**CS673F13 Software Engineering**

**Group Project - Type II Diabetes Management CDSS**

**Project Proposal and Planning**



|  |  |  |  |
| --- | --- | --- | --- |
| Team Member | Role(s) | Signature | Date |
| Bogdan Chayka | Backup project leader, Design leader | *Bogdan Chayka* | 11/07/2013 |
| Jason Lu | Configuration leader | *Jason Lu* | 11/07/2013 |
| Jeff Andre | Project leader, Requirements | *Jeff Andre* | 11/07/2013 |
| Maura Huff | Implementation leader | *Maura Huff* | 11/07/2013 |
| Wenjie Shi(Jenny) | Environment and integration leader | *Wenjie Shi* | 11/07/2013 |
| Yike Xue (Eva) | QA leader | *Yike Xue* | 11/07/2013 |
| Yingyuan (Allen) Zhang | Requirements leader | *Yingyuan Zhang* | 11/07/2013 |
| Yulu Liu | Design leader | *Yulu Liu* | 11/07/2013 |

**Revision history**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Author** | **Date** | **Change** |
| **1.0** | **All Team members** | **09/25/2013** | **xxxx** |
| **2.0** | **Jeff Andre, Yike Xue** | **10/8/2013** |  |
| **3.0** | **Jeff Andre, Yike Xue, Bogdan Chayka** | **11/07/2013** |  |

[Overview](#h.87t9hln2vjz0)

[Related Work](#h.mps353x5ezyl)

[Detailed Description](#h.fg3z0hpd4q9v)

[Management Plan](#h.ds8oyr75pnh1)

[Process Model](#h.27177f40uci)

[Risk Management](#h.a4oqwntk3mw)

[Monitoring and Controlling Mechanism](#h.ywdoc2clc9yt)

[Schedule and deadline](#h.tadq5mb0pici)

[Quality Assurance Plan](#h.72e1f4uawy2r)

[Metrics](#h.b2haznn3yyz2)

[Standard](#h.vc72k6dweldv)

[Inspection/Review Process](#h.f1c69ifi68h7)

[Testing](#h.r5d5mhtlf0kq)

[Defect Management](#h.54a4wuncjg1c)

[Process improvement process](#h.jhct37ebxxpn)

[Configuration Management Plan](#h.hw41vg4ykxen)

[Configuration items and tools](#h.bwlb4d4vdox2)

[code commit guidelines](#h.yyauft6zr9hw)

[References](#h.8mva2050iy7t)

[Glossary](#h.ty3i2nqffhtc)

# Overview

This project will be developed for the MET Health Informatics Lab (MET-HILAB) under direction of Professors Guanglan Zhang and Vladimir Brusic (the customers). The main goal of the project is to develop a type 2 diabetes management web based application. The purpose of this application is intended to be used as a teaching tool that will demonstrate implementation of a real clinical algorithm, examples of successful treatment decisions, situations where clinical algorithms may fail, and to provide a platform for familiarization of students with the process of development and implementation and use of medical algorithms.

Users of this application will be students and teachers in the health informatics course.

# Related Work

For our best knowledge similar software systems do not exist, although UI of the Project will simulate UI of Electronic Health Records (EHR) system. Our application will have limited functionality, comparing with existing EHR, only enough to support demonstration of working Clinical Decision Support System (CDSS) based on real world Medical Algorithm.

# Detailed Description

This project is divided into five major parts a) user interface, b) a database, c) a type II diabetes management algorithm, d) health calculators, and e) security. Figure 1 shows a diagram of the application.

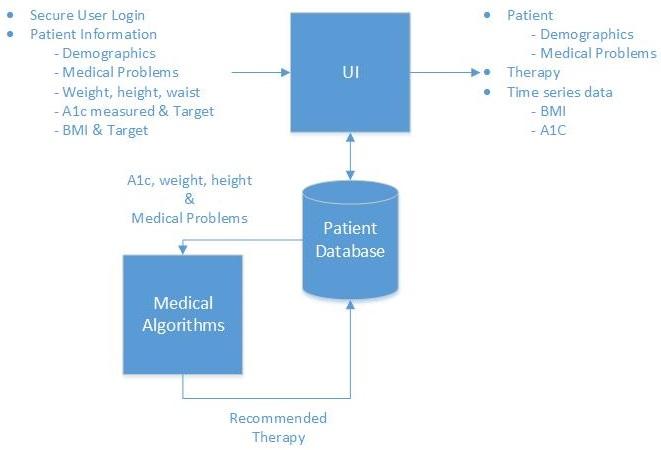


Figure 1. System Diagram for Type II Diabetes Management WEB Application

**3.1 User Interface**

The UI is used for patient information management and will simulate EHR with a form to input patients’ demographic and medical information. It will also display patient data and algorithm results for suggested therapy.

The user should be able to:

- register as a new user

- login in with username and password

- logged in user can register new patient

- input patient demographic and medical data

- update patient data

- view the list of existing patients and patient data he/she created

- delete patient from the database

- search patients with PID, First or Last name

- manage users with administrative privileges

**3.1.1 User Login**

A login page should allow a registered user to login and/or register if not already registered. The application will allow registration only unique users.

The user must login to access patient information (register, retrieve or update patient info).

**3.1.2 Register New Patient**

The user should be able to register a new patient and input demographics and medical data.

Demographics:

- Patient ID (will be generated by application)

- First name

- Last name

- DOB - date-of-birth

- Gender

Medical Data:

Medical data will include an associated date for each input.

- Problems (allergies, medical contraindications, diseases, complications)

- Visits (Vitals and lab results: A1C, height, weight, systolic and diastolic blood pressure)

- Services (medications, therapy)

**3.1.3 View Patients list**

The user should be able to see list of patients that he/she created. Each record will include following information: patient number, first and last name, date of birth, username who created the record.

**3.1.4 Update Patient data**

The user should be able to update existing patient demographic and medical data. Only user who created patient record allowed update it or users with administrative privileges.

**3.1.5 View Patient data**

The user should be able to retrieve and view existing patient data. Only user who created patient record allowed to view it or users with administrative privileges.

Would be good if UI will have functionality graphically present the time series patient’s data, for example plot a patient’s blood glucose levels across his/her visit history.

**3.1.6 Delete Patient from the database**

The user should be able to delete a patient’s record. Only user who created patient record allowed delete it or users with administrative privileges.

**3.1.7 Search patients**

User should be able to search patients by PID, First Name or Last Name.The user can search only records that he/she created, users with administrative privileges can search the whole database.

**3.1.8 Manage Users**

Users with administrative privileges should be able to add, update and delete users.

**3.2 Diabetes Management Algorithm**

The user should be able to run the algorithm on any patient in the database so that they can view the recommended therapy based on the algorithm results.

The algorithm will interact with the database to retrieve medical data of the selected patient, will provide a recommended therapy for diabetes management, user will be able to edit suggested therapy/recommendations and store it in database. The algorithm will be based on the AACE Comprehensive Diabetes Management Algorithm reported in (Garber, 2013).

The Glycemic Control Algorithm will be implemented for this project.

**3.3 Database**

Relational database will be used to store patient data and algorithm results. The database will be accessed by registered user to store, update and delete patient information.

There will be a separate table in database to store data about medications.

To meet the new requirement in iteration 2, a new table “user\_profile” is added to the database in order to manage the users conveniently and separate the privileged administrator-user from others.

**3.4 Health Calculators**

Would be good to provide the following medical calculators.

- Body Mass Index

- A1C-to-Average Blood Glucose Level

**3. 5 Security**

Due to the nature of this application with patient information in a database, security issues need to be addressed such as:

- Secure login to access patient information.

- Validation of all user input to prevent malicious attacks such as SQL injection, and to prevent errors and mistakes from normal users.

- Would be good to implement SSL certification on a final stage.

# Management Plan The project management plan includes the sections below.

## Process Model

All iterations of this project will be executed using the agile process as close as possible.

The first iteration will provide a baseline to solicit feedback from customer on features and requirements. The baseline will have a secure login and patient information management including add, update, view, and delete. It will also include a database to store patient information.

The second iteration will include additional features based on customer feedback from the first iteration plus a working diabetes management algorithm.

The third iterations will include final features from customer feedback from the second iteration, code refactoring, and will be the final tested and delivered version.

## Objectives and Priorities

Overall objectives and priorities are listed below. Priorities on detailed requirements/features will be assigned in pivotal tracker.

Feature will be priority in the first iteration, and quality will improve after the second iterations, and quality will be priority for the third iteration.

* First iteration: deliver working Web Application with UI,database, secure login and patient management functionality.
* Second iteration: Implementation of Medical algorithm, suggested changes and/or additional features from customer feedback
* Third iteration: suggested changes and/or additional features from customer feedback, code refactoring, deliver working web application according to requirements

|  |  |  |
| --- | --- | --- |
| # | **Feature** | **Level of Priority** |
| 1. | Secure Log in | High |
| 2. | User registration | High |
| 3. | Add new patient | High |
| 4. | Update/Delete patient | High |
| 5. | Create new visit for patient | High |
| 6. | Update/delete visits | High |
| 7. | List of all patients with limited access by user | High |
| 8. | Access to patient's record only by users who created the record or users with administrative privileges. | High |
| 9. | User management by users with administrative privileges. | High |
| 10. | Implementing Diabetes Management Algorithm that can retrieve individual patient data from the database and represent results. | High |
| 11. | Storing results from #10 in database with an opportunity to edit suggested therapy/recommendations | High |
| 12. | Implementing Medical Calculators: a) Body Mass Index b)A1C-to-Average Blood Glucose Level | Medium |
| 13. | Input validation for all forms | High |
| 14. | Graphical representation of BMI and lab results (A1C level) | Medium |
| 15. | Ability to show step by step algorithm’s decisions (for teaching purposes) | Medium |
| 16. | User password reset | Low |
| 17. | Check user password complexity | Medium |
| 18. | Preventing SQL injections | High |
| 19. | Generating patient ID | Medium |
| 18. | Seach patients by patient ID, First or Last Name | High |
| 19. | Calendar for entering DOB | Medium |
| 20. | Tabs on patient view page and visit page | Medium |

.

## Risk Management

The table below shows associated risks for the project. Each project meeting will have risk as an agenda item.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | Risk | Impact  1- 10  (10 is largest) | Mitigation Plan | Resp. member | due date/status |
| 1 | Medical Algorithm analysis and implementation | 10 | Finish flow diagram | Jeff | flow diagram done.  Implemented and tested first iteration. |
| 2 | Database design | 9 | Finalize ER model. Define entities/tables and make flexible to add new entities. | Bogdan | done |
| 3 | UI requirements | 8 | Get feedback from customer on first iteration. | Jeff and team | 1st iteration done. |
| 4 | Adding new Features | 5 | Finalize most features by end of 1st iteration. | Jeff | Features reviewed and approved by customer. |
| 5 | Absent due to sick time, holidays, etc. | 4 | Assign backups.  Buffer time to scheduled tasks.  Team meeting. | Jeff | team members can fill in for each other. |
| 6 | Graphics for display of time series data | 5 | Research solutions and perform isolated tests. | Maura | Identified a solution. |
| 7 | Security | 5 | Identify security threats and create a list of necessary implementations to prevent them. | Jason | Implement secure login and user input validation on client and server side. Use built in cakePHP security features. |
| 8 | Estimation of project | 7 | Prioritize requirements and evaluate team members’ skills | Jeff & team | Done for 1st iteration. |
| 9 | Necessary Skills | 7 | Asses team skills for assignments. | Jeff | cakePHP has learning curve. Skills are good at this point. |

Table 1. Risk analysis for project.

## Monitoring and Controlling Mechanism

Each team leader will provide status to the project leader in an email by Wednesday before the team meeting. The teams are divided by MVC where Jason is the controller team leader and Jenny is the UI/view team leader, and the project leader will work with the model/database team.

Status should include what was achieved during the week and any issues preventing progress.

## Schedule and deadlines

|  |  |  |  |
| --- | --- | --- | --- |
| **Planning** | Responsible | Unit Due | Final due |
| SCMP | Jason | 23-Sep | 26-Sep |
| SPMP | Jeff, Bogdan | 23-Sep | 26-Sep |
| SQAP | Wenjie, Eva | 23-Sep | 26-Sep |
| Proposal | Jeff | 23-Sep | 26-Sep |
| Initial Requirements | Jeff | 23-Sep | 26-Sep |
| Presentation | Wenjie, Eva, Bogdan, Jason | 25-Sep | 26-Sep |
|  |  |  |  |
| **Iteration 1** |  |  | 17-Oct |
| UI Wireframe | Bogdan | 30-Sep | 17-Oct |
| UI v1 |  | 10-Oct | 17-Oct |
| Secure Login |  | 3-Oct | 17-Oct |
| Database v1 | Bogdan, Jeff, Eva | 10-Oct | 17-Oct |
| Algorithm Analysis | Bogdan, Jeff | 10-Oct | 17-Oct |
| SDD v1 | Yulu | 10-Oct | 17-Oct |
| SRS v1 | Jeff, Allen | 10-Oct | 17-Oct |
|  |  |  |  |
| **Iteration 2** |  |  | 7-Nov |
| UI v2 |  | 31-Oct | 7-Nov |
| Database v2 | Eva, Bogdan | 31-Oct | 7-Nov |
| Algorithm Implementation v1 | Bogdan, Jeff, Jason | 31-Oct | 7-Nov |
| SDD v1 | Yulu | 31-Oct | 7-Nov |
| SRS v1 | Jeff, Allan | 31-Oct | 7-Nov |
|  |  |  |  |
| **Iteration 3** |  |  | 6-Dec |
| UI v3 |  | 21-Nov | 6-Dec |
| Database v3 |  | 21-Nov | 6-Dec |
| Algorithm v3 |  | 21-Nov | 6-Dec |
| All Docs v3 |  | 21-Nov | 6-Dec |

# Quality Assurance Plan

## Metrics

A software quality metric is "a function whose inputs are software data and whose output is a single numerical value interpreted as the degree to which software possesses a given attribute that affects its quality."

|  |  |
| --- | --- |
| **Metric** | **total** |
| **Code Lines** | 3733 |
| **Defect density** | 12/3.733=3.214 |
| **Achieved requirement proportion** | 8/10 |
| **Programmer productivity** | 3733/(16+13+15+12+10.5+28+22+9+27) |
| **Test cases passed rate** | 21/31 |

* Product Metrics
  + Size
    - Code Lines.
    - Using some code-counting tools(***SourceCounter***) which can get the total line numbers of the project. Keep the records in a diagram and the team can see the change of the group’s workload.
    - At the end of iteration 2, the total Lines of Code: 3733
      * Model: 646
      * View: 1594
      * Controller: 696
      * Database: 181
      * Algorithm:306
      * Misc: 181
  + Performance
    - Number of software problems reported by customer.
    - Number of software problems reported by team.
    - They are all tracked by GitHub.
  + Reliability
    - Test cases pass-rate ( = failed test number/ total test number)
    - Every failure happened in tests will be recorded as well as the total testing time. Programmers’ goal is lower this metric in the next iteration.
* Process Metrics
  + Defect Density
    - Defect density = numbers of defects found in per KLOC.
    - Defect density = numbers of defects found in per page of documentation.
    - It can be tracked divided into modules. Each programmer can take charge of some of the modules, and record their performance using Module defect density (=bugs / module size). Finally, the mean of all the module defect density is defect density.
  + Defect Fixing Time
    - In the implementation process, programmers will spend some time on fixing defects. They can summarize this time every week and compare it with the total working time and figure out whether they should change the programming approach.
  + Requirement Stability
    - Achieved requirement proportion=number of initial requirements /total number of requirements.
    - Before every iteration process, we will have some requirements of customers. We can just simply use the user story number. After the iteration, we can know how many requirements/user stories we have reached. Hopefully, this proportion should be higher and higher.
  + Efficiency
    - Programmer productivity =code size/ programming time.
    - In our weekly report, every group member will record the working time of a week. For those have new codes, they can calculate their own programmer productivity and compare with others.

## Standard

* + - documentation standard
      * Basic documentation. To make sure software implementation step by step, some basic documents are needed.

SPMP ( Software Project Management Plan)

SQAP ( Software Quality Assurance Plan)

SCMP ( Software Configuration Management Plan)

Requirements Specification ( Project Requirements)

Database Schema

Project proposal

Test plan

Test Report

* Supplementary documentation. To track the work of every member periodically.

Weekly-report

Meeting-minutes

All documentation should satisfy the requirement of correctness, completeness, consistency, non-ambiguity and timely. The responsible persons will be discussed in the weekly-meeting.

* + - * Presentation PPT

There are three presentations through the whole project. The PPT should be unified, concise and correct. The responsible persons will be discussed in the weekly-meeting.

* + - Coding standard
      * Comment should be follow [PHPDoc](http://www.google.com/url?q=http%3A%2F%2Fwww.phpdoc.org%2F&sa=D&sntz=1&usg=AFQjCNEldwVFfM4mfGgONQo8W00WDGVonQ) standard.
      * PHP / JavaScript Variables naming by [camelCase](http://en.wikipedia.org/wiki/CamelCase)
        + Don’t use [Hungarian notation](http://en.wikipedia.org/wiki/Hungarian_notation)
        + Not necessary for weak type languages.
      * HTML attributes all in lowercase.
      * Use ‘-’ to combine words in HTML classes and ids.
        + <a class=”btn-primary” id=”submit-button”>Push me</a>
      * Opening brackets follow right after statement, not a new line.
        + for($I = 0; $i < 10; $i++) {
        + /\* something here \*/
        + }
      * Do remember to add semicoma in the end of each line in JavaScript.
      * \_\_Indent by two spaces.

## Inspection/Review Process

* Inspection
  + The team leader, Jeff will be the project manager who inspect the overall software engineering process.
  + The leaders of each part of the project should be responsible for inspecting the specific part and lead the related documents.
  + During the implementation process, the inspection of every module should be taken charged of by at least one assigned group member.
  + All the inspection result should be shown to the whole group members, via Github or Pivotal Tracker. The important discoveries would be discussed in regular meetings.
* Review
  + Before handing in an idea or coding unit, the designer or programmer should review what he or she has done and avoid the mistakes.
  + Before any integration with others working result, the involvers should review all the related components in the project and try to make sure they are compatible.
  + At the beginning of each phase, QA should review the SQAP and update it according to the project progress and corrective actions.
  + Using GitHub for review tracking

## Testing

QA will monitor testing efforts to assure that test schedules are adhered to and maintained to reflect an accurate progression of the testing activities. They will assure that tests are conducted using approved test procedures and appropriate test tools, and that test anomalies are identified, documented, addressed, and tracked to closure. In addition, QA will assure that assumptions, constraints, and test results are accurately recorded to substantiate the requirements verification/validation status.

Following are the requirements of testing processes.

* + - Unit Testing
      * Testers will follow the test plan to test the unit for its correctness and stability.
      * The programmer and at least one of the other team members would take part in the unit test. The programmer should list the core function of the unit and do the White Box Testing. The other tester should do some Black Box Testing and make sure each unit works well. All the testing result would be posted on the “Pivotal Tracker” so that anybody can comment or make suggestion on them.
    - Module Testing
      * After integration of some units, at least four people should do the module testing, including two programmers who are responsible for the most units in this module and at least two other testers. The testing result would be posted on the “Pivotal Tracker” for the whole group to keep track of them.
    - System Testing
      * At the end of every iteration, the whole group members should take part in the system testing. Some of us should focus on the UE(User Experience), and some of us should focus on the algorithm running and some of us should focus on the database interface and so on. The testing results will not only be discussed on the “Pivotal Tracker”, but also in regular meetings.

Test Cases Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Page/view** | **Test description** | **Tested by** | **Bugs** | **Test passed** |
| 1 | Add Patient | Input all patient information, press submit and verify all information is correct on show patient page. | Jeff Andre | see bugs in Git | no |
| 2 | Show Patient | Check demographic & contact information | Jeff Andre | see bugs in Git | no |
| 3 | Show Patient | Check drug allergies, diagnosis, visit history after adding a visit | Jeff Andre | see bugs in Git | no |
| 4 | Add Visit | Input all visit information (negative numbers), press Save button.  Negative number values can not be inserted into related all tables, and all radio button works | Yingyuan Zhang | Negative number values can be inserted into related tables. | no |
| 5 | Add Visit | Input all visit information (positive numbers), press Save button.  Positive number values can be inserted into related all tables, and all radio button works | Yingyuan Zhang |  | yes |
| 6 | Add Visit | Input all visit information (non numerical character), press Save button.  System can show error, and do not execute this insertation | Yingyuan Zhang |  | yes |
| 7 | Add Visit | Do not input all visit information, press Save button.  System can show error about the required value, and do not execute this insertation | Yingyuan Zhang |  | yes |
| 8 | Current Visit | Press Save button on add visit page, system can redirect to Current visit page, and show all the latest added visit data on this page. | Yingyuan Zhang |  | yes |
| 9 | User management\_sign up | Click on the "New User" button to get to a new page and start to add a new user. Input a new user information and press sign up. The system can redirect to a new page--after\_sign\_up page saying "Thank you! You have successfully sign up for this applcation, however, your account need to be approved by administrators. After approval, you will receieve an email notification." | Yike Xue |  | yes |
| 10 | User management\_add new user | After new users sign up, the administrator can activate the selected users by clicking the "Activate" button. After that, the selected users are activated unless the administrator clicks the "dectivate" button. However, the users cannot get an email as the tip mentions above. | Yike Xue | see bugs in Git | no |
| 11 | User management\_sign up | When creating a new user with unformated input, for example, the email address without ʺ@ʺ, then the system would infer the creator that "Please enter a valid email address." and never pass down the error input to the next layer(controller and database). | Yike Xue |  | yes |
| 12 | User management\_sign up | When creating a user with a duplicate User email, the system would report a database error:SQLSTATE[23000]: Integrity constraint violation: 1062 Duplicate entry 'admin@diabetesavior.com' for key 3 | Yike Xue | see bugs in Git | no |
| 13 | User management\_edit | When editing a user with unformated input, for example, the email address without ʺ@ʺ, then the system would infer the editor that "Please enter a valid email address." and never pass down the error input to the next layer(controller and database). Only when all the blanks are filled in in correct format, this edit action would be done and a line of tip shows "User was edited." | Yike Xue |  | yes |
| 14 | User management\_edit | When editing a user, if the editor doesn't want to change the password, just leave the "Password" and "Confirm Password" empty, then the password would not be changed; if editor does want to change the password, then input the new password into these two blanks and then the password would be updated. | Yike Xue |  | yes |
| 15 | User management\_dectivate | If the administrator wants to dectivate a user, just click on the selected user's "dectivate" button, then this user would be dectivated(cannot log in, view patient, visit and many other pages) with a line of tip can be seen at top of the list--"User Yike Xue deactivated!" until this user is activated again by the administrator. | Yike Xue |  | yes |
| 16 | User management\_delete | When a administrator wants to delete a user and never activate it again, he(or she) can simply click on the "delete" button on the right of the user. Then a confirming message box would come out and ask "Are you sure to delete...". Clicking on "Confirm" means this user and all the data related to this user would be deleted permanently. Clicking on "Cancel" means this user would not be deleted at this time. | Yike Xue |  | yes |
| 17 | Add Patient | After logging in, a user can click on the "Add Patient" from the "Patient" tag and get into the add\_patient page. Input all patient information in correct format, press submit and verify all information is correct. Then we can view the information of this new patient on a new page and a line of tip saying "Patient was added. Please save the patient ID so you can search for the patient later" | Yike Xue |  | yes |
| 18 | Update Patient | On the patient view, click on the "Update patient" button can get into the Update Patient page where we can update all the information about a patient after "submit" our inputs. If inputing any invalid information, for example, "s" in ZIP blank, the request would be denied with a line of tip--"Unable to edit patient. Please, try again.". If the inputs are all valid, then we will get a line of tip--"Patient was edited." and view the updated patient. | Yike Xue |  | yes |
| 19 | Update Patient | When updating patient, we might change some of the information used to generate the patient\_ID, for example, a new lastname--Black, which is different from the former lastname--White. After we submit the update data, the patient information would be update but the patient\_ID remains the same. | Yike Xue |  | yes |
| 20 | Search Patient | After clicking on the "Search Patient" label and geting into the Patient\_Search page, the user would be asked to input the patient's PID, name(including First Name and Last Name) and DOB to search this patient accuractly. If all the required information is correct, then the system would redirect to the patient\_show page showing this patient's information. If not, the user would get a tip on the botton of the page, saying "Patient not found, please enter information again or add a new patient." with a "Add New Patient" button bellow it. Attetion: the search is "Case insensitive". | Yike Xue |  | yes |
| 21 | Delete Patient | When viewing a patient information on the patients\_show view, the user can click on "Delete Patient" button to delete the patient. Before the patient is deleted, a message box would come out and ask "Do you want to delete..." and only when the user click on "Confirm" button can this patient be deleted successfully. After that, the user would get back to Patient Search view. | Yike Xue |  | yes |
| 22 | List Patient | For an administrator user, he(or she) can also view the list of all the patients he(or she) created by clicking on the "List Patients". In the List Patient page, some basic information of the patients are shown for the convenience of searching patients quickly. Clicking on the Patient ID then the user can get to the Patient Information page of this selected patient. | Yike Xue |  | yes |
| 23 | Add Visit | On the visits\_add page, if the user inputs any invalid information and clicks on "Save" button, for example, inputs "s" in Weight Goal blank, which is supposed to be a number, the system would report a Database Error: SQLSTATE[HY000]: General error: 1364 Field 'prescriber\_username' doesn't have a default value | Yike Xue | see bugs in Git | no |
| 24 | Add Visit | On the visits\_add page, if the user inputs all the valid information and changes the type unit more than once and clicks on "Save" button, the system would report a Database Error: SQLSTATE[HY000]: General error: 1364 Field 'prescriber\_username' doesn't have a default value. | Yike Xue | see bugs in Git | no |
| 25 | Run Algorithm | Add visit with 3 first medicine allergies = yes, and A1c at 6.6, 7.6 and 9.1. Then check algorithm results, mono therapy, dual therapy, and triple therapy without any of the allergic drugs. | Jeff Andre |  | yes |
| 26 | Accept Algorithm Results | Check accept with run algorith test. | Jeff Andre |  | yes |
| 27 | Edit Algorithm Results | Check edit with run algorithm test. | Jeff Andre |  | yes |
| 28 | Run Algorithm | Check run algorithm with other allergies = yes including all set to yes. | Jeff Andre |  | yes |
| 29 | Login | Before the users get access to the data of patients, they have to log in. After type in the right username(email) and password, the user can view more functions on the page than that before login. The system would redirect to an index if the user successfully logged in and be ready to do the following action. | Yike Xue | see bugs in Git | No |
| 30 | Add Visit... | In many pages that we need to fill in some blanks, the results vary from different browsers. For example, when I input "16%" in " Glycated hemoglobin (A1C):" on Add Visit page, the Chrome would remind me that "Please fill in this blank" but the IE would just come out of a Database Error: " SQLSTATE[01000]: Warning: 1265 Data truncated for column 'A1c' at row 1" | Yike Xue | see bugs in Git | No |
| 31 | Calculators\_BloodGlucoseLevel | Click on "Blood Glucose Level" and get into the new page, where we can view a table showing the relationship between A1C value and eAG value, and the diagnosis based on these values. We can adjust the value up and down and view the change of the other fields' values or messages. | Yike Xue |  | Yes |

## Defect Management

* + - For every iteration process, the team members who find the defects should update it to the GitHub.
    - Define defects: any obstacles for user including program crash, data corruption, display of an error message.
    - Use “Pivotal Tracker” to track the bugs in a user story and discuss how to improve the software. After a bug is modified, it should be committed on the GitHub with a clear ‘commit name’. For example, “Modified bug1 in View2(Problem 3 is solved).”
    - Set the Value At Risk of the number of defects and the Value At Risk of the defect fixing time. Once we cross the line, we will have to adjust our schedule and rearrange the workload.
    - At the end of iteration2, the total number of defects: 35 (26 of which are bugs and 6 of which are enhancements)
      * Open: 15(13 bugs and 2 enhancement)
      * Closed: 20(16 bugs and 4 enhacement)

# Configuration Management Plan

## Configuration items and tools

* + - Version control tool: git
    - Project repository on Github
      * <https://github.com/bumetcs/cs673>
    - Issue (bug) control tool: Github issues
      * <https://github.com/bumetcs/cs673/issues>
    - Deployment tool: [Capistrano](https://github.com/capistrano/capistrano)

## **Change management and branch management**

* + - Everyone should have their own branch.
    - All developments are on “dev” branch.
    - Merge into master branch should be only performed by deployment owner.
    - Feature/bug branches will not be existed before first prototype.

## Code commit guidelines

* + - Pull “dev” branch before starting a day’s work.
    - Commit constantly to local.
    - Push to remote to wrap up working.
    - Deploy via Capistrano script.
    - Jason will do Weekly deployment on Thursday, 5:00 pm.

# References

[CakePHP](http://cakephp.org/)

Tutorials: <http://blog.the-nerd.be/category/cakephp/>

# Glossary

EHR - Electronic Health Record - health record in digital format

CDSS - Clinical Decision Support System - software with designed to assist physicians with decision making tasks.

A1c - blood test that reflects average blood sugar level over last 2-3 months

BMI - Body Mass Index