**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* | 9/25/2013 |
| Jason Lu | Configuration leader | *Jason Lu* | 9/25/2013 |
| Jeff Andre | Project leader, Requirements | *Jeff Andre* | 9/25/2013 |
| Maura Huff | Implementation leader | *Maura Huff* | 9/25/2013 |
| Wenjie Shi(Jenny) | Environment and integration leader | *Wenjie Shi* | 9/25/2013 |
| Yike Xue (Eva) | QA leader | *Yike Xue* | 9/25/2013 |
| Yingyuan (Allen) Zhang | Requirements leader | *Yingyuan Zhang* | 9/26/2013 |
| Yulu Liu | Design leader | *Yulu Liu* | 9/26/2013 |

**Revision history**

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

[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.

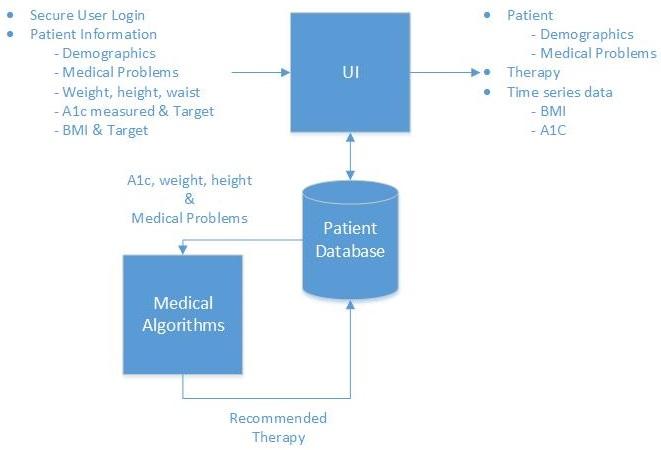


Figure 1. System Diagram for Type II Diabetes Management WEB Application

**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 existing patient data

- delete patient from the database

- search patients with PID, First or Last name

**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

- 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 (A1C, height, weight, waist)

- Services (medications, therapy)

**3.1.3 Update Patient data**

The user should be able to update existing patient demographic and medical data.

**3.1.4 View Patient data**

The user should be able to retrieve and view existing patient data.

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.

**3.1.5 Delete Patient from the database**

The user should be able to delete a patient’s record. Deleted record should be store in separate database table with the information about who and when had deleted the record.

**3.1.6 Search patients**

User should be able to search patients by PID, First Name or Last Name

**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 for user login.

**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. | Implementing Diabetes Management Algorithm that can retrieve individual patient data from the database and represent results. | High |
| 8. | Storing results from #7 in database with an opportunity to edit suggested therapy/recommendations | High |
| 9. | Implementing Medical Calculators: a) Body Mass Index b)A1C-to-Average Blood Glucose Level | Medium |
| 10. | Input validation for all forms | High |
| 11. | Graphical representation of BMI and lab results (A1C level) | Medium |
| 12. | Ability to show step by step algorithm’s decisions (for teaching purposes) | Medium |
| 13. | User password reset | Low |
| 14. | Check user password complexity | Medium |
| 15. | Preventing SQL injections | High |
| 16. | Separating user privileges (professors and students) | Low |
| 17. | Generating patient ID | Medium |
| 18. | Seach patients by patient ID, First or Last Name | High |
| 19. | Calendar for entering DOB | Medium |
| 20. |  |  |

.

## 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 |  |
| 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 | 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 |  | 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 that can be interpreted as the degree to which software possesses a given attribute that affects its quality."

* 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.
  + Performance
    - Number of software problems reported by customer.
    - Number of software problems reported by team.
    - Number of open vs. closed software problems.
    - They are all tracked by GitHub.
  + Reliability
    - Mean time to failure (MTTF).
    - 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

Database Schema

Design Document

Test plan

Test Report

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
      * 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.

## 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 “Github” and “Pivotal Tracker” to track the bug and discuss how to improve the software.
    - 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.

# 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