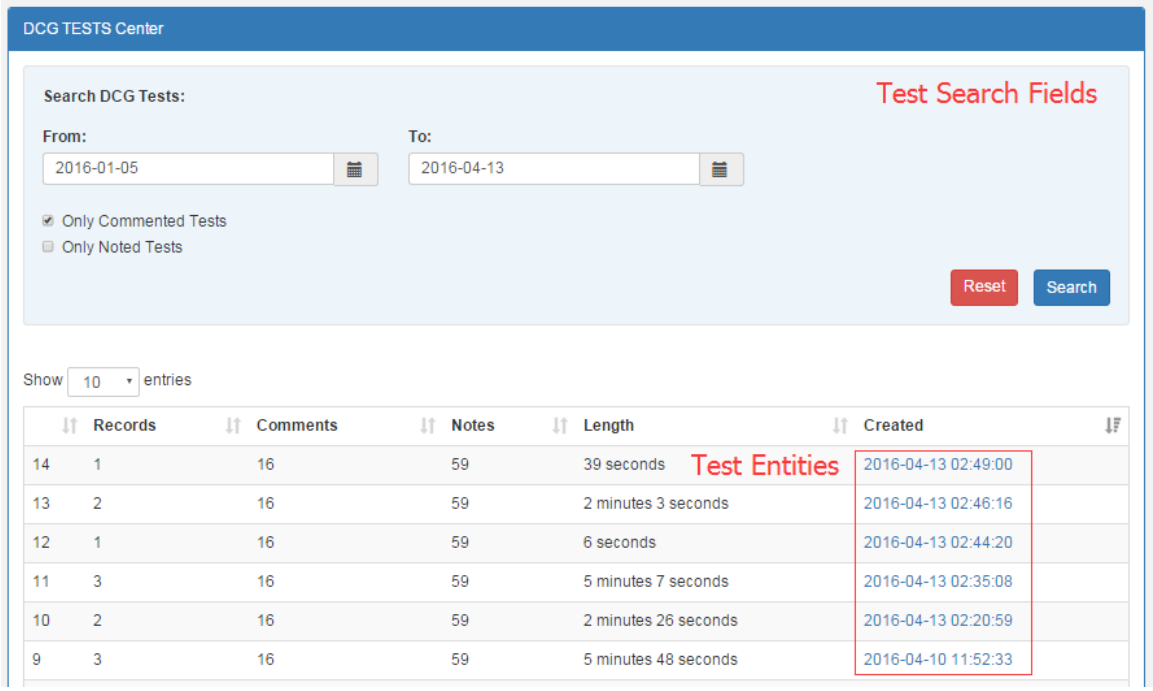


Figure 13. Test List Center

2.2.3 Doctors List

On this page, the patient users can see a list of authorized doctors. When new doctors send a connect request to the user to obtain the access to the user's DCG data, it will also show the new doctor on this page. For example in Fig. 14, when the patient user receives a connection request from a doctor, the user can choose to accept or decline the request by clicking one of the buttons in right most column. Searching is not provided because a patient is not expected to have too many doctors. Pagination is located at the bottom if a patient has more than four doctors.

Figure 14. Doctor List Page

2.2.4 DCG Test Detail

The DCG Test Detail page is a core page in the DCG web application. It contains all information related to a DCG test. On the top-left corner there is the basic information of the DCG test which is similar to the content in the Test Center. Below the basic information is a list of records in the test. Because a DCG test can be long-term, the system splits a long test into several records, each containing 10 minutes of DCG data. If a user clicks on one of the records in the list, the application will asynchronously retrieve the DCG data for that record and then render the DCG on the chart on the right side. The patient user can drag, zoom in, and zoom out of the chart. The user can also set a fixed interval for the time axis by clicking the corresponding button on the top of the chart. When a user chooses another record in a test, the chart will asynchronously reload and render a new record, and the comments associated with the record are also updated below the chart section. The user can leave a note on the record shown in the chart, by clicking the button at the bottom-right of the chart. The note will then show up in the Notes list for the DCG test at the bottom-left corner. To optimize user experience, every block on this page can be hidden by clicking on the cross at the top-right corner of the block, and then the other blocks will expand to fill the window width.

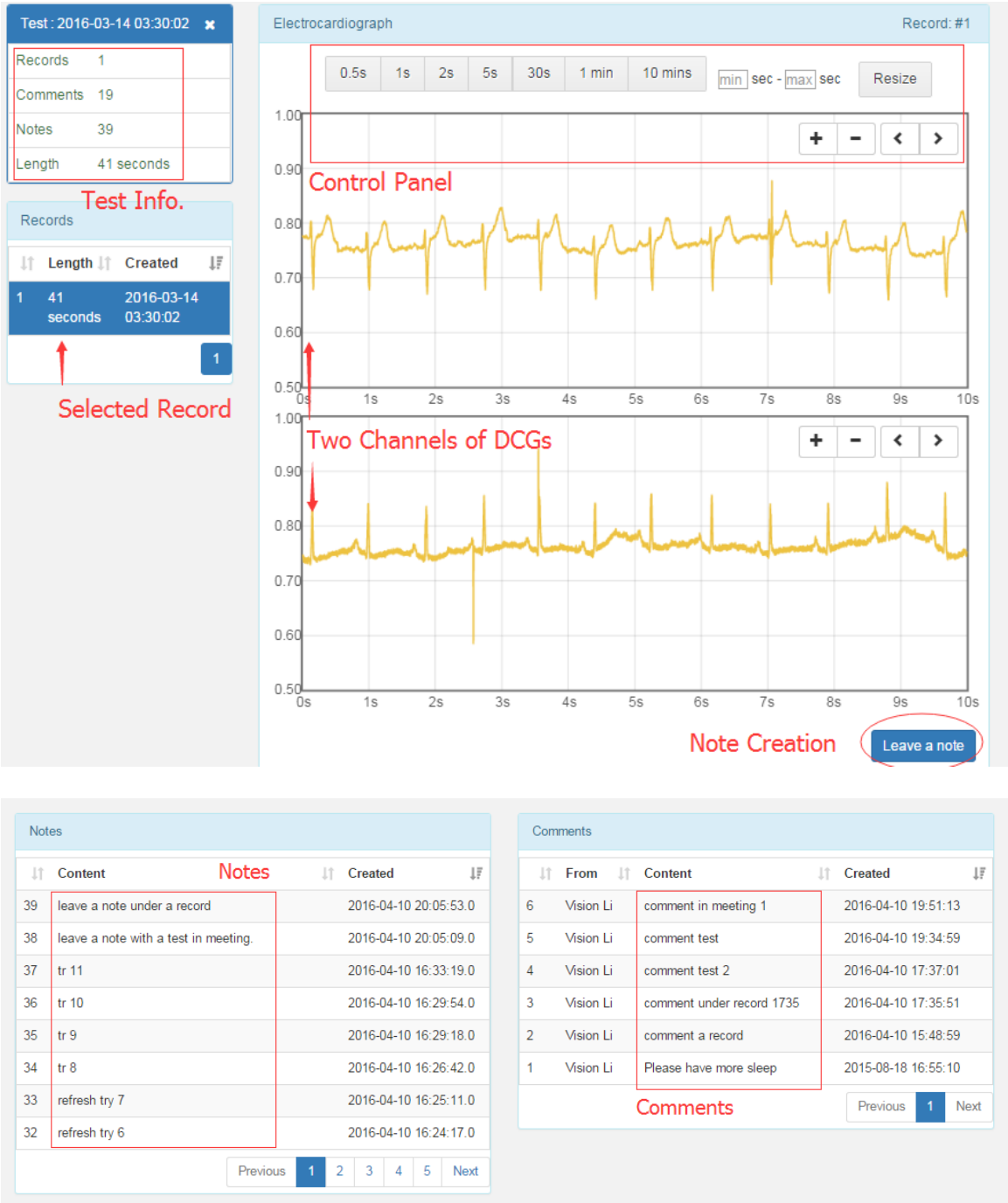


Figure 15. Electrocardiograph Detail Page

2.2.5 Comment History

The Comment History page lists comments from authorized doctors. This page contains information on all comment including the ID, the doctor's name, an abbreviate

form of the comment content, the associated record or note, and the date and time that the comment was created. A comment can either be a simple message, a comment on a DCG record or on a patient's note, as shown in Fig. 16. If it is on a record, the column For Note will read N/A. There will be a badge "New" shown with the comment if it has not been viewed yet.

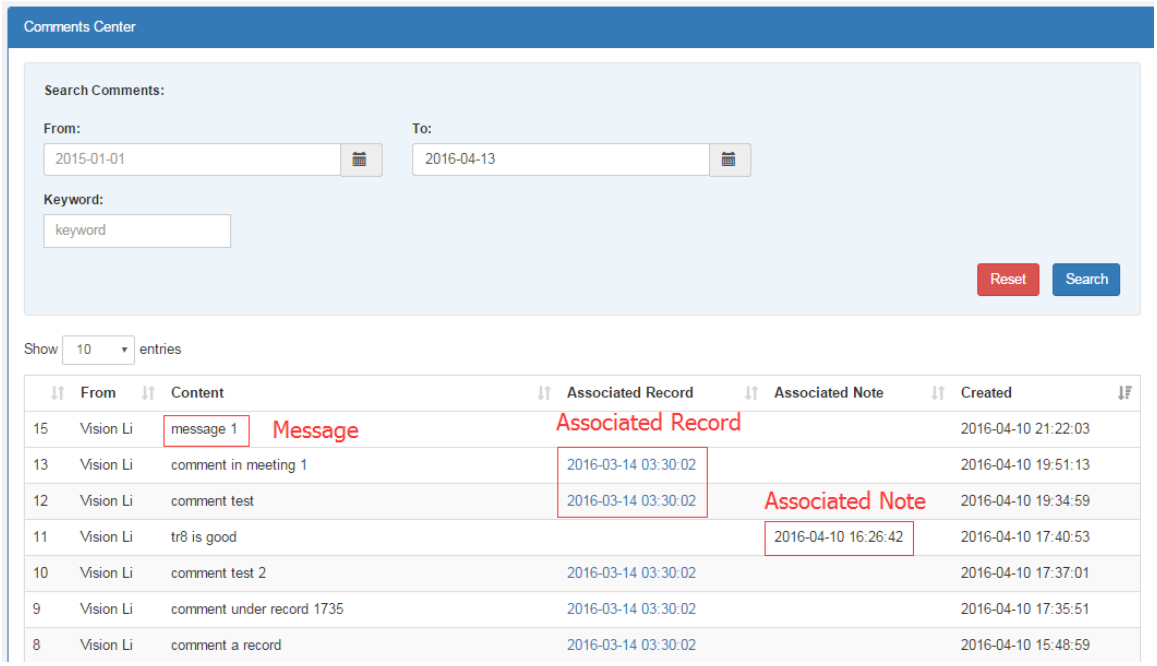


Figure 16. Comments History

2.2.6 Note History

The Note History page is similar to the Comment History page. It archives all notes a patient user wrote either on the web application or through the mobile app. A note can be associated with a specific date and time, or with a series of records from one test in that time. If it is associated with records, the number of records and the corresponding test will be shown in the row.

2.2.7 Create Note

This page enables a patient users to write a note for some event. The event can be an abnormal DCG test, an irregular medicine treatment, or some physically uncomfortable experience. Several options are provided to help the patient user specify the note category. There is a date and time selector below the text area that is used to log the time of the event. Then, the user can choose whether or not to associate the note with the DCG test. The user can choose a date for the tests, and then the application will load all tests created on that day. The user can then select one of the tests. Another HTTP request will be sent to retrieve the records associated with that test, which are listed at the bottom. The user can multi-select records associated with the note. This process of categorization helps a doctor user more easily locate the problem with a patient's heart.

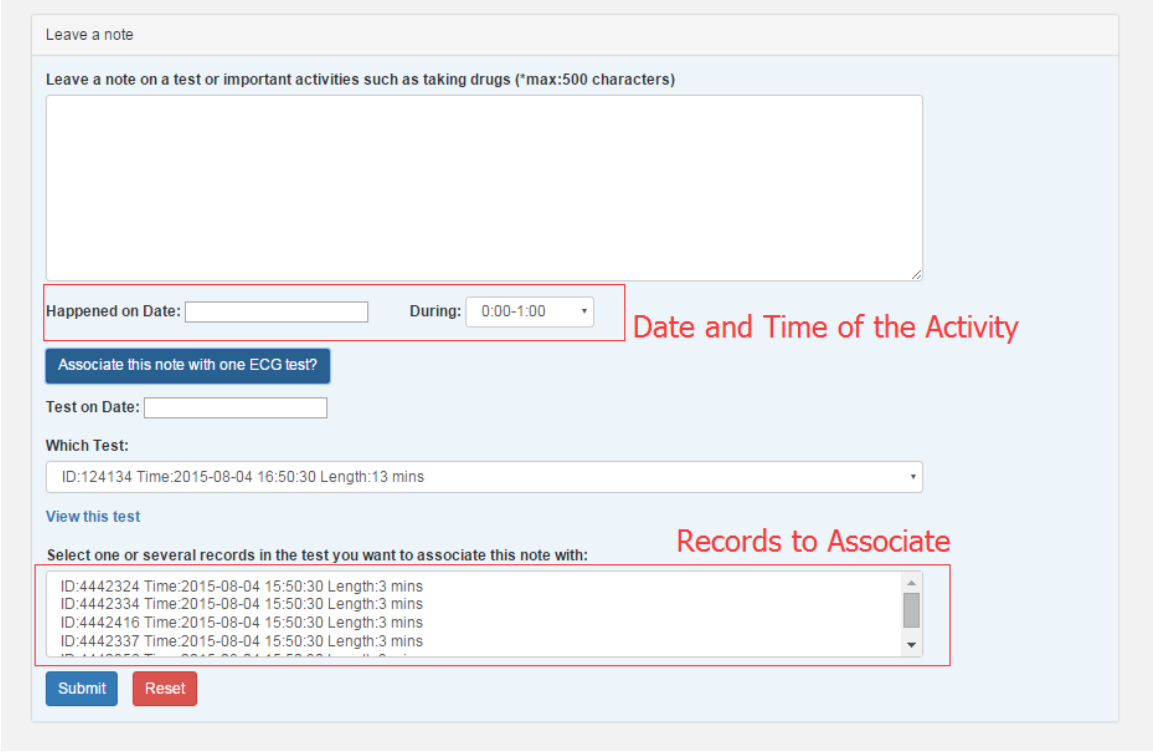


Figure 17. Create Note

2.2.8 Profile

The Profile page is a regular page containing personal information. The patient users can edit some basic information here.

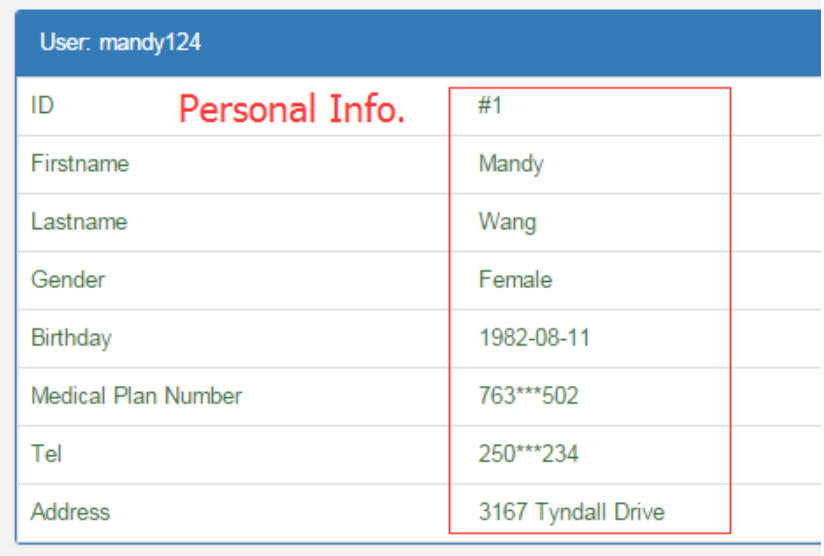


Figure 18. Personal Profile

2.3 Doctor

2.3.1 Dashboard

The doctor user has a similar user interface to the patient user. The doctor user also has a dashboard page as the index page after login. From here, the doctor user can see the latest tests and notes of all connected patients.

2.3.2 Patient List

The Patient List page shows all patients currently active in the DCG system. Each row in the table represents a patient. The last column of one row shows the relationship between this patient and the doctor user. The doctor user can view the patient's basic information and choose to send connect requests here. Once the patient accepts a request,

the status in the last column will change to “Connected”. The doctor user can access patient user’s DCG data if the relationship is connected.

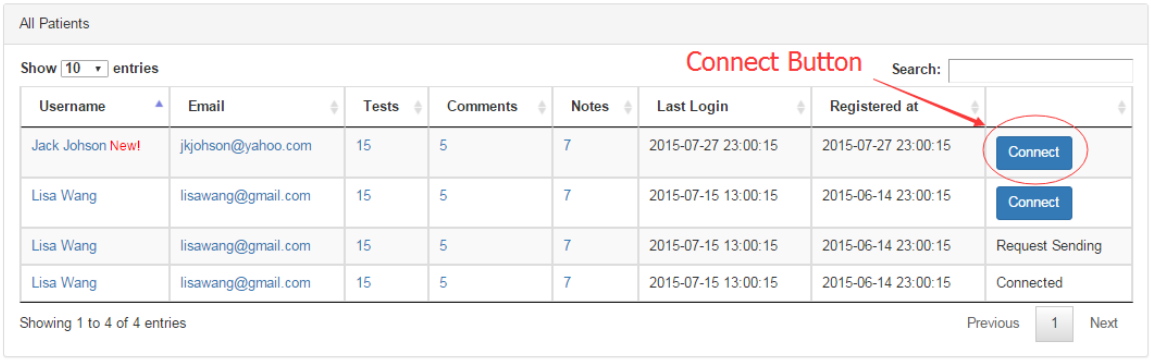


Figure 19. Patients List

2.3.3 Patient DCG Test Center

The Test Center page is almost the same as the DCG Test Center page in the patient user group. There is a table listing all DCG tests for the user on the page. However, the doctor user can view only the DCG tests of patient users who are connected. The information inside the table is the same as that in the patient’s test center.

2.3.4 Patient DCG Detail

This page is similar to that of the patient user group. The difference is that the doctor user leaves comments for the record, instead of leaving notes.

2.3.5 Comment History

A list of the doctor’s comments history shows on this page. The last column will show the status of the comment and whether or not it has been viewed by the patient. This helps the doctor determine if a contact is needed to inform the patient user of some matters that need attention.

2.3.6 Patients Notes

This page is similar to the patient's Note History page. The difference is that there is a blue button in the last column. If the user clicks one of the blue buttons, a dialog will pop up and the doctor user can leave a comment on that note. Notes search is also available to filter patients' notes.

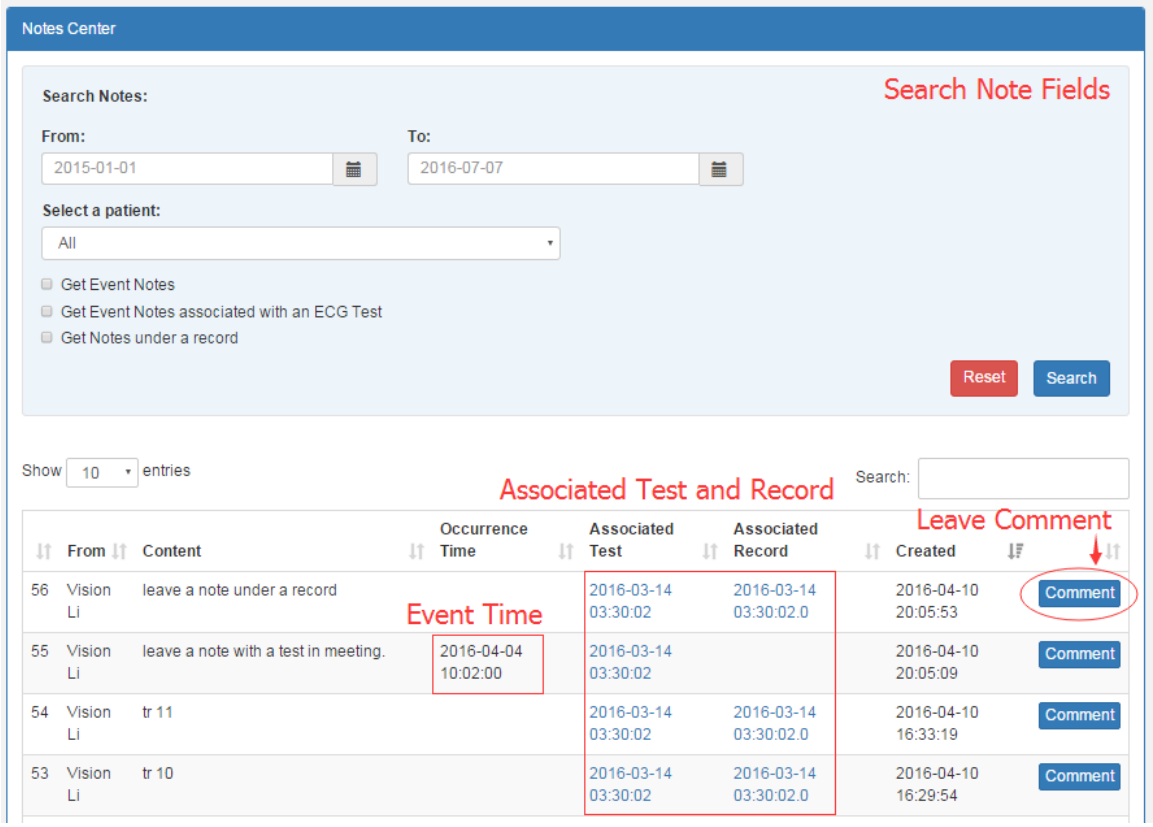


Figure 20. Patients Notes Page

2.3.7 Create Comment

This page is just a dialog. Because a comment is always under a patient's note or test, the categorization of the comment is automatic. The user can trigger this dialog either from the location shown in Fig. 21 or from the button on the DCG Test Detail page. In

the dialog, the user needs only to write content in the context area and submit it when finished.

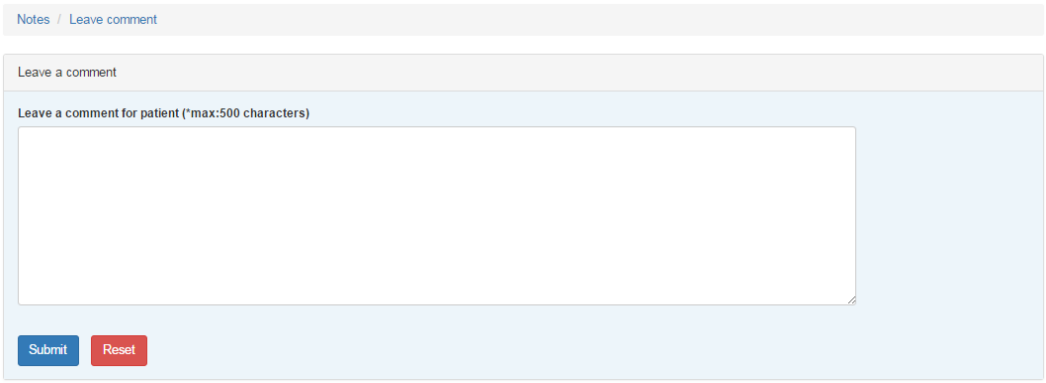


Figure 21. Comment Creation

Chapter 3 System Architecture

3.1 The wireless DCG system architecture

The DCG web application is part of the wireless DCG system. It is designed in a typical modern web application system architecture (Fig. 22). The system is made of five components: web application, mobile app, database server, database, and hardware device. The web application and mobile app belong to the presentation layer. They are responsible for the presentation of the data to the users. The database server is in the service layer. It serves other components in the system. The database server can receive the data from the hardware device through an HTTP request from the mobile app. It can also manipulate data in the database through a database connection. The web application and mobile app also send requests to the database server to retrieve data from the database. The database, as the persistent layer, stores persistent data, including user information and DCG-related data.

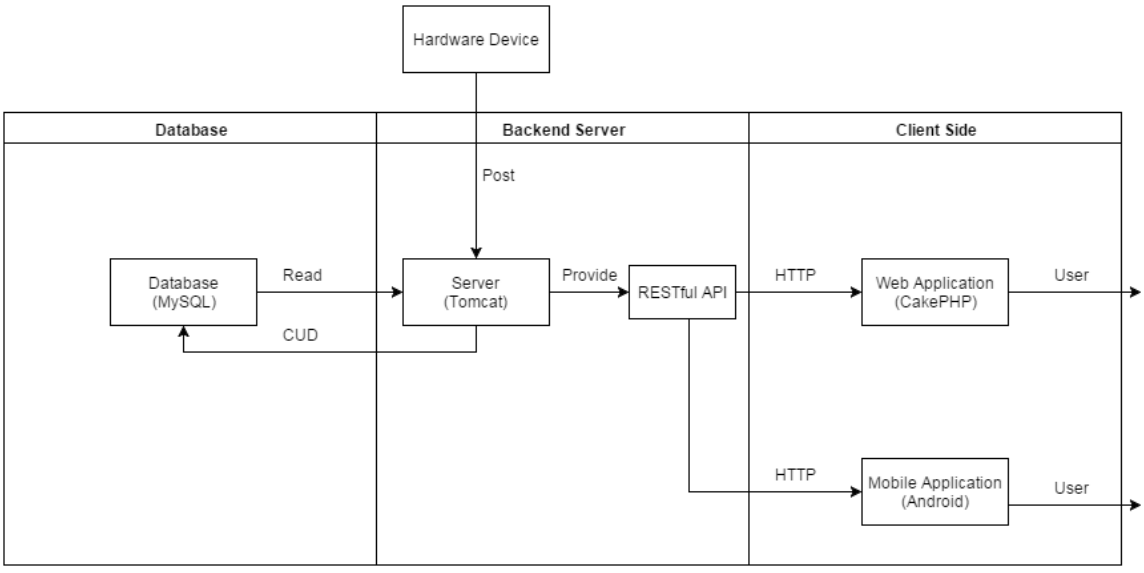


Figure 22. System Architecture of the DCG System

The workflow of the DCG system is described as follows. First the hardware device collects the raw data from the human body. The raw data is a series of voltage values which are numbers no larger than 256 that equals 2^8 (8 bit or 1 Byte). The sample rate is five milliseconds per sample Then, the mobile app receives the data from the device through BLE. When mobile connects with a WIFI, the app will transfer DCG data in a package with a maximum length of ten minutes. Therefore, the maximum size of the package is 120KB (1000/5 * 60 * 10 * 1 = 120,000 Bytes = 120KB). The database server receives data and saves it in the database. The database server then provides a set of RESTful APIs to the web application. Finally, the web application requests data from the database server through the API and then present data to the user through web pages. This workflow requires viewing authority. The web application also accepts actions from the users, such as creating a comment on a record or deleting a note. These actions which are known as CRUD [8], will edit data in the database through the API. They require the editing authority.

The web application is also called the web server as the web application itself is an Apache server. It does not connect to the database, instead it communicates with the database server. The major role of the web server is to present data to the user. Thus, the data loading performance is important. The DCG web application server uses several strategies to improve performance, which are discussed in the next chapter.

3.2 System Architecture of the DCG web application

To achieve the functionalities described in chapter 2, DCG web application needs to have the following features:

1. It should be able to send HTTP requests to API to retrieve data from database server.
2. It will have authentication and authorization mechanism to ensure security.
3. It contains caching system to cache query results.
4. It has HTML files to show contents to the user.
5. It can perform dynamic effects on web pages for good user experience for CRUD actions.

The DCG web application meets these requirements by using CakePHP MVC framework and RequireJS JavaScript module loader.

3.2.1 MVC Frameworks

A web framework can help a developer to better design web applications. It also provides many built-in helper functions for the web application development. The DCG web application uses CakePHP as the web framework, which is based on PHP programming language. The architecture that CakePHP uses to build the web application is called model-view-controller (MVC) [9], which is shown in Fig. 23.

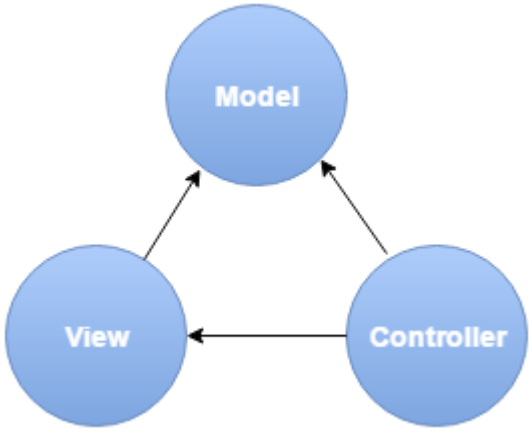


Figure 23. MVC Frameworks

The MVC architecture divides the development of a web application into three parts: model, view, and controller. Each part is responsible for some activities in the application. A model provides a representation of data, and any concrete manipulation or validation of the data can be implemented on the corresponding model; for instance, the DCG test can be defined as TestModel. Controller is responsible for processing requests from the user. It will run the functions inside and return the result to a view. Functions in the model can be called in the controller. View stores the static files for rendering web pages, including HTML, CSS, and JavasSript files. HyperText Markup Language (HTML) has become the standard way of developing web pages, and JavaScript programming language has become the main language to dynamically manipulate and update content on web pages. Cascading Style Sheets (CSS) is a common style sheet language that helps render element on web pages in a desired way, such as with a specific color, font family, font size, height, and width. View also accepts variables from the controller to update the page dynamically. In conclusion, the MVC framework provides decent support to show dynamic data on the web application and also provides clear layers and functionalities to demonstrate that changes have taken place in a layer.

3.2.2 Customized Architecture

MVC framework needs to be customized for the DCG web application. One reason is that the application does not directly connect to a database except for testing, so the model is not needed for wrapping data; another is that RequireJS is used in the application to better organize JavaScript files.

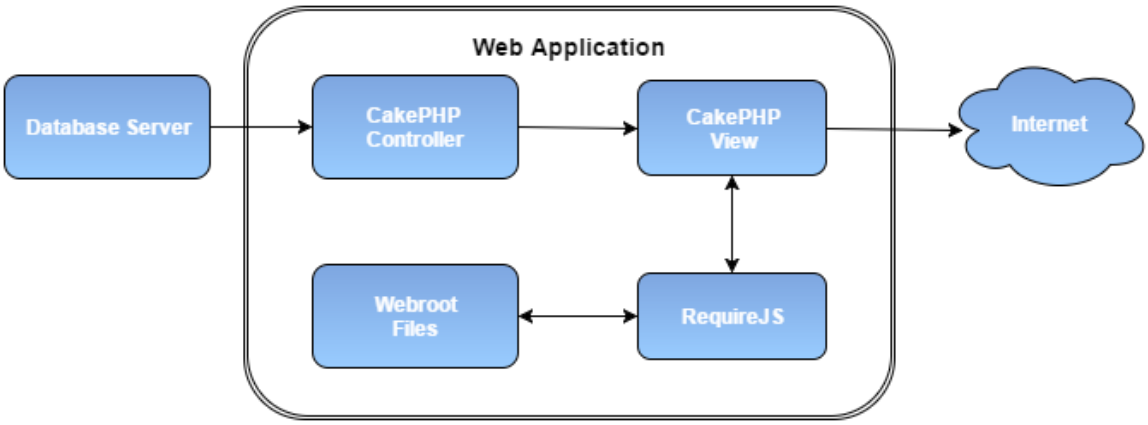


Figure 24. The DCG Web Application System Architecture

The customized web architecture is shown in Fig. 24. CakePHP and RequireJS are used in the application. The application still uses controller to retrieve data, and then data are transferred for visualization. Moreover, RequireJS is used to asynchronously load all JavaScript files from a single entry to optimize user interface operations. This customized architecture ensures the application holds the abilities of a frontend server, such as requesting and processing data, caching, etc, and also enables the application with decent user interface effects similar to a single-page application.

3.2.3 Design Patterns

SOLID principles are applied in the DCG web application to make the system more readable, maintainable and extendable. In computer programming, the SOLID principles means keeping code dry and avoiding any duplication. The patient controller and doctor controller are written in conformance with SOLID principles.

View layer layout is also implemented based on SOLID principles. The header, footer and all reusable forms, dropdowns, and two-channel DCG view are written in separate files regarded as stand-alone components. These components can be inserted into any web page so that they need to be coded only once but can be used multiple times. This

also makes the view more maintainable and extendable. For example, when a new link or tag needs to be added to the footer, only one change is needed in the footer layout instead of adding it to every web page's footer. Moreover, when constructing a new page with some existing components, one can directly load those layouts onto the new page instead of coding a duplicated one. If customization of an existing component is needed, there are two ways to do that, either by creating a new layout for this customized component or by decoupling the layout one more level, if possible, so that the common part of the component can be reused as a sub-component.

3.3 Database

The DCG system chose a MySQL database for the storage of persistent data. The DCG data are compressed and saved as a medium binary object (max size 16 million characters) in MySQL. MySQL is a database storing data in tables. "These tables are related to each other, which is called the relationship model of data" [10]. There are a number of data models in the DCG web application: the patient user model, the doctor user model, the relationship model, the DCG test model, the DCG records model, the comment model, and the note model. These models have relationships with each other. For example, one test model has many records, each record model has one or many notes and comments, and the comment model belongs to the doctor user model. These are relationship types in a relational database. MySQL is a popular relational database and is compatible with the Tomcat server. For this reason, it was chosen as the database for the application. A complete description of the database schema can be found in the appendix where each table inside is a data model. There is also an SQL file provided in the appendix to help

quickly build the schema and seed of the DCG project database in any relational database system.

3.4 RESTful API

Representational state transfer application program interface (API) is a set of accessible stateless endpoints provided by a server. In this project, it is achieved by utilizing the Jersey library. One benefit of using APIs is that web application developers do not need to know the details of how the backend server communicates with the database and the hardware device. “Using APIs can decrease the development complexity and time” [11] and also improve the level of security, as persistent data stored in the database will be decoupled from the client side. The operations and actions towards the database are hidden from the outside world. A complete API documentation is provided in the appendix, written in the popular Blueprint A1 format, and is also available to download online.

In conclusion, the MVC framework makes the application architecture distinct, and API provides a rule web application that can communicates with the database server. In the next chapter, the application performance will be discussed, which is an important aspect of web applications.

3.5 Mercurial - Version Control

Mercurial is an open source code source version control software. It is used to provide version control towards the process of DCG web application development. It is also compatible with Github server so a remote repository can be set in Github to save changes of code online. During the development, the code changes are periodically pushed to default branch of the remote repository and the repository url is in the

Appendix A. Version control enables multiple developers work on the project at same time and also provides a backup of the application.

Chapter 4 Performance

The data loading performance is critical to web applications. In general, the loading time for opening a web page should not be more than four seconds. Longer loading time on switching pages will make the user impatient to use the application. The loading time can be split into two parts: the time of transferring files from the application server to the user's browser and the time during which the web application retrieves data from the database server. The time of transferring files can be reduced by compressing and minifying them. The duration of retrieving data from the database server can be decreased by implementing caching in the web application. This chapter will introduce how to implement these methods to increase the performance.

4.1 Gzip Compression

The waiting time (transmission time) of a user opening a web page largely depends on the size of the loading files, so file compression is necessary to obtain decent performance. Gzip compression is short for GNUzip, which is a convenient compression method supported by GNU free software. The idea of gzip compression is to clean up text in the file and replace duplicate contents with pointers to the first instance of each string.

Table 4. Compression Ratio with Gzip Compression

Library	Size	Compressed size	Compression ratio
testPage.js	12.2 KB	6.4 KB	54%
leaveNotePage.js	7.70 KB	3.93 KB	51%
doctorTestPage.js	12.4 KB	6.52 KB	54%
stylesheet.css	33.1 KB	22.1 KB	33%
common.css	2.48 KB	1.62 KB	35%
normalize.css	7.61 KB	1.89 KB	75%

The table above illustrates the savings after using gzip compression on several JavaScript files in the DCG web applications. The file sizes above are reduced by 33% to 75%. Modern browsers can resolve these gzip files after downloading them and convert them back to normal files. The overall loading time thereby becomes much shorter than the transmission time without prior compression.

To implement gzip compression in the DCG web application, there is a need for configuration changes in PHP. Open php.ini with any text editor,and then change the line for output compression from off to on as follows:

```
zlib.output_compression = On
```

Then use the code phpinfo() to check whether gzip compression has been activated. If compression is not enabled in the server, then enable module “mod_deflate” to allow the compression for output files from the server. If gzip is enabled in the application, the header in the transferred files will look like Fig. 25, where the file is encoded with gzip compression.

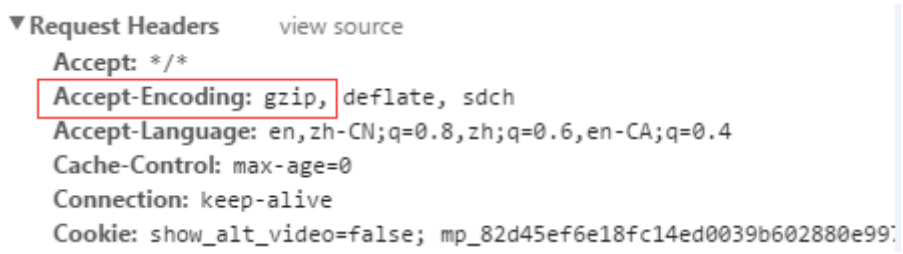


Figure 25. Encoding file with gzip

4.2 Minify Resources

JavaScript, CSS, and HTML files in a web application can be minified in production mode. Minify resources will remove comments and useless header, delete blank spaces, remove empty lines, and replace variable with shorter characters. Minified files are usually not human-readable, but using them will improve the performance. They are commonly used in the production release phase. After minification, the size of the file is usually significantly reduced. For instance, the sizes of popular libraries jQuery and Bootstrap can be decreased from 276KB to 94KB and from 118KB to 98KB, respectively.

To implement recourse minification in the DCG web application, one way is to manually copy the resource file to the online tool, download the minified file into the original folder, and load these files with the min file name extension in production mode. Another way is to download the CakePHP plugin minify; this plugin will help automatically minify resource files and also provides gzip compression at runtime [12]. The third way is to use the optimizer in RequireJS, which can minify all CSS and JS file in the entire project.

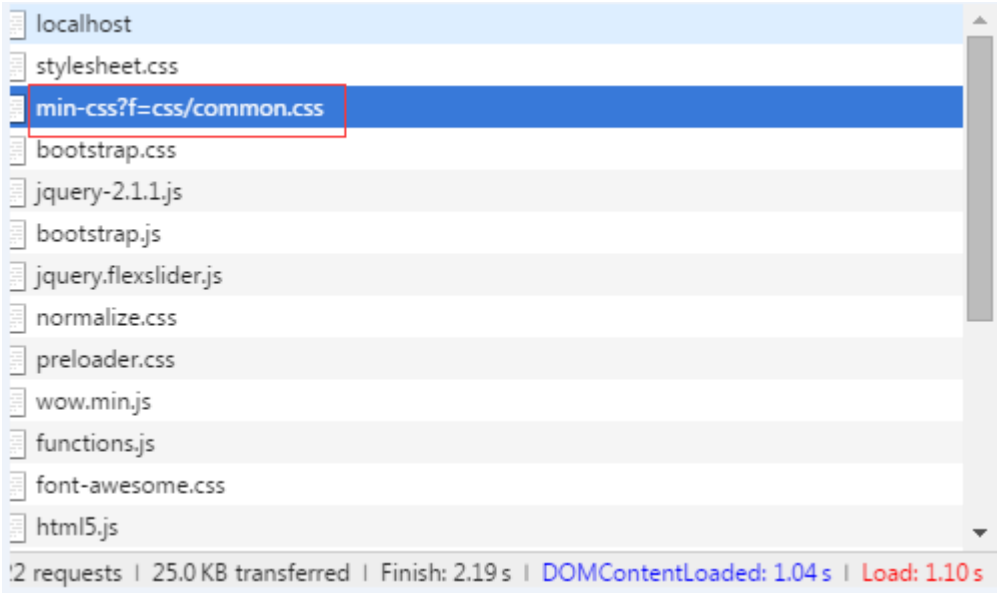


Figure 26. Loading Minified CSS File

Table 5 shows the evaluation of the loading speed of the homepage in DCG web application after gzip and minify optimization. The average speed of transferring files in continuous requests from Ashburn, US is about only 15ms.

Table 5. Optimized Loading Performance

URL	Load zone	Successful	Failed	Last avg
www.cardiacare.ca/../bootstrap-dateti...	Ashburn, US (Amazon)	576	0	2.33ms
www.cardiacare.ca/../animate.css	Ashburn, US (Amazon)	576	0	9.71ms
www.cardiacare.ca/../index.css	Ashburn, US (Amazon)	576	0	2.05ms
www.cardiacare.ca/../services-icons.png	Ashburn, US (Amazon)	576	0	2.59ms
www.cardiacare.ca/../common.css	Ashburn, US (Amazon)	576	0	4.2ms
www.cardiacare.ca/../font-awesome.css	Ashburn, US (Amazon)	576	0	22.02ms
www.cardiacare.ca/../responsive.boots...	Ashburn, US (Amazon)	576	0	2.34ms
www.cardiacare.ca/../logo3.png	Ashburn, US (Amazon)	576	0	2.23ms
www.cardiacare.ca/../index_bg.jpg	Ashburn, US (Amazon)	576	0	6.75ms
www.cardiacare.ca/../uvic-logo.png	Ashburn, US (Amazon)	576	0	2.35ms
www.cardiacare.ca/../stylesheet.css	Ashburn, US (Amazon)	576	0	5.77ms
http://www.cardiacare.ca/	Ashburn, US (Amazon)	576	0	38.28ms
www.cardiacare.ca/../dataTables.boot...	Ashburn, US (Amazon)	576	0	2.33ms
www.cardiacare.ca/../bootstrap.min.css	Ashburn, US (Amazon)	576	0	33.46ms
www.cardiacare.ca/../require.js	Ashburn, US (Amazon)	576	0	11.72ms

4.3 Caching

Caching is a cost-effective method to save loading time by retrieving an item from a local storage instead of remote. There are many types of caching that can be implemented in a web application, such as web cache, disk cache, and memory cache. Caching is extremely useful for relatively static web contents because they may not change for a long time, so every time the data that is updated from the remote storage is the same, making it suitable for long-term caching.

4.3.1 File Caching

File caching or web caching is common in web applications. By setting a cache header (Fig. 27) for a file, such as a cascading style sheet or a JavaScript file, a modern browser can read the header and choose to cache this file for a period. Whether the browser will cache the file and for how long depends on the header content. If cache-control is set to max-age:0 or no-cache, then the browser will not cache this file or will cache it for 0 second. If cache-control is set to a max-age: x, for instance as in Fig. 27, the file will be cached. Setting max-age: 1800 means that the browser will cache this file for 1800 seconds which is 30 minutes.

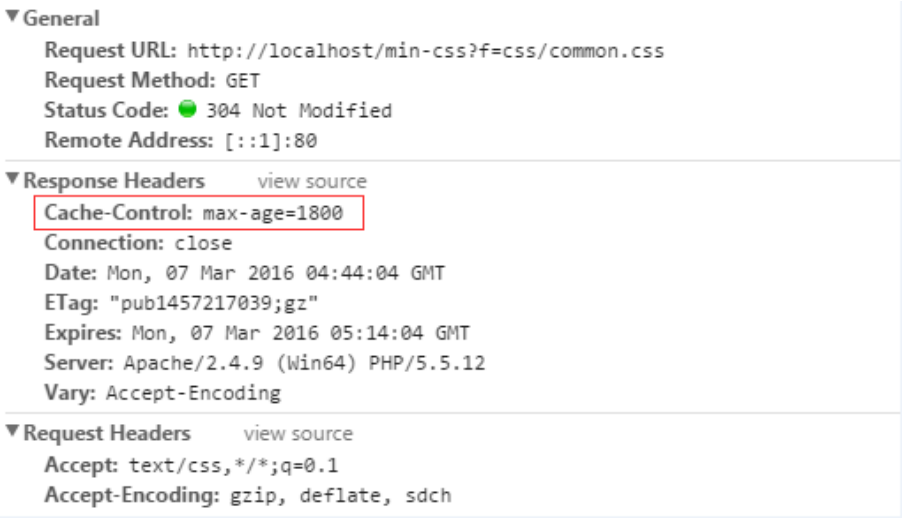


Figure 27. Set File Cache Header

It is worth mentioning that when a new release of a production is introduced, the files in the application may have already changed, but the user may still load files of the old version from the browser cache until they expire. To solve this problem one approach is to append a version suffix to every caching file, for example, appending the version code 201610103 to file planModel.js will look like /planModel.js?201610103. In each new release, the cache key will change to a new one such as 201610105. Thus, the browser

will recognize that the file has changed to /planModel.js?201610105, which is different from the previous one, and thereby load the new file instead of using the cached one.

4.3.2 Query Result Caching

Querying the database is time-consuming, so the caching of some common query results in the server is also very useful and may greatly improve performance of the application. The code snippet below shows an example of caching a query result that infrequently changes in a Get Latest Test action. It stores the ten most recent tests.

```
class Test {
   function latest($id) {
      $query = Cache::read('latest_test'. $id, 'short');
       if (!$query) {
         $query = $this->HttpGetRequest($url, $data);
         Cache::write('latest_test'. $id, $result, 'long');
      }
      return $result;
   }
}
```

However, using the query result cache in the DCG web application will be slightly different. First, the DCG web application receives data from an API request instead of a database query, so the API request result can be cached instead. Second, many data from requests are user specific, meaning that each users will obtain his or her own data, which is different from those of other users. Fortunately, user-related data have unique IDs to distinguish them from others. For instance, the request of a DCG test will return a result containing the test's unique ID in the database. Then the result can be cached with a cache key of "test_". $id, which is distinct from that of other test caches.

A large volume of caches also increases the burden to the server. To avoid this issue, a cache manager can be set to limit the volume of the cache, such as up to 1G. If the cache

is full, then the oldest entry in the cache is removed in favour of the newest, which is known as the LRU (least recently used) mechanism.

4.3.3 Memory Based Caching

CakePHP uses disk cache (FileCache engine) by default which requires expensive I/O actions. It makes more sense to use an in-memory cache to store data for the application, CakePHP has several other built-in cache engines that can provide memory-level caching, such as APC and MemcacheEngine. An example of configuring the cache class and engine is shown below.

```
CacheConfig('short',
array(
   'path' => HC . 'short' ,
    'duration' => '1800s',
    'engine' => 'memory',
));
```

4.4 Asynchronous Loading

By default JavaScript files load when users open a web page and before loading JavaScript finishes, the page will not render. "Asynchronously loading the JavaScript can enable web page to render first, and then to finish loading JavaScript files" [13]. Some JavaScript needs to load before parsing the page, but some is not needed. For example, Google Analytics file ga.js and Facebook library all.js can be loaded after rendering the page. Thus, this type of JavaScript should be loaded asynchronously so that it will not stop the page from rendering, thereby improving the loading performance.

Implementing asynchronous loading is very simple; add 'async' to the script tag, as follows:

```
<script src="testPage.js"  async></script>
```

It is important when using the asynchronous loading strategy to make sure that the script is not referred to in any synchronous loading scripts; otherwise, a js error of "attribute not found" will pop up, as the script has not been loaded yet.

Alternatively, an application can use RequireJS to asynchronously load other JavaScript files. RequireJS is a JavaScript modular loading framework that helps an application organize its JavaScript files in a modular way. The DCG web application uses RequireJS to manage its JavaScript files. The JavaScript files that need to be loaded in a global scope will be added into index.js, and the following code is used in the layout to activate RequireJS and then execute index.js.

```
echo $this->Html->script("require",[
   "data-main" => "/js/main"
]);
```

Other JavaScript files can be added to the corresponding root JavaScript file when they need to be loaded. In this way, the files are grouped into modules, and loaded asynchronously so that they do not stop the page from rendering.

4.5 Garbage Collection

"Memory leakage is a major issue for web applications as it causes latency that user can perceive" [14]. JavaScript is used to achieve dynamic web pages and interaction in a browser, and it has a built-in garbage collection functionality. However, many objects in JavaScript cannot be correctly cleaned in some cases, because when a switching page or closing dialogs, the variable reference to the object will be deleted while the object may be still referred to somewhere, such as binding in an event or remaining in a DOM. These objects have no way of being collected and cleaned, thereby causing a memory leak.

Memory usage can be profiled using browser developer tools. To analyze memory leakage in the DCG web application, open the Doctor Comment page and press F12 to open the Dev tool in Chrome, then switch to the Timeline tab, check the Memory checkbox, and click the record button to start recording. During the recording, repeatedly open and close the Comment Creation dialog several times and then finish recording. The Dev tool will record the memory usage during the period, and the result is shown in Fig. 28.

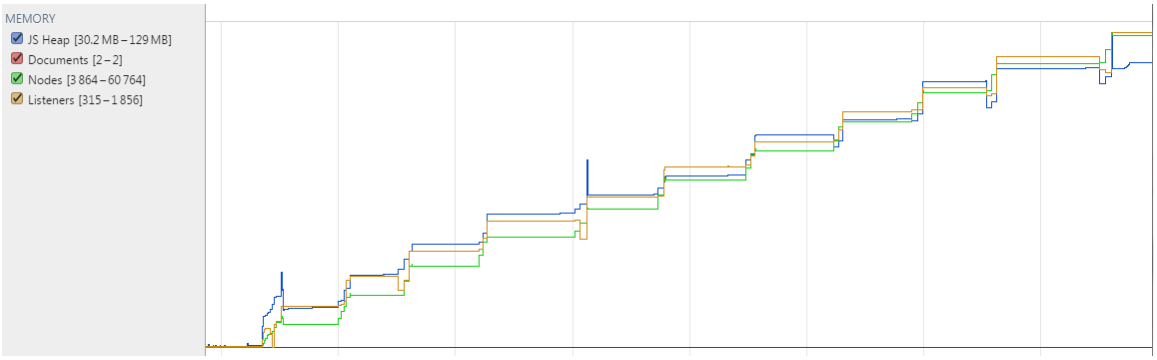


Figure 28. Memory Usage before Applying Object Cleaning

As time goes on, memory usage will gradually increase if the user stays in the application. This will cause the user's browser or device to respond slowly to new requests, and the screen will appear frozen for a while every time a new action comes in from the user which significantly affect performance.

To prevent memory leakage in the web application, the objects in JavaScript should be cleaned from time to time. First, if it is an event bind, remember to unbind it using a clean method at the end of the business logic model:

```
$(element).bind( "click.testEntry", showRecords);
clean: function() {
   $(element).unbind("click.testEntry");
}
```

Second, if a “viewmodel” for a view is initialized in the business logic model, the objects in the viewmodel cannot be garbage-collected because the business logic model holds a reference to the view model. The solution is very simple: the viewmodel reference should be removed from the model so that the garbage collection can collect the objects in the viewmodel.

```
clean: function() {
   this._viewModel = null;
}
```

After implementing unbind and reference removed, the memory usage of repeatedly opening and closing the Comment Creation dialog is shown in Fig. 29. The memory usage can be greatly reduced by closing the dialog.

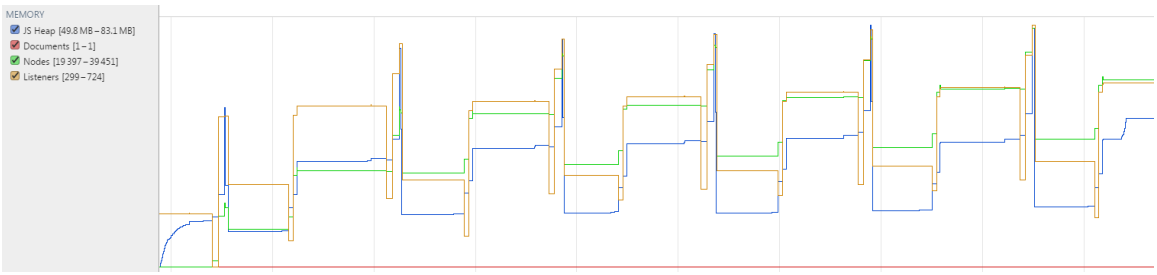
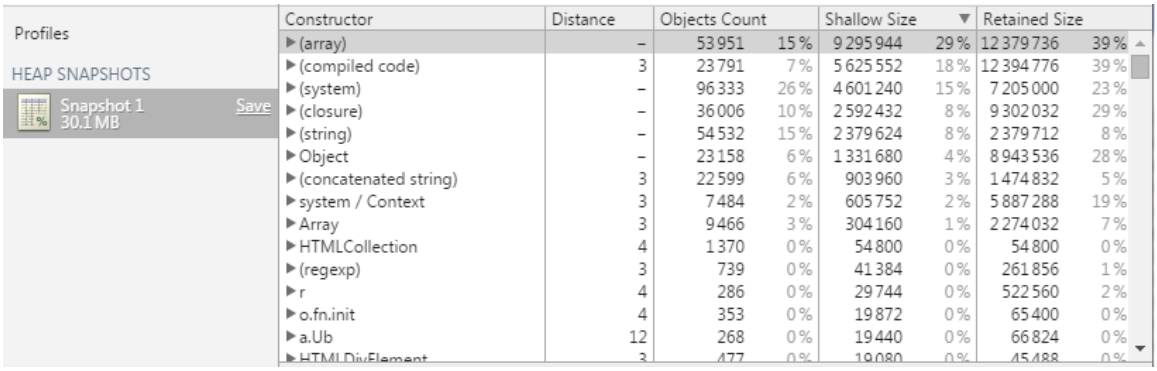


Figure 29. Memory Usage after Applying Object Cleaning

Another useful tool in Chrome is profiling, which can take a snapshot of the current JavaScript heap and show which object in the heap occupies the most space. Once again, open the Comment Creation dialog and then open the Dev tool, switch to the Profiles tab, choose Take Heap Snapshot and take a snapshot, a snapshot will be generated, as shown in Fig. 30. Arrays occupy the most space in the heap. The table can be expanded to find which array occupies the most space, thereby helping to locate the positions of memory leaks.



Constructor	Distance	Objects Count		Shallow Size		Retained Size	
(array)	–	53 951	15 %	9 295 944	29 %	12 379 736	39 %
(compiled code)	3	23 791	7 %	5 625 552	18 %	12 394 776	39 %
(system)	–	96 333	26 %	4 601 240	15 %	7 205 000	23 %
(closure)	–	36 006	10 %	2 592 432	8 %	9 302 032	29 %
(string)	–	54 532	15 %	2 379 624	8 %	2 379 712	8 %
Object	–	23 158	6 %	1 331 680	4 %	8 943 536	28 %
(concatenated string)	3	22 599	6 %	903 960	3 %	1 474 832	5 %
system / Context	3	7 484	2 %	605 752	2 %	5 887 288	19 %
Array	3	9 466	3 %	304 160	1 %	2 274 032	7 %
HTMLCollection	4	1 370	0 %	54 800	0 %	54 800	0 %
(regexp)	3	739	0 %	41 384	0 %	261 856	1 %
r	4	286	0 %	29 744	0 %	522 560	2 %
o.fn.init	4	353	0 %	19 872	0 %	65 400	0 %
a.Ub	12	268	0 %	19 440	0 %	66 824	0 %
HTMLDivElement	3	477	0 %	19 080	0 %	45 488	0 %

Figure 30. Profiling Snapshot in Dev Tool

Using gzip compression, file minification, caching and garbage collection, performance is enhanced greatly so the user will not feel blocked when opening web pages in the application. In the next chapter, security, how to secure the DCG web application, will be described.

Chapter 5 Security

As web applications become more powerful, they possess more interfaces and their security vulnerability increases correspondingly. It is vital to keep users' sensitive data secure to prevent interception by hackers or malicious users. In the DCG web application, there are several potential threatens described as follows:

1. A normal user may be able to see a patient's DCGs without proper authorization.
2. A hacker may steal all users' data in the database by using SQL injection.
3. A hacker is able to filch user's username and password under a cross-site scripting attack.
4. A hacker can get user's credential in browser to log in to user's bank account with CSRF.
5. A malicious user can still see user's information after user logs out by using the same computer and clicking Back button in the browser because user's session is cached.

This chapter will introduce several security strategies that the DCG web application implements to defend against potential attacks described above.

5.1 Authentication

Authentication is a common way to protect user data. Only authenticated users can use the patient or doctor management center in the application. To avoid fake account registration, an identifying image code can be used to distinguish between a human user and a computer program. A verification email is also necessary for identification; a user needs to click an activation link in an email to obtain full access to an account. Patient

accounts in the DCG web application may not need to sign up through the application, as they should already exist when customers start to use the hardware device to log DCG data. A flow chart is shown in the appendix. It is very clear that a guest user can access only the public static pages in the application, while authenticated users can access other personal pages containing DCG data.

The CakePHP framework has a built-in authentication mechanism. In PatientController, there is an array of allowed page actions in the beforeFilter method including all page actions that need authentication in the controller. Usually, an action or method in a controller includes all business logic for a specific page, and a beforeFilter method will run before any method is triggered in the controller. If the method is included in the action array and the user does not log in, then the user will be redirected to the login page. If a user has logged in, a session will be created for this user, including a session_id, which is a sixteen-digit string generated every time by the server. When a user logs out, the web application will send another API request to the server to expire the session_id. This prevents a hacker from hacking into the system by stealing a user's session_id with corresponding username and password, making the authentication process more accountable.

5.2 Authorization

Authorization helps determine the specific access right of every user to pages and functionalities, which authentication cannot. In some cases, an area of content or a function should be accessed only by a group of people. Not all authenticated users in the application, such as editing the post of a user; or viewing the personal information of a user. In the DCG web application, patient personal information containing the birth date,

name, health card number and DCG data is considered sensitive data. These pages should be accessible only to authorized users, such as the patient himself/herself, and accepted doctors. The authorization process is implemented in the isAuthorized method, as shown below. In the post controller, if a user is authorized for specific methods, then the function will return true, otherwise the action is denied.

```
if ( $this->action ==  'gettest' ||  'getrecord') ) ) {
   $testId = $this->request->data[“test_id”];
   if ($this->Test->isAuthrozied( $testId, $this->Auth->User->id)) {
      return true;
   }
}
```

5.3 SQL Injection

SQL injection is a code injection method that can be applied in web applications. Usually, there are multiple input fields in a web application. The user can input some strings into these fields to send data information to the application. For example, in a login form, when the user inputs a username and password and clicks the Log in button, the data will be sent to the application. Then the application will run a query to check whether the username and password are valid. Commonly a query is a combination of a static SQL and the users input string, for instance,

```
$query = "select * from user where name='"+ username + "'and pass='" + password + "'";
```

This query works fine when the user inputs a normal username and password. However, if a user either deliberately or accidentally inputs a quotation mark after the username. It will cause the statement to change to where name= ‘’. Now the query will select all users from the database, which will lead to the exposure of all users information in the application.

Fortunately, the DCG web application does not need to defend against SQL injection because it is decoupled from the database. However, a string escape strategy has been implemented in places that will take users' input as data in the DCG web application to ensure that there are no potential malicious input threats, such as XSS. An escape for inputs in CakePHP can be typically written as:

```
$this->Form->input('username', array('escape' => true));
```

The special characters in this input will be escaped when transferred to the controller. For example, the input "marc'" will be escaped to "marc&apos".

5.4 Cross-site scripting

Cross-site scripting is another type of attack on input fields. Attackers inject a client-side script into input field to let web application run the malicious script. The code snippet below shows an example of inserting JavaScript code into the search field. If the web application does not implement defence, when enter is pressed, this script will run and an alert will pop in the window. In practice, an attacker can use cross-site scripting to change his profile to a fake login page, and let a normal user open his URL. The user may assume it is a normal login due to session expiration and input his username, and password on the page. Upon hitting the submit button, the user's username and password will be sent to the attacker's web application.

Preventing cross-site scripting in the DCG web application also requires escaping input fields, the same as approach as in 5.3. For instance, if an attacker inputs a script:

```
<script>alert("Hacker")</script>
```

Escaping will sanitize the input and produce the escaped string:

```
&lt;…&#039;Hacker; &#039;…&gt;
```

Special characters are replaced with HTML entities, which are not harmful to the web application.

5.5 Cross-site request forgery

"A cross-site request forgery (CSRF) makes use of user's credential information to maliciously force a user to submit a hidden request" [15]. The cross-site request forgery is usually hidden in a link in an email or message. If a user clicks the link, it will force the user to send an unwanted request to a web application using authenticated credentials, such as for a money transaction or a password change, etc. It is difficult for the web application to distinguish between a request from the CSRF and a normal request from a user because both have valid credentials.

In the DCG web application, cross-site request forgery can be prevented by enabling CSRF protection in CakePHP. The mechanism of CSRF protection is to set unique tokens on forms with expiration. A token is generated when the user renders the page and expires once the user submits the form. The use of the token is to ensure that the form is submitted only once otherwise it expires. This can prevent the cross-site request from submitting the same form to the application.

```
$configuration = array(
    'csrfExpires' => '+3 days'
);
```

5.6 Others

The security level of a web application has greatly increased with implementing the strategies introduced above. They are common vulnerable areas in a web application while there are several other techniques that are good to have and can help further protect a web application in safety.

Restricting HTTP method, most times every method is aimed to receive no more than two kinds of request, it is good to restrict HTTP type that a method can receive, for example, in CakePHP framework, the HTTP type can be restricted to get method by adding requireGet to the method.

```
SecurityComponent :: requireGet()
```

Restrict actions to SSL, actions in CakePHP can be restricted to only accept SSL request by adding this before the action:

```
SecurityComponent :: requireSecure()
```

To activate form tempering prevention mentioned above, setting $validatePost to be true as illustrated in the following code. Security component will validate the form every time a user submits before executing other actions.

```
SecurityComponent :: $validatePost() = true;
```

Log out and then hit back button is a common bug in web application, when user logs out, the session will be cleared, but the cache for the previous pages are still in browse. If user clicks back button, the user will see previous page which requires authentication. This is extremely harmful if it is a public computer, malicious user can get access to a user's private page on it. This is also the reason why most bank account systems do not allow user to use back button.

Logout and back button bug also exists in DCG web application, to prevent this bug, we declare cache setting to no-cache and 0 seconds expiration specifically for login and logout page, as shown in the following snippet.

```
setHeader("cache-control: max-age: 0, no-cache");
```

Mock web application is not so harmful but unofficial, someone can develop a totally same application with DCG web application if he knows the API documentation of DCG

project, the database server cannot distinguish an API request from official application or not, to manage the API access and copyright, an api_key can be implemented in API requests, api_key is a series of characters, it is privately shared between database server and web application, every time web application sends request it has to include the api_key and only valid api_key request will be accepted by database server, otherwise will throw an unauthorized error, in this way it can ensure only official web and mobile application can access API, thus improve security.

Chapter 6 Scalability

The DCG web application works well when the dataset is small, such as when there are few concurrent users on the application and the daily page view is not high. When the application has a large user base, the request frequency from the client side will increase substantially, and scalability may become an issue for the application. The average response time may become too long for normal usage. Thus, strategies and suggestions will be described in this chapter to overcome these challenges in the future.

6.1 Vertical and Horizontal Scaling

There are two ways of scaling the application system, vertically and horizontally. Vertical scaling means adding more CPUs to a single computer, while horizontal scaling means adding more computers at the same level, the later is preferred because hardware is not expensive nowadays. When the threshold increases in the early stage, the CPU, memory and bandwidth of the DCG web application can be upgraded to meet the needs, such as by upgrading to eight/sixteen-core CPU with a 10G/s to 100G/s bandwidth. When the threshold is too large for a single server, a distributed system can be implemented to handle the problem.

Fig. 31 illustrates a test of the concurrency ability of the DCG web application by using an online test software. The virtual active users are mocked to visit the homepage of the DCG web application. The number of concurrent users increases from 1 to 100 as time goes on. The loading time stays at about 100ms, meaning the current bandwidth, CPU, and memory of the application server are capable enough to handle 100 concurrent users.

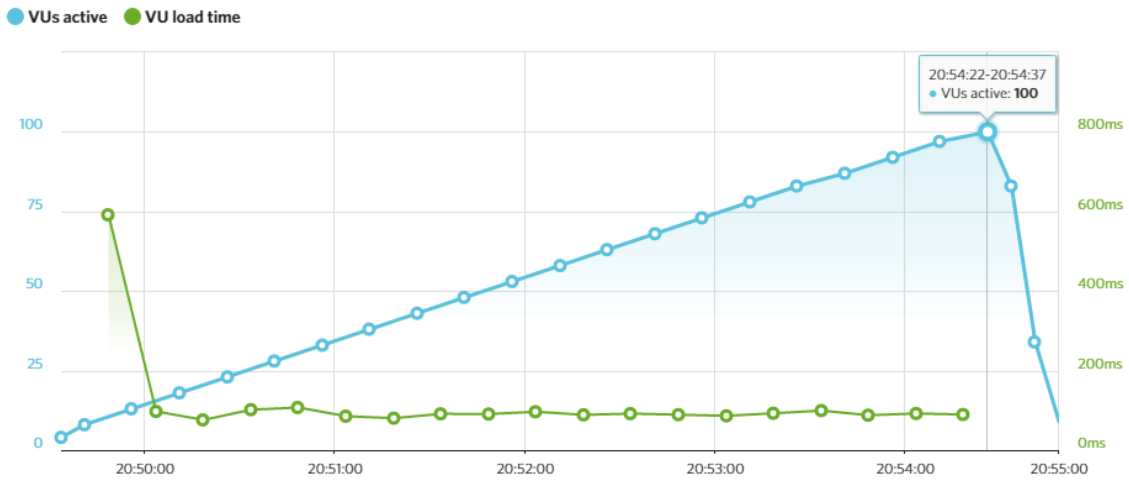


Figure 31. DCG web application high concurrency test

6.2 Load Balancer

The load balancer is widely used to resolve high concurrency issue. As the bridge between requests and request nodes, it can reasonably allocate requests to idle nodes to take full advantages of resources to respond to requests in a timely fashion. Fig. 32 shows a replicated system with a load balancer in front. Load balancer resolves multiple requests which send to the server at the same time, and then dispatch them to idle client server respectively.

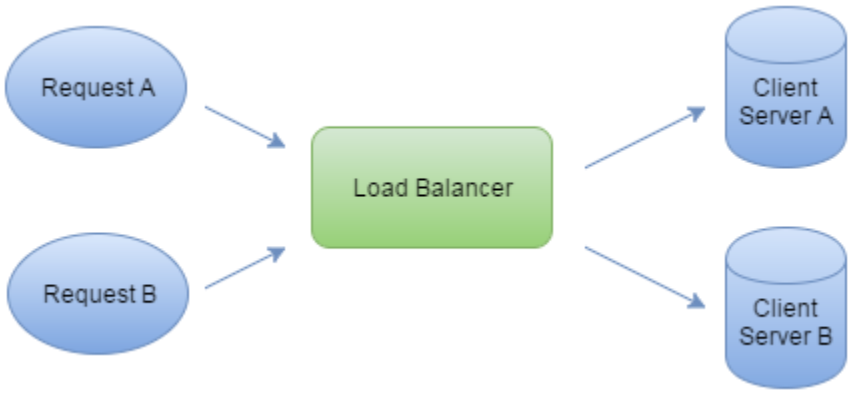


Figure 32. Load Balancer

To prepare for a potential large number of concurrent requests, a distributed system architecture can be implemented for the backend server, which means that there may be more than one backend server for scaling purposes in the future. At that time a load balancer is needed to balance the load between two or more servers and make the best use of the available resources. The DCG web application uses an Apache server which can enable a proxy module (mod_proxy_balancer) to achieve load balancing. Here is a simple example of how to implement load balancing in an Apache front-end server.

```
<Proxy "balancer://cardiacare">
BalancerMember "http://10.0.1.1:8080"
BalancerMember "http://10.0.1.2:8080"
</Proxy>
ProxyPass "/test" "balancer:// cardiacare "
ProxyPassReverse "/test" "balancer:// cardiacare "
```

6.3 Scaling Database

In general, a query to the database is heavy and can easily reach the limitation bar. The database can be scaled to speed up the query request to the database. Similar to a server, a database can be replicated by electronically copying from one to another. This ensures that users can access data when one database i/o is busy.

When data clusters become large and complicated, such as when the DCG data grows immensely, it is good to split the database at that time. One set of databases saving only data for the user model, and another set of databases responsible for DCG-related data. This can enhance the DCG data retrieving speed. The database can also be split by regions, such as one set for the United States and one set for Canada.

Indices can help improve the query speed on the database. For the DCG web application, building indices on the patient_id in the patient-doctor relationship table on the DCG test_id in records table, comments table and notes table will increase the

searching speed when the user opens the test detail page that will query for the test. test_id is associated with data, including records, comments, and notes.

6.4 Alternatives

The DCG web application uses MySQL as the database because it is compatible with database server. MySQL is free and easy to deploy while SQL server has a better response time, when a query has many joins and sub selection [16]. The following table shows the time spent on different relational databases to execute a query with table join actions on a large dataset:

Table 6. Query Latency in Relational Databases Data From [16]

MySQL	Postgres	SQL Server	Oracle
1.7s	35s	0.4s	0.6s

It can be seen that Microsoft SQL server outperforms the other databases when handling joining queries, and Microsoft is going to release Linux version SQL server in 2017 which will enlarge the user group one step further. Microsoft SQL server can replace MySQL in the DCG database server to provide better query performance.

In addition to database, there are different types of web application servers available. Table 7 shows a detailed comparison of these servers.

Table 7. Web Application Servers Throughtput Data From [17]

	Apache	Nginx	Lighttpd	G-WAN	IIS
CPU(%)	25	21	8	50	8
Throughput (req/sec)	14000	23000	27000	48000	50000
Response Time(ms)	6.3	5.6	3.6	3.4	2.8

Apache, Nginx, Lighttpd and IIS can saturate the test network (930 megabits per second). IIS and G-WAN exhibit superior throughput. Moreover, IIS has a decent response time, which will effect the performance when the user has a high ping. IIS can be considered as an alternative for the DCG web application in the future.

Chapter 7 User Experience

The user experience represents the satisfaction and degree of comfort of utilizing a web application. It depends on many factors, such as the usability and the consistency. The DCG web application should reach the following standards in terms of user experience:

1. The user should be able to use the application to see DCGs on web page.
2. The application should cost minimal time for a first-time user to learn to use .
3. The user should not perceive much difference when using the application on different browsers or different pages.
4. A user can easily search and find the DCG web application from social media or search engines.

In this chapter several strategies will be discussed to realize a optimized user experience for the DCG web application.

7.1 Usability

The user experience should follow the basic principles that usability always comes first. All buttons and functions in the web application should be simple, convenient and easy to use. The user should not need to spend much time learning how to use the web application functions, which is also advocated by the concise creed style. The web page should not contain too much information, but instead, should present only the main content in the middle horizontally and vertically. The navigation bar is at the top, the bottom has short links and copyright information, and the rest of the page should be kept blank.

7.2 Browser Compatibility

Browser compatibility is the hardest part of web development in terms of the user experience. Modern web browsers have evolved to be able to resolve most cascading style sheets, but there are still some styles that lead to the different presentation effects on different browsers. Some are so incredibly different from each other that will potentially influence the usage of the web application. Consequently, it is necessary to manually test the styling result of the web page on popular browsers (IE11, Chrome, Firefox, and Safari on Mac) to make sure that the application maintains good browser compatibility.

One way to check whether the styling will be compatible with browsers is to use the online tool at http://caniuse.com. Fig. 33 shows that the flex module functions well on most browsers but it is not fully compatible with IE.

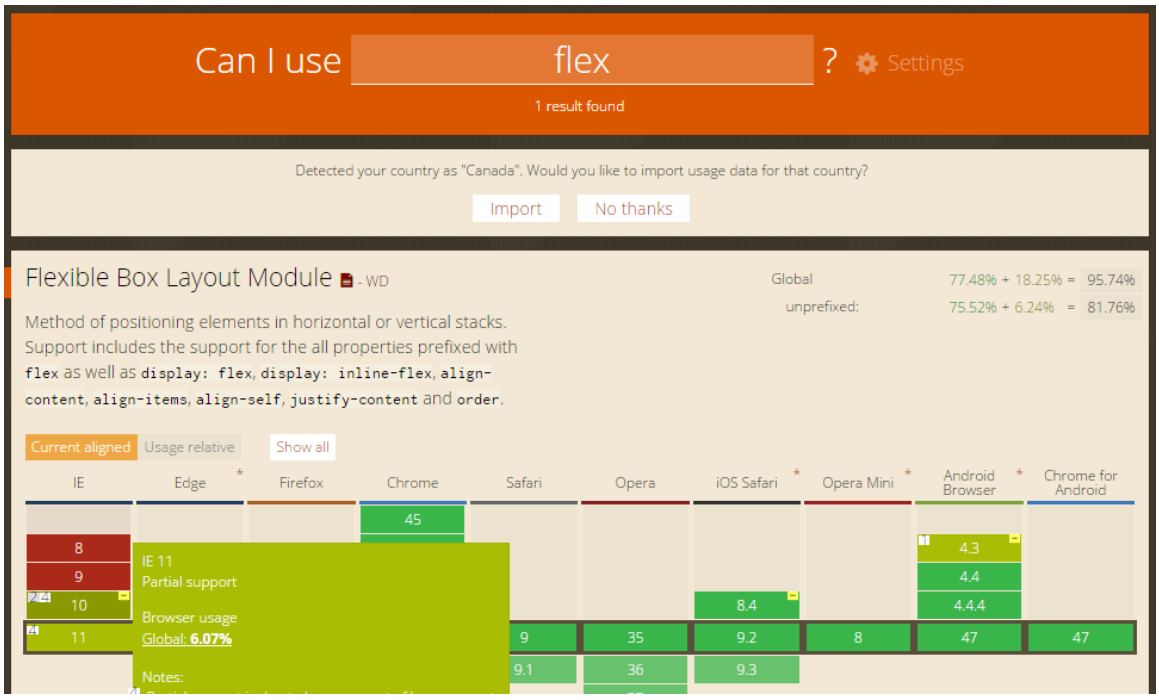


Figure 33. Test flex Styling Compatibility

Using flex module may cause an issue on IE frequently, especially when mixed with min-height or max-height usage. The developers should be careful when using uncommon styling and always to check the behaviours of styling on different browsers.

7.3 Component Consistency

Creating consistency and using common user interface elements makes users feel comfortable and quickly acclimated to the web application. In an application, a consistent font size and color should be used. For the DCG web application, the default font family is "Ubuntu, 'Helvetica Neue', Helvetica, Arial", and default font-size is 13 pixels. A consistent header and background color should also be chosen. The background color for the footer in the DCG web application is RGB(51, 51, 51), and buttons (the sign in, sign up, and register buttons) of proceeding actions on different pages should be the same styling illustrated in Fig. 34.

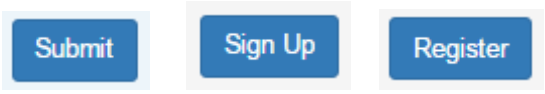


Figure 34. Consistent Action Buttons

The overall styling of web pages should not change much before and after login, otherwise, it will make the user uncomfortable and feel as if he is logging in to a different web application. #fff is still the background color for the main content on the page. In addition to consistent button, the styling of the input fields, tables, and lists should also be kept same across pages. They should be clean to view and easy to click and manipulate, Fig. 35 shows an example of note creation form in note page. The fields are listed one by one vertically. Create button keeps same styling as the Login button. The text area keeps the same styling with that on the Comment page.

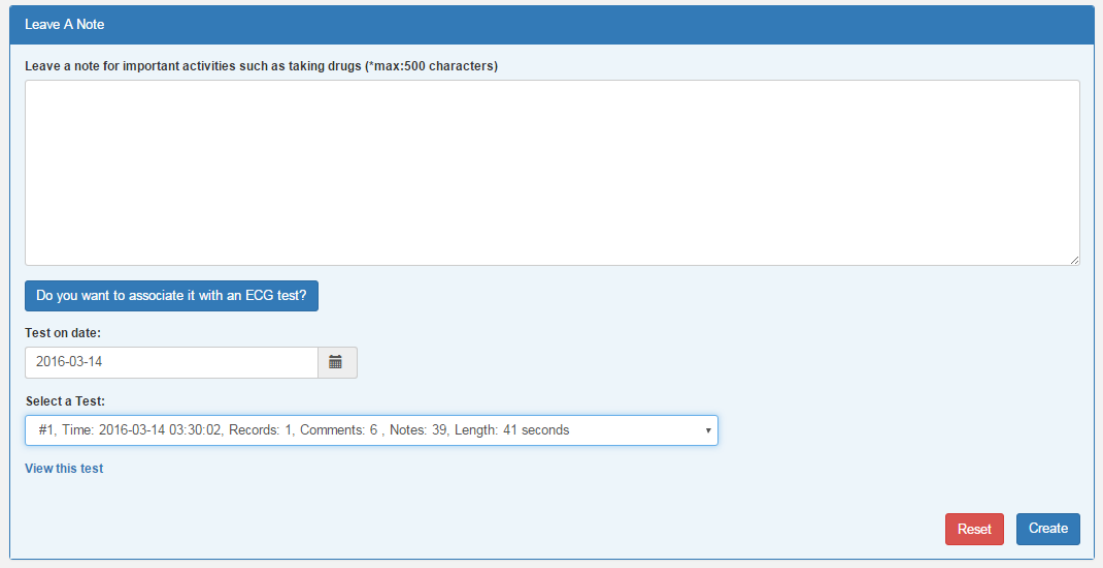


Figure 35. Note Creation Form

Defective user experience leads disruptive effects to the application, while it also decreases page rank of the application. Search engine optimization strategies are introduced in the following paragraphs that will increase page rank to make up insufficiency of defective user experience.

7.4 Search Engine Optimization

Search engine optimization (SEO) affects the ranking of a web application in a search engine. Several strategies can be implemented to optimize the web application to obtain a higher ranking in the search engine (such as Google). High page rank helps boost the DCG web application and let users more easily find the application on the Internet.

7.4.1 Page Ranking Algorithm

A graphic example of calculating page rank is shown in Fig. 36. The circles represent web pages. The diameter of them stands for the value of page rank for that page, and the arrow with direction means the link from one page to another page. Page B gets many

incoming links from other pages, so page B has a high page rank. Page C doesn't have many incoming links, but it still gets a high page rank because it is referred as an outbound link by a high page rank page, page B. To put this into mathematic expression. The page rank algorithm will be introduced in next paragraph.

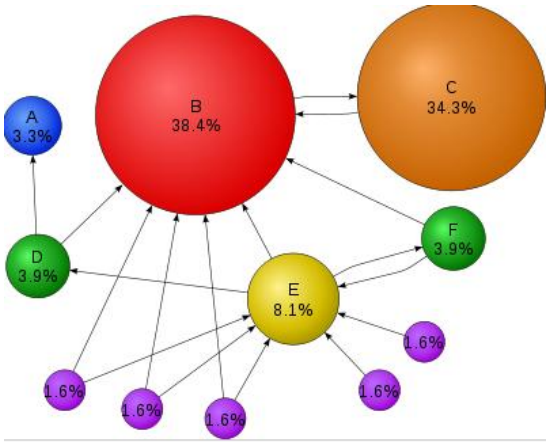


Figure 36. Mathematical Page Ranks for a Simple Network Reprint From [18]

Popular search engines like Google and Baidu have complicated algorithm to calculate page ranking, Google hires hundreds of software engineers at Mountain View headquarter just aiming to enhance the search engine performance and accuracy. Page ranking represents the probability of ultimately stay on this page after series of clicking links and highly depends on the incoming links on other pages, it keeps changing according how many pages in scope and links on pages, for example, when search "orange", there are three page as search results, x, y, z. If there are no links on pages then

$$P(x) = P(y) = P(z) = \frac{1}{3} = 0.33 \qquad (7.1)$$

Suppose instead page y links to page x, and page z also links to page x, then page rank for page x will increase, which is contributed by the other two pages, the page rank of page x will be:

$$P(x) = \frac{1}{3} + \frac{P(y)}{1} + \frac{P(z)}{1} = \frac{3}{3} = 1.0 \quad (7.2)$$

Because all incoming links belong to page x, page x holds valuable content that other two pages need to share. Users viewing other two pages have a higher chance to go to page x in the end. The probability is calculated as 100 percent. Thus, holding more incoming links will substantially increase the page rank on search engines.

7.4.2 Metadata

Metadata is the header of a web page in HTML, containing the basic information of the page. In the data title, keywords can be declared. This information is fundamental for the rank. Search engines will get these tags in metadata when crawling and calculate the rank for these tags. The next time a user searches a word the same as or similar to the title or a keyword, search engines will present the page in the order of the page rank for the word.

7.4.3 Incoming links

The most efficient way to increase the page rank is to have quality incoming links. As described in the mathematical equations above, incoming links from websites with a high page rank will count a lot for the ranking. To increase the number of incoming links, a web application should have valuable content or something people love to share on their social media. A social share feature can be added to the application to increase number of incoming links.

7.4.4 Alt tag

Images and videos on a web application should come with alt tags. Search engines will search for the alt tag and save the page URL as a reference for animations. This will increase page view if the page has good images or video.

7.4.5 Content hierarchy

Keeping the web content in a standard structure is also beneficial to the page rank. A designer should always code the primary title in <h1></h1> tag, and sub titles in <h2> and <h3>, as the search engines will specifically search these title tags for keywords.

7.5 Localization

Localizing the application significantly expands the user scope. By default, contexts on the page are shown in English. The localization of the web content is implemented in two parts: one part is to localize the strings on the page; the other part is to localize the contents from a database.

To localize a string on web page, instead of coding it in a tag like <h2>Test</h2>, wrap it with PHP echo in the following code, and then put translated strings into corresponding .po files.

```
<h2><?php echo __("Test"); ?></h2>
```

The software Poedit can be used to generate .po files. Manually input source strings in the left column and translated strings on the right column. Click save to generate .po files. Each .po file is for one language. The contexts on a web page can load the translation according to user's language setting in browser.

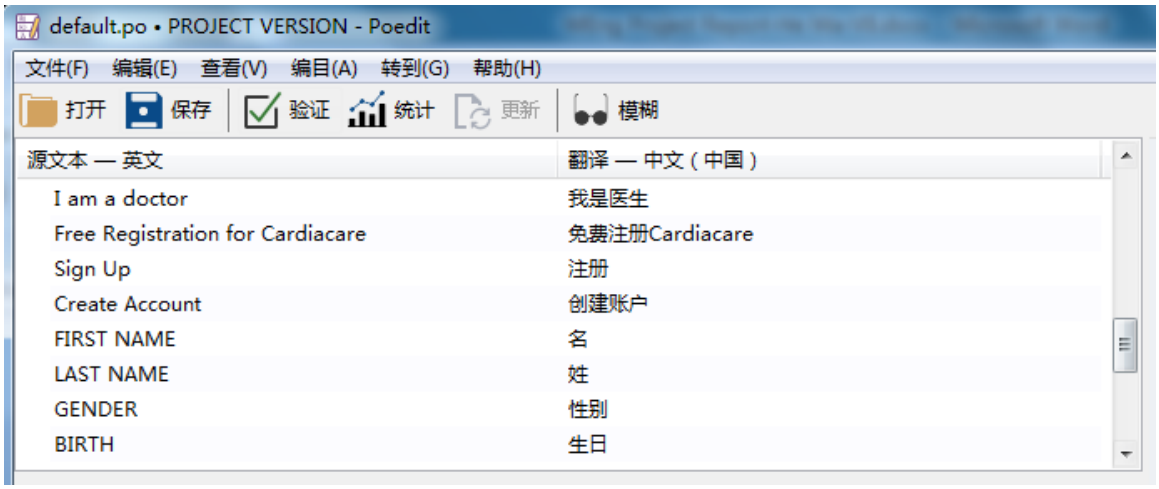


Figure 37. Translation in Poedit

To translate contents in the database, such as comments and notes, it can be done in the controller. After getting the object, call the following command to obtain localized strings, the localized strings will be retrieves from corresponding files.

```
Messages.localize(locale, object.content);
```

7.6 Google Analysis

Inserting Google analysis into application can help analysis the page view and other volumes of the web application. Google analysis provides helpful information such as user's location distribution, the source URL users come from, average staying time on the application, Google ads hit volume, etc. To insert Google analysis to the application, first need to apply the tracking id on Google website, and then add Google's script into header layout.

By implementing strategies discussed above, the DCG web application provides user a good user experience and becomes searchable in popular search engines.

Chapter 8 Testing

Testing is an important part of the process of developing a software. “Testing of a software is to ensure the quality of source code and every piece of code works as expected. In web applications, the testing objects are the functions in the controllers which need to be tested in Internet environment” [19]. There are three common types of testing, unit test, integration test, and selenium test.

8.1 Unit and Integration Tests

The unit test is a test method to test units in a software. For instance, the unit test can be used to test the getRecords method in DCGEestController. First creating a mock test records map, and then call the getRecords method using different parameters, such as a normal test_id, a test_id that does not exist in the map, a negative int, and an empty string. Then use “assert expression” to check the result. If all results are equal to the expected values, it means that the getRecords method works fine in the system.

Using a unit test is necessary and beneficial, as it helps find and locate bugs in early stages of development. It is also not influenced by changes in other methods in the software. The unit test is therefore a good illustration of the features of a method. It is easy to understand what the method actually does by reading test cases in its unit test, so it is always good to write unit test for buggy methods once they are created.

“The integration test is another type of software testing, in which functions are grouped into the module and tested as a whole” [20]. There are two ways to put the modules into groups, top down and bottom up. Top down tests top-level methods correctness and includes lower level methods when they are called inside while bottom up is the opposite

starting from base level methods that have no function calls inside and gradually going up to the top level.

The selenium test is based on selenium framework. The unit test and integration test validate only the program correctness, but they know nothing about how the rendered web pages look. Selenium can test this as it can mock users behaviour such as clicking and selecting in a browser to test if the web application can work in the browser as expected.

Testing is regarded the last door of preventing software failure. If testing fails to detect a software flaw, it will be directly exposed to the user. Thus comprehensive test cases are needed to ensure coverage of all scenarios in a software.

8.2 PHPUnit Test

In the DCG web application, PHPUnit will be used for the unit test. It is already included in full package of CakePHP framework, and if not, it can be installed via composer as follows:

```
"phpunit/phpunit": "3.7.38"
```

Then run “composer install” in the console to install the dependency.

During the testing, some test may create, update or delete the persistent data in database, so it is necessary to copy a new database for the test. The following code snippet shows how to set the test database in CakePHP:

```
$test = array(
   'datasource' => 'Database/Mysql',
   'host'       => 'cardiacare.ca',
   'login'      => 'admin',
   'password'   => 'M523895h',
   'database'   => 'cardiacare_test'
);
```

For example, to test the return correctness for the DCG test method, a test function can be written as the following code snippet:

```
function DCG_test( $id = 1 ) {
   $query = $this->HttpGetRequest($url, $data);
   $associated =  array(Test => array('id' => 1, 'title' => ' First Test'));
   $this->assertEquals($associated, $ query);
    $this->assertEquals(sizeof($ query->records), 5);
}
```

The test can be run either from the command line or in a browser. If in a browser, input the test method URL path which is at the same level as the tested method, and then the browser will run the test code and output a test coverage to show the covered part in the original method. This helps check whether or not all codes in the method are covered. The red section in the test coverage means that the test case is defective.

In general, testing does not introduce new features or functionalities to the application, but it is not a waste of time to do it, as it can help maintain the code quality, and prevent code bugs and feature regression, which are critical to the stability of a web application. In practice, quality testing is performed by an independent team that is in charge of the quality assurance of the software. Automated testing, such as the unit test and integration test, are highly recommended to keep the software quality.

Chapter 9 Conclusions

9.1 Future Work

This project implements the DCG web application and analyze several main aspects of it. There are some future works that can be done to further improve the DCG web application.

1. The DCG web application uses a single database to store data as, currently, the dataset is small. In the future, the database may need to be split into two databases, as the amount of data grows. One of the new database will be specifically for handling DCG-related data.
2. Currently, the DCG web application mainly presents DCGs to the user. When the application is released, there will be much DCG data created by users every day. The volume of data will greatly increase to gigabytes, and terabytes. Big data technology (Hadoop Distributed File System, MapReduce, Hive, etc.) can be implemented to analyze the generality and specialty of mass users data based on various measures, such as age, region, gender, social role, and marital status. The analysis result can then be presented to the user. Moreover, if by modeling doctors comments data using a machine learning algorithm, the DCG web application may be able to find a regular pattern and automatically provide some suggestions to the user.

9.2 Conclusions

Throughout the practical implementation of the DCG web application, this report covers all aspects of web application development, including functionality, architecture,

performance, security, scalability, user experience and testing. The DCG web application achieves the core function of presenting DCGs. It has an MVC architecture and implements compression and caching to enhance performance. Input field escaping ensures security, and a load balancer improves system scalability. The components are consistent throughout the application for optimal user experience, and the PHPUnit test is added to guarantee software quality. The assessment of these methods and the improvement of the application have been discussed in the report. Although theses methods may not apply to all types of web applications, such that adjustments are needed in best practice, these strategies are still very useful and meaningful to the DCG web application and many other common web applications.

Bibliography

[1] (2016) 2009 Tracking Heart Disease and Stroke in Canada. [Online]. Available: http://www.phac-aspc.gc.ca/publicat/2009/cvd-avc/pdf/cvd-avs-2009-eng.pdf

[2] (2016) Electrocardiographic abnormalities predict deaths. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/16644339

[3] (2016) Electrocardiography on Wikipedia. [Online]. Available: http:// en.wikipedia.org/wiki/Electrocardiography

[4] Ronald M. Birse,rev. Patricia E. Knowlden Oxford Dictionary of National Biography 2004

[5] (2016) Taobao Sales Center [Online]. Available: http://www.taobao.com

[6] J. d. Hu; W. Ye; W. b. Zou. Portable Ambulatory Electrocardiogram Monitoring System Based on Bluetooth, IEEE Conference Publications, 2009

[7] (2016) Background image downloaded from Pexels. [Online]. Available: https://www.pexels.com/photo/doctor-medical-medicine-stethoscope-34843/

[8] (2016) Create, read, update, delete on Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Create,_read,_update_and_delete

[9] (2016) Model-View-Controller on w3schools. [Online]. Available: http://www.w3schools.com/aspnet/mvc_intro.asp

[10] (2016) Relational database on Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Relational_database

[11] Ye Zhou ; Mobile Life & New Media Lab., Key Lab. of Universal Wireless Commun., Beijing, China ; Yang Ji. “Design of rest APIs for the exposure of IMS capabilities towards Web Services”. 2011 IEEE.

[12] (2016) Minify plugin on Github. [Online]. Available: https://github.com/maurymmarques/minify-cakephp

[13] (2016) Google Developers Rules. [Online]. Available: https://developers.google.com/speed/docs/insights/UseAsync#overview

[14] Jacques A. Pienaar, Robert Hundt. “JSWhiz: Static analysis for JavaScript memory leaks”. Code Generation and Optimization (CGO). 2013 IEEE

[15] (2016) Cross-Site Request Forgery on OWASP. [Online]. Available: https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)

[16] (2016) The High-Performance SQL Blog. [Online]. Available: https://www.periscopedata.com/blog/count-distinct-in-mysql-postgres-sql-server-and-oracle.html

[17] Web Application Servers Performance Blog. [Online]. Available: http://www.webperformance.com/load-testing/blog/2011/11/what-is-the-fastest-webserver/

[18] (2016) Page Rank on Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/PageRank

[19] Lei Xu. Dept. of Comput. Sci. & Eng., Southeast Univ., Nanjing, China. Baowen Xu. "A framework for web applications testing". 2004. IEEE

[20] (2016) Integration Testing on Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Integration_testing

Appendix A Source Code

The source code of DCG web application can be downloaded at https://github.com/MarcHe124/cardiacare, source code should work on a apache or iis server on windows, linux or mac.

Appendix B API Documentation

For complete API blueprint document please visit online at
https://github.com/MarcHe124/cardiacareAPI

FORMAT: 1A
HOST: http://ecg.ece.uvic.ca/rest/

The DCG API

The DCG website's api blueprint(Under updating).

Group Patient

/patient/login

This method lets patients log in.

<table>
  <tr>
    <td colspan="2">Required JSON Body </td>
  </tr>
  <tr>
    <td width="50%">username</td>
    <td width="50%">String</td>
  </tr>
  <tr>
    <td>password</td>
    <td>String</td>
  </tr>
</table>

POST [POST]

+ Request (application/json)

```
{
   "username": "arthurma124@gmail.com",
   "password": "sdaofijonwaenf293jr2o3jo2i3j"
}
```

+ Response 200 (application/json)

```
{
   "result":"success",
   "userid": "25",
   "usercol":"patient",
   "session_id":"1254254654"
}
```

/patient/register

This method lets patients sign up account.

<table>
 <tr>
  <td colspan="2">Required JSON Body </td>
 </tr>
 <tr>
  <td width="50%">firstname</td>
  <td width="50%">String</td>
 </tr>
 <tr>
  <td>lastname</td>
  <td>String</td>
 </tr>
 <tr>
  <td>gender</td>
  <td>male | female</td>
 </tr>
 <tr>
  <td>birthday</td>
  <td>Date</td>
 </tr>
 <tr>
  <td>username</td>
  <td>String</td>
 </tr>
 <tr>
  <td>password</td>
  <td>String</td>
 </tr>
 <tr>
  <td>address</td>
  <td>String</td>
 </tr>
 <tr>

```
   <td colspan="2"><b>Optional JSON Body </b></td>
  </tr>
  <tr>
   <td>medical_plan_number</td>
   <td>String</td>
  </tr>
  <tr>
   <td>phone</td>
   <td>String</td>
  </tr>

  <tr>
   <td>subscribe</td>
   <td>true | false</td>
  </tr>
</table>


### POST [POST]

+ Request (application/json)

      {

         "firstname": "he",
         "lastname": "ma",
         "gender": "female",
         "birthday": "1985-07-15",
         "medical_plan_number": "512526346346",
         "phone": "604-445-6645",
         "address": "8717 cook crescent, Richmond",
         "username": "arthurma124@gmail.com",
         "password": "sdaofijonwaenf293jr2o3jo2i3j",
         "subscribe": false
      }


+ Response 200 (application/json)

      {
         "result":"success",
         "userid": "25",
         "usercol":"patient",
         "session_id":"1254254654"
      }
```

/patient/logout

This method lets patients log out.

<table>
  <tr>
    <td colspan="2">Required JSON Body </td>
  </tr>
  <tr>
    <td>session_id</td>
  </tr>
</table>

POST [POST]

+ Request (application/json)

```
{
   "session_id": "12512632623626"
}
```

+ Response 200 (application/json)

```
{
    "result":"success"
}
```

/patient/confirmrequest

This method lets patients accept or deny doctor adding request.

<table>
  <tr>
    <td colspan="2">Required JSON Body </td>
  </tr>
  <tr>
    <td colspan="2">session_id</td>
  </tr>
  <tr>
    <td colspan="2">doctor_id</td>
  </tr>
  <tr>
    <td width="50%">decision</td>

```
   <td width="50%">true | false</td>
  </tr>
</table>
```

POST [POST]

+ Request (application/json)

```
{
   "session_id": "12512632623626",
   "doctor_id": "34636347347",
   "decision": true
}
```

+ Response 200 (application/json)

```
{
   "success":true
}
```

/patient/viewed

This method lets patients update view status.

```
<table>
  <tr>
   <td colspan="2"><b>Required JSON Body </b></td>
  </tr>
  <tr>
   <td colspan="2">session_id</td>
  </tr>
  <tr>
   <td colspan="2">data_id</td>
  </tr>
  <tr>
   <td width="50%">data_type</td>
   <td width="50%">doctor | test | comment</td>
  </tr>
</table>
```

POST [POST]

+ Request (application/json)

```
{
   "session_id": "12512632623626",
   "data_id": "325235235",
   "data_type": "test"
}
```

+ Response 200 (application/json)

```
{
   "success":true
}
```

/patient/{id}

Get the basic information for a patient account. You will need to have a valid session id.

<table>
  <tr>
    <td>Required Parameters </td>
  </tr>
  <tr>
    <td>session_id</td>
  </tr>
</table>

GET [GET]

+ Response 200 (application/json)

```
{
   "result":"success",
   "username": "arthurma124@gmail.com",
   "firstname": "he",
   "lastname": "ma",
   "userid":"133",
   "total_tests":33,
   "total_doctors":13,
   "total_comments":112,
   "total_notes": 242,
   "birthday": "1985-07-15",
   "medical_plan_number": "512526346346",
   "phone": "604-445-6645",
```

"address": "8717 cook crescent, Richmond"
}

/patient/{id}/gettests

Get the tests of a patient. You will need to have a valid session id.

<table>
  <tr>
    <td colspan="2">Required Parameters </td>
  </tr>
  <tr>
    <td colspan="2">session_id</td>
  </tr>
  <tr>
    <td colspan="2">Optional Parameters </td>
  </tr>
  <tr>
    <td width="30%">limit</td>
    <td width="70%">Minimum 1, maximum 100.</td>
  </tr>
  <tr>
    <td width="30%">page</td>
    <td width="70%">Minimum 1, maximum 1000.</td>
  </tr>
</table>

Appendix C Database SQL

The database sql(Structured Query Language) was exported from mysql database, which contains the tables structure and some seed data of the whole cardiacare database system, it can almost be used in any relational database system that can resolve sql command and import cardiacare database by using this file.

```
-- phpMyAdmin SQL Dump
-- version 4.1.14
-- http://www.phpmyadmin.net
--
-- Host: 127.0.0.1
-- Generation Time: Feb 06, 2016 at 12:43 AM
-- Server version: 5.6.17
-- PHP Version: 5.5.12

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET time_zone = "+00:00";


/*!40101 SET
@OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET
@OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET
@OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;

--
-- Database: `cardiacare`
--

-- --------------------------------------------------------

--
-- Table structure for table `access_request`
--

CREATE TABLE IF NOT EXISTS `access_request` (
  `doctor_id` int(5) NOT NULL DEFAULT '0',
  `userid` int(5) NOT NULL DEFAULT '0',
  PRIMARY KEY (`doctor_id`,`userid`)
```

```
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--
-- Table structure for table `doctor_info`
--

CREATE TABLE IF NOT EXISTS `doctor_info` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `user_name` varchar(15) NOT NULL DEFAULT '',
  `user_pass` varchar(20) NOT NULL DEFAULT '',
  `email` varchar(40) NOT NULL DEFAULT '',
  `firstname` varchar(20) NOT NULL DEFAULT '',
  `lastname` varchar(20) NOT NULL DEFAULT '',
  `phone` varchar(20) NOT NULL DEFAULT '',
  `mobile` varchar(20) NOT NULL DEFAULT '',
  `lasttime` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
  `hospital` varchar(50) NOT NULL DEFAULT '',
  `patient_num` int(10) NOT NULL DEFAULT '0',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=7 ;

--
-- Dumping data for table `doctor_info`
--

INSERT INTO `doctor_info` (`id`, `user_name`, `user_pass`, `email`, `firstname`,
`lastname`, `phone`, `mobile`, `lasttime`, `hospital`, `patient_num`) VALUES
(1, 'vision', '123456', 'liweizheng1992@gmail.com', 'vision', 'li', '123456', '123456', '0000-
00-00 00:00:00', 'General Hospital', 0),
(2, 'Tom', '123456', 'liweizheng1992@gmail.com', 'vision', 'li', '123456', '123456', '0000-
00-00 00:00:00', 'General Hospital', 0),
(3, 'Ray Allen', '123456', 'liweizheng1992@gmail.com', 'vision', 'li', '123456', '123456',
'0000-00-00 00:00:00', 'General Hospital', 0),
(4, 'Allen Iverson', '123456', 'liweizheng1992@gmail.com', 'vision', 'li', '123456', '123456',
'0000-00-00 00:00:00', 'General Hospital', 0),
(5, 'allen', '123456', '123@gmail.com', 'vision', 'li', '123', '123', '0000-00-00 00:00:00',
'123', 0),
(6, 'alle2n', '123456', '123@gmail.com', 'vision', 'li', '123', '123', '0000-00-00 00:00:00',
'123', 0);

-- --------------------------------------------------------

--
-- Table structure for table `doc_comment`
```

```
--

CREATE TABLE IF NOT EXISTS `doc_comment` (
  `doctor_id` int(11) NOT NULL DEFAULT '0',
  `userid` int(11) NOT NULL DEFAULT '0',
  `comment_time` datetime NOT NULL,
  `comment` mediumtext NOT NULL,
  `noteid` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `starttime` datetime NOT NULL,
  PRIMARY KEY (`noteid`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=4 ;

--
-- Dumping data for table `doc_comment`
--

INSERT INTO `doc_comment` (`doctor_id`, `userid`, `comment_time`, `comment`,
`noteid`, `starttime`) VALUES
(1, 1, '2015-05-05 10:12:22', 'my first comment from weizheng li hahahahaha', 1, '2015-
05-01 10:31:38'),
(0, 0, '0000-00-00 00:00:00', '', 2, '2015-05-01 10:31:38'),
(0, 0, '2015-05-10 19:58:43', '%s', 3, '0000-00-00 00:00:00');

-- --------------------------------------------------------

--
-- Table structure for table `ecgdata`
--

CREATE TABLE IF NOT EXISTS `ecgdata` (
  `dataid` int(10) unsigned NOT NULL,
  `datacontent` mediumblob NOT NULL,
  KEY `dataid` (`dataid`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--
-- Table structure for table `ecgdatainfo`
--

CREATE TABLE IF NOT EXISTS `ecgdatainfo` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `starttime` datetime NOT NULL,
  `length` int(10) unsigned NOT NULL,
  `ifdone` tinyint(1) NOT NULL,
```

```
  `userid` int(11) NOT NULL,
  `ifcs` tinyint(1) NOT NULL DEFAULT '0',
  `HR` varchar(10) NOT NULL,
  `QT_interval` varchar(10) NOT NULL,
  `QTc` varchar(10) NOT NULL,
  `QRS_duration` varchar(10) NOT NULL,
  `PR_interval` varchar(10) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=2105 ;

--
-- Dumping data for table `ecgdatainfo`
--

INSERT INTO `ecgdatainfo` (`id`, `starttime`, `length`, `ifdone`, `userid`, `ifcs`, `HR`,
`QT_interval`, `QTc`, `QRS_duration`, `PR_interval`) VALUES
(1985, '2015-03-12 14:08:29', 14, 1, 1, 0, 'No Value', 'No Value', 'No Value', 'No Value',
'No Value'),
-- --------------------------------------------------------

--
-- Table structure for table `patient_info`
--

CREATE TABLE IF NOT EXISTS `patient_info` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `user_name` varchar(15) NOT NULL DEFAULT '',
  `user_pass` varchar(20) CHARACTER SET utf8 NOT NULL DEFAULT '',
  `email` varchar(40) NOT NULL DEFAULT '',
  `firstname` varchar(20) NOT NULL DEFAULT '',
  `lastname` varchar(20) NOT NULL DEFAULT '',
  `phone` varchar(20) NOT NULL DEFAULT '',
  `emergencynum` varchar(20) NOT NULL DEFAULT '',
  `emergencymes` varchar(100) NOT NULL DEFAULT '',
  `ifsendsms` tinyint(1) NOT NULL DEFAULT '0',
  `facebookid` varchar(20) NOT NULL DEFAULT '',
  `googleplusid` varchar(30) NOT NULL DEFAULT '',
  `ifappendloc` tinyint(1) NOT NULL DEFAULT '0',
  `savelength` int(10) unsigned NOT NULL DEFAULT '60000',
  `ifcsmode` tinyint(1) DEFAULT '0',
  `ifturnoffbt` tinyint(1) DEFAULT '1',
  `lowbpm` int(10) unsigned NOT NULL DEFAULT '40',
  `lastdevice` varchar(50) NOT NULL DEFAULT '',
  `lastip` varchar(20) NOT NULL DEFAULT '',
  `lasttime` datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
  `version` varchar(10) NOT NULL DEFAULT '',
```

```
  `imei` varchar(20) NOT NULL DEFAULT '',
  `access_request` int(5) NOT NULL DEFAULT '0',
  `doctor_num` int(10) NOT NULL DEFAULT '0',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=26 ;

--
-- Dumping data for table `patient_info`
--

INSERT INTO `patient_info` (`id`, `user_name`, `user_pass`, `email`, `firstname`,
`lastname`, `phone`, `emergencynum`, `emergencymes`, `ifsendsms`, `facebookid`,
`googleplusid`, `ifappendloc`, `savelength`, `ifcsmode`, `ifturnoffbt`, `lowbpm`,
`lastdevice`, `lastip`, `lasttime`, `version`, `imei`, `access_request`, `doctor_num`)
VALUES
(1, 'liwanbo', '123456', '', '', '', '', '', 0, '', '', 0, 60000, 0, 1, 40, '', '', '0000-00-00 00:00:00',
'', '', 0, 0),
(2, 'wanboli', '123456', '', '', '', '', '', 0, '', '', 0, 60000, 0, 1, 40, 'LGE Nexus 4',
'134.87.137.93', '2014-12-04 12:53:13', '1.7', '356489054542001', 0, 0),
-- --------------------------------------------------------

--
-- Table structure for table `relation`
--

CREATE TABLE IF NOT EXISTS `relation` (
  `doctor_id` int(5) NOT NULL DEFAULT '0',
  `userid` int(5) NOT NULL DEFAULT '0',
  PRIMARY KEY (`doctor_id`,`userid`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `relation`
--

INSERT INTO `relation` (`doctor_id`, `userid`) VALUES
(1, 1),
(2, 1),
(3, 1),
(4, 1);

-- --------------------------------------------------------

--
-- Table structure for table `userfeedback`
--
```

```
CREATE TABLE IF NOT EXISTS `userfeedback` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `userid` int(11) NOT NULL,
  `feedback` varchar(150) NOT NULL DEFAULT '',
  `sendtime` datetime NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=11 ;

--
-- Dumping data for table `userfeedback`
--

INSERT INTO `userfeedback` (`id`, `userid`, `feedback`, `sendtime`) VALUES
(1, 12, 'dhjjbvbvhhj', '2014-07-11 13:08:36'),
(2, 12, '4354866655', '2014-07-11 13:09:49'),
(3, 1, 'no', '2015-01-11 14:42:23'),
(4, 1, 'ECG software', '2015-01-11 14:43:28'),
(5, 1, 'u', '2015-02-09 16:12:31'),
(6, 1, 'uo', '2015-02-09 16:12:39'),
(7, 1, 'uo', '2015-02-09 16:14:53'),
(8, 1, 'uo', '2015-02-09 16:14:54'),
(9, 1, 'uo', '2015-02-09 16:14:56'),
(10, 1, 'uo', '2015-02-09 16:15:00');

--
-- Constraints for dumped tables
--

--
-- Constraints for table `ecgdata`
--
ALTER TABLE `ecgdata`
  ADD CONSTRAINT `ecgdata_ibfk_1` FOREIGN KEY (`dataid`) REFERENCES
`ecgdatainfo` (`id`) ON DELETE CASCADE;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET
CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION
*/;
```

Appendix D Flow Chart

The flow chart shows all workflows based on navigation of web pages and layouts.

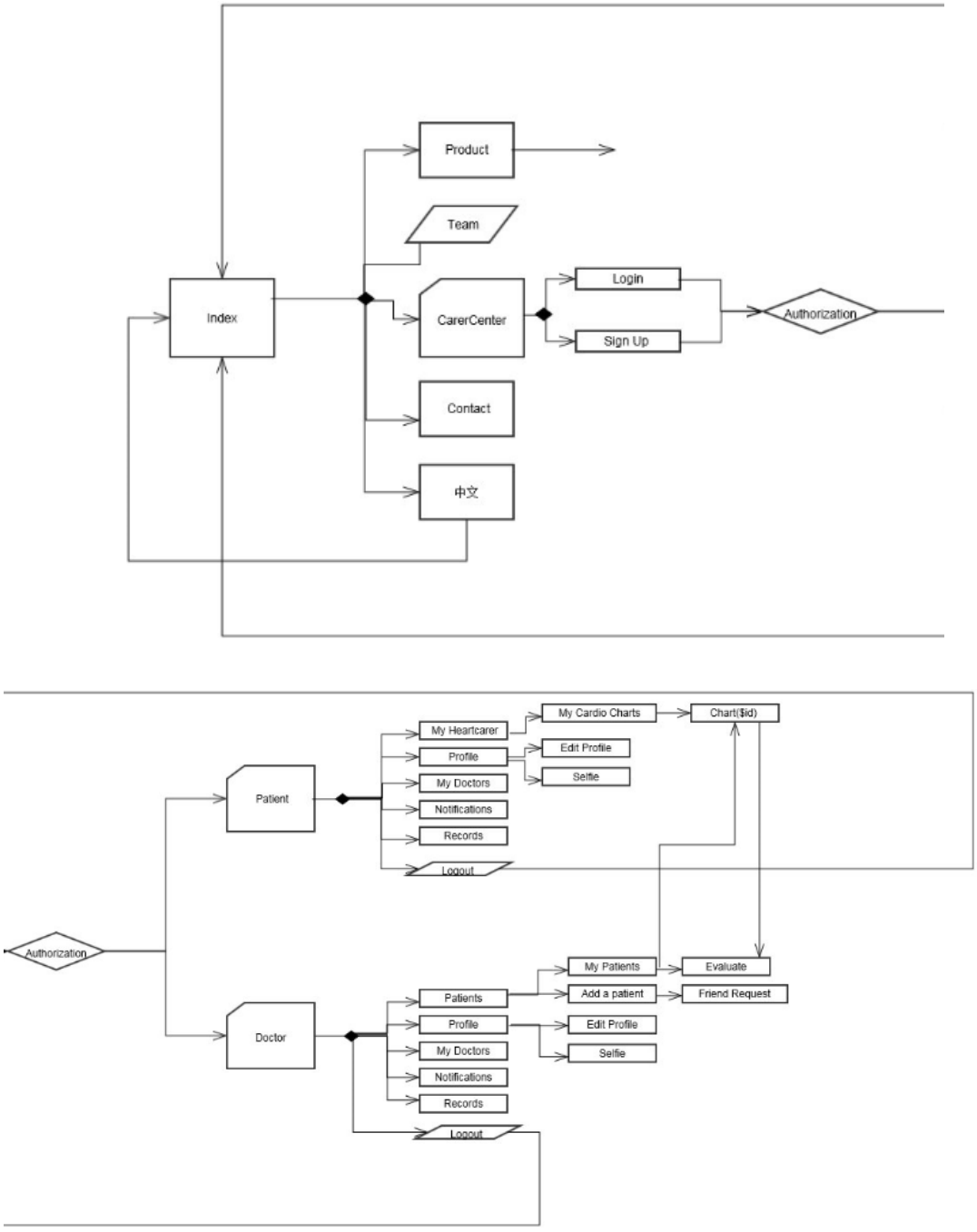


Figure 38. Flow Chart of DCG Web Application Workflows

Appendix E Database Schema

The database schema is drawn in MySQL workbench which describes database's structure. It provides an overview of relationships among data tables and help build seed database using database management system.

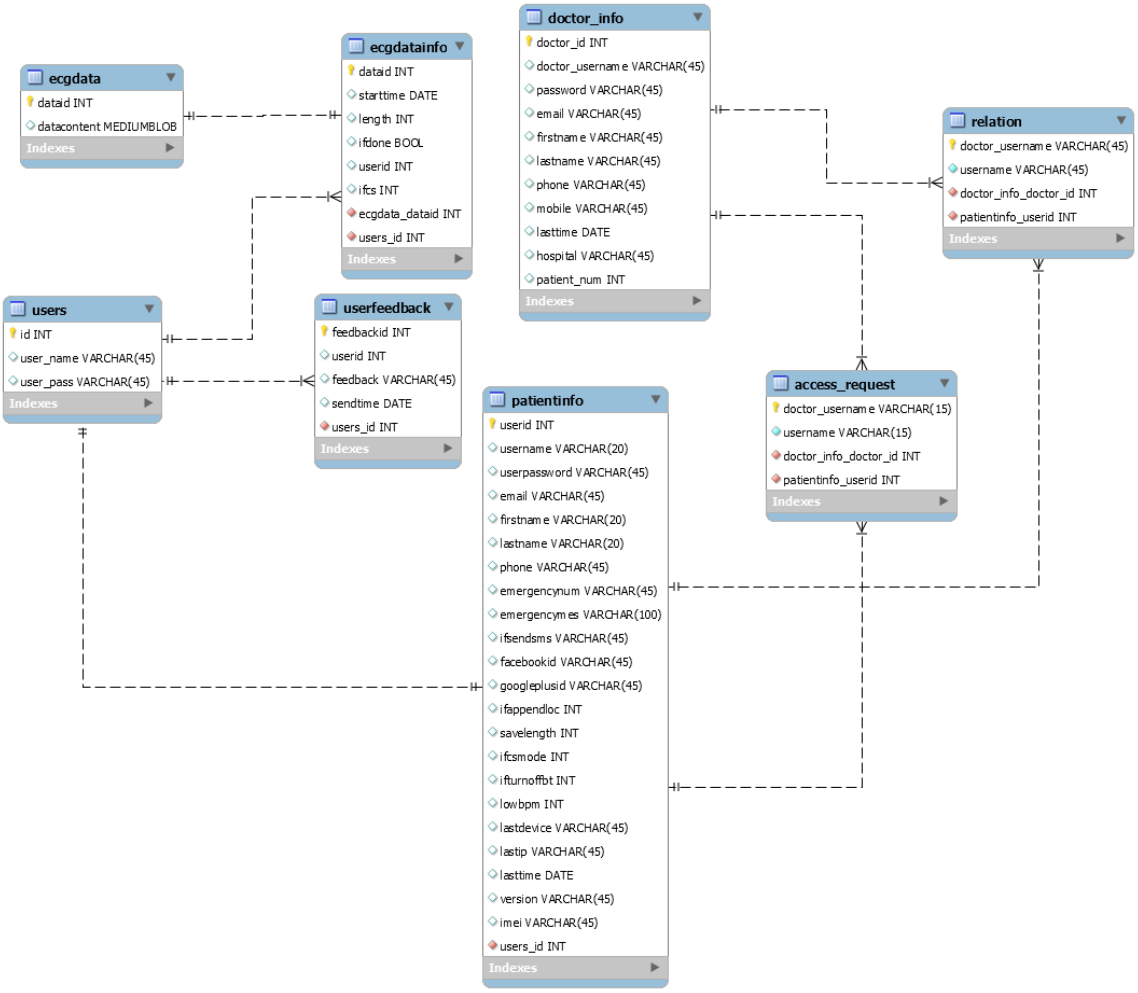


Figure 39. Database Schema in Mysql