SOFTWARE DEVELOPMENT PLAN

FOR

Monitoring Heart Disease using Mobile Applications

Student Name	Chenxi Han
Supervisor	Vladimir Brusic
	2018.11.20

Record of Reviews and Changes				
Change ID CI #	**Date Reviewed**	**Date Approved**	**Comment**	**Signature**

TABLE OF CONTENTS

1. Scope 5
1.1. Identification 5
1.2. System Overview 5
1.3. Document Overview 5
1.4. Relationship to Other Plans 6
2. Overview of Required Work 6
2.1. Project and Software to be Developed or Maintained 6
2.2. Program or Project Management Approval Authority 6
2.3. Requirements Management 6
2.4. Software Life Cycle 7
2.5. System Documentation 7
2.6. Schedules and Milestone Constraints or Considerations 8
2.7. Security 8
2.7.1. Development Security 8
2.7.2. Application Software Security 9
3. Plans for General Software Development or Maintenance Activities 9
3.1. Software Development and Maintenance Process 9
3.2. General Plans for Software Development and Maintenance 9
3.2.1. Software Development and Maintenance Methods 9
3.2.2. Standards for Software Products 10
3.2.3. Computer Hardware Resource Utilization 11
3.2.4. Access for Customer Review 12
3.2.5. Reusable Software Products 12
3.2.6. Handling of Critical Requirements 12
3.2.7. Recording Rationale 12
4. Plans for Performing Detailed Software Development Activities 12
4.1. Project Organization 12
4.2. Project Resources 12
5. Contract Management 13
6. Notes 14
7. Appendices or Attachments. 14
8. Appendices 15

1. Scope

1.1. Identification

Project Title: Mornitoring Heart Disease Using Mobile Application

Abbreviation: TBD

Version number: 0.0.0

Release number: TBD

Programming language: TBD

System function description: Collect and analyze healthy data from personal health record, wearables, smart devices and sensor systems. The result of decision making will be presented on mobile application.

Limitations: TBD

1.2. System Overview

Purpose of the system: We will develop a software system for capturing, processing, storing and the analysis of health data and decision-making support needed for maintenance of health of people with heart disease. The system will focus on comprehensive data collection from wearable devices, linked smart devices, early detection of risks and possible complications that will trigger alarms and recommendations to see health care specialists. The system will use commercially available devices and sensors and we will focus on the development of software system.

Applied software: TBD

Project sponsor: Vladimir Brusic.

Users: TBD

Developer: Chenxi Han

Support agencies: N/A

Operation site: UNNC

1.3. Document Overview

Purpose of this document: To help direct the project, monitor progress, enable effective management, help establish process that will result in good quality of the product and ensure that deliverables are clear and done on time.

1.4. Relationship to Other Plans

This project shares specific methodology and development idea with another project Monitoring Pregnancy Using Mobile Application by Xiang Zhang. The overall functions of two systems are similar but aim to different type of healthy problems (heart disease and pregnancy risk).

2. Overview of Required Work

2.1. Project and Software to be Developed or Maintained

The main aim of this project is to develop and implement a system for online analytics of heart disease monitoring data that can be used to predict various complications and rapidly advise patient about emerging risks. Our application will monitor several basic signs (weight, blood pressure, temperature, physical activity, heart rate, and sleep) and integrate these data into a decision-making system.

2.2. Program or Project Management Approval Authority

This project has been approved by Prof. Paul Dempster, the convener of COMP3050 UNNC, and by supervisor Prof. Vladimir Brusic.

2.3. Requirements Management

We have developed the initial data model and process model and collected initial user requirements from Prof. Brusic. This will be refined by Chenxi Han after discussion with Prof. Nasser. We will use recommended practices for effective and efficient requirement management. There are five stages of the development of software requirements:

- Requirements elicitation
  The developer will gather the information from the users and other stakeholders. This will be done through interviews, workshop discussions, analysis of documents and literature, and discussions with supervisor.

- Requirements analysis
  The results from elicitation will be converted into a set of descriptions that are enriched by our analysis and will represent sets of requirements in logical, and complementary ways that are consistent with user needs. The analysis will involve the design of data model and process model.

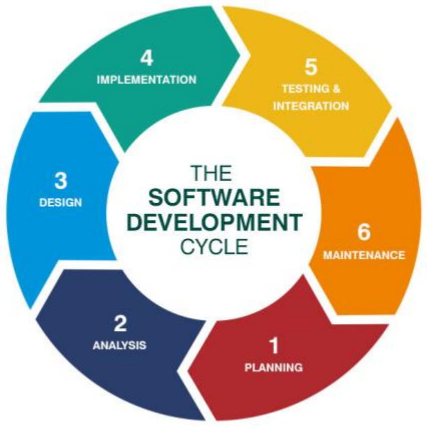
- Software requirements specification
  The sets of requirements from the analysis step will be combined with knowledge from user cases, user stories, functional requirements, and literature to design the software requirement specification in a persistent and well-organized format.

- Validation
  The requirements will be evaluated for compliance with user needs and alignment with project objectives.
- Management
  Requirements will be re-evaluated at each cycle of (agile) product development and changed as needed.

2.4. Software Life Cycle

A typical software life cycle has the following steps:

Planning → Analysis → Design → Implementation → Testing and integration →Maintenance



https://online.husson.edu/software-development-cycle/

We are developing the working prototype of scientific software. The actual product will be developed later.

2.5. System Documentation

In this project we will prepare and maintain the following documents:

- System design diagram
- Software development plan
- Project proposal
- Project website
- Meeting minutes
- Agile project plan with Gantt Chart
- Interim report and final report
- Dissertation first draft and final draft
- User manual
- System requirement
- Other software engineering material

In addition, we have also implemented a system for versioning documents.

2.6. Schedules and Milestone Constraints or Considerations

-	2018/10/9	Project launched
-	2018/10/22	Project proposal submission
-	2019/12/27	Project Preparatory stage ended
-	2019/1/7	Interim report submission
-	2019/4/22	Machine learning development completed
-	2019/5/6	Dissertation submission
-	2019/5/7	Demonstration day

2.7. Software Security

1. Software security (pre-deployment) activities include [1]:

   - Secure software design
   - Development of secure coding guidelines for developers to follow
   - Development of secure configuration procedures and standards for the deployment phase
   - Secure coding that follows established guidelines
   - Validation of user input and implementation of a suitable encoding strategy
   - User authentication
   - User session management
   - Function level access control
   - Use of strong cryptography to secure data at rest and in transit
   - Validation of third-party components
   - Arrest of any flaws in software design/architecture

2. Application security (post-deployment) activities include [1]:

   - Post deployment security tests
   - Capture of flaws in software environment configuration
   - Malicious code detection (implemented by the developer to create backdoor, time bomb)
   - Patch/upgrade
   - IP filtering
   - Lock down executables
   - Monitoring of programs at runtime to enforce the software use policy

2.7.1. Development Security

We are deploying the following practices [2]:

- The program will only implement specified functions; these functions will be thoroughly tested.
- Software development and changes will adhere to the minimum authority to minimize the possible harm;

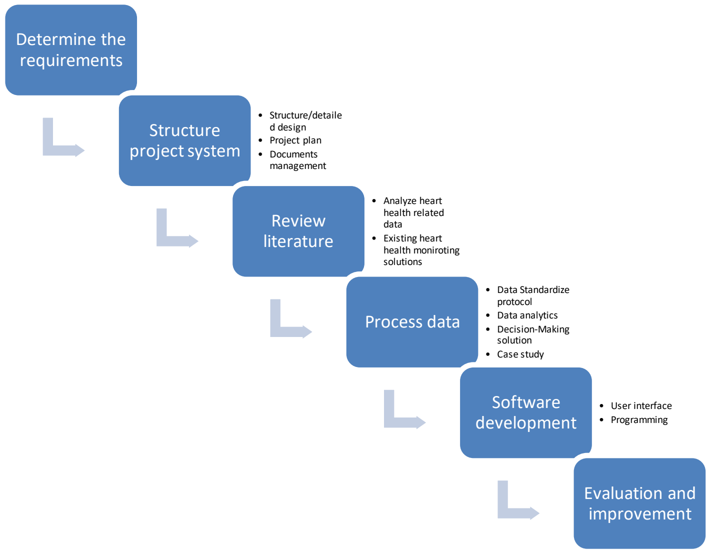
- Constraint of preventing unauthorized threat injection: input will be filtered and perform checks will be performed on all inputs.
- Testing will ensure the accuracy of all search functions, analyses and reporting.
- Exhaustive testing will prevent software crashes.

2.7.2. Application Software Security

Additional deliverables: We will develop a security policy for the project. For our project we will initiate Security Risk Analysis IAW DoDD 5200.28 and establish a System Security Working Group (SSWG) to determine top-level security specifications for the application.

3. Plans for General Software Development or Maintenance Activities

3.1. Software Development and Maintenance Process



Our group will use agile software development [3]. We will keep contact with customers, and update software requirements while designing the software. We will perform multiple iterations to achieve all specifications and satisfy the user requirements. This process is reflected in our project schedule. Since planned product for this project is a functional prototype, we will focus on software development and implementation, while software maintenance plan and activities will be developed later (out of scope of this project). [Relate to the Scope]

3.2. General Plans for Software Development and Maintenance

3.2.1. Software Development and Maintenance Methods

In Software Development part we will focus on the use and deployment of:

a) Standardized data format

b) Data stream protocol

c) Machine learning employment

d) Machine learning optimization

e) System integration

f) Human-Computer interaction

g) Mobile application development

In the Software Maintenance part:

We are developing the working prototype, so the maintenance is out of the scope of this project, nevertheless, our design is modular to facilitate software maintenance.

3.2.2. Standards for Software Products

Optional for Small Requirements

3.2.2.1. This paragraph shall describe or reference the standards to follow for this particular system for representing requirements, design, code, test cases, test procedures, and test results. Reference may be made to other paragraphs in this plan or its appendices and attachments if the standards are better described in context with the activities to which they will be applied. Provide standards for code for each programming language used. They shall include at a minimum: TBD

3.2.2.2. Standards for format (such as indentation, spacing, capitalization, and order of information):

1. Codes for one line should not exceeds 40 characters
2. Indentation (Tab) should be 4-space wide
3. Indentation format should be unified and hierarchical
4. Comment should be ahead of its target
5. Main function should be placed at the bottom

3.2.2.3. Standards for header comments (requiring, for example, name or identifier of the code; version identification; modification history, purpose; requirements and design decisions implemented; notes on the processing (such as algorithms used, assumptions, constraints,

limitations, and side effects); and notes on the data (inputs, outputs, variables, data structures, etc.): TBD

3.2.2.4. Standards for other comments (such as required number and content expectations): TBD

3.2.2.5. Naming conventions for variables, parameters, packages, procedures, files, etc.

1. Constant name should be all uppercase
2. Variable name should be all lowercase
3. Function and class name should start with uppercase and followed by lowercase
4. All names or identifier should have specific meaning related to their roles in order to achieve readability

3.2.2.6. Restrictions, if any, on the use of programming language constructs or features.

Not applicable.

3.2.2.7. Restrictions, if any, on the complexity of code aggregates.

Not applicable.

3.2.2.8. Standards for coding for best network performance, if applicable to the languages and procedures utilized in the program. Refer to Performance Guide for an Automated Information System (AIS) for additional guidance.

See AIS webpage.

3.2.2.9. Other requirements

[Move red text from section 2.3 here or wherever suitable in 2.3.?.? subsections. Delete this explanation after moving the programming practices into 2.3.?.? sections]

3.2.3. Computer Hardware Resource Utilization

TBD

3.2.4. Access for Customer Review

This paragraph shall describe the approach to follow for providing the customer or their authorized representative access to developer and subcontractor facilities for review of software products and activities. It shall cover all contractual clauses concerning this topic.

3.2.5. Reusable Software Products

3.2.5.1. Incorporating Reusable Software Products

May be tailored out for legacy COBOL systems. This paragraph shall describe the approach to follow for identifying, evaluating, and incorporating reusable software products, including the scope of the search for such products and the criteria to use for their evaluation. It shall also identify reuse checkpoints and milestones. Identify and describe candidate or selected reusable software products known at the time this plan is prepared or updated, together with benefits, drawbacks, and restrictions, as applicable, associated with their use.

TBD

3.2.5.2. Developing Reusable Software Products.

May be tailored out for legacy COBOL systems. This paragraph shall describe the approach to follow for identifying, evaluating, and reporting opportunities for developing reusable software products.

TBD

3.2.6. Handling of Critical Requirements

TBD

3.2.7. Recording Rationale

Not applicable. Already discussed in other parts of this document.

4. Plans for Performing Detailed Software Development Activities

4.1. Project Organization

The project is organized in accordance to guidelines issued for COMP3050.

4.2. Project Resources

Describe resources to be applied to project. Include, as applicable:

4.2.1. Personnel resources. Reference MIST.

Developer: Chenxi Han

Group Supervisor: Vladimir Brusic.

4.2.2. Overview of developer facilities to be used, including geographic locations in which the work will be performed, facilities to be used, and secure areas and other features of the facilities as applicable.

Geographic locations: University of Nottingham Ningbo China.

Facilities: Personal computers and UNNC facilities.

Secure areas and others: None

4.2.3. Customer-furnished equipment, software, services, documentation, data, and facilities required.

Smart wearables (smart watch, smart wristband etc.)
Smart phone with internet and Bluetooth availability
IoT sensor devices (smart scale, wireless air detector etc.)
Electronic personal health record

4.2.4. Other required resources, including a plan for obtaining the resources, dates needed, and availability of each resource item.

None

Any time a new Project Manager has been assigned to the project, they will conduct the Project Manager Checklist in the BPD. Append the assessment to the SDP.

5. Contract Management (Not applicable)

Describe the acquisition or management approach by summarizing the procurement concept, type of contracts, contracting management strategy, contract or oversight procedures, and major contractual features to be used on the project. In addition, address aspects of documents that establish procurement authority and govern the actions of participating procurement, contracting, and manufacturing activities. Summarize the requirements for industrial facilities, and planned use of government furnished equipment, services, and facilities. The Project Manager should review the following, as applicable:

5.1. Define the inter-relationships of the participating organizations, test agencies, and other government and non-government entities involved in, or supporting, the project.

Not applicable.

5.2. Identify required reports to senior management and the participating organizations, and required contractor inputs, if necessary.

Not applicable. Reporting requirements are described elsewhere.

5.3. Describe the approach for accomplishing transfer and turnover of the project deliverables to the customer and support organization. Indicate when transfer and turnover planning will begin and who will be involved in the transfer planning group.

Not applicable.

6. Notes

If included, this section shall contain any supplemental information not elsewhere included in this document.

TBD

7. Appendices or Attachments.

Appendices and attachments are supplemental materials and may provide information published separately for convenience in document maintenance (e.g., charts, classified data, matrices, other plans, etc.). As applicable, each appendix and attachment shall be referenced in the main body of the document where the data would normally have been provided. Appendices and attachments may be bound as separate documents for ease in handling. Appendices shall be lettered alphabetically (A, B, etc.), while attachments are sequentially numbered (1, 2, 3, etc.). The appendices and attachments to this document are divided according to project versus release information. The appendices contain general project information that pertains to and aids in the management of all subsystems. For ease of management, some project-related documents may be included as appendices to the SDP; these can include the CMP, Database Specification (DS), Design Document (DD), etc. Attachments contain information that is specifically related to a release. For each release, there will be an attachment to the SDP numbered in succession (1, 2, 3, etc.). The alphabetically lettered annexes to these release attachments will remain constant; however, they will be prefaced with the same number as the associated release (1A, 1B, 1C, 1D, 2A, 2B, 2C, 2D, etc.).

TBD

8. Appendices

All appendices should have successive alphabetic designations. The following shall be appended to the SDP:

8.1. **Appendix A** (ACRONYMS & ABBREVIATIONS: (if the project uses other acronyms and abbreviations than the standard set listed in the BPD Acronyms)

8.2. **Appendix B** (GLOSSARY: (if the project needs to define other terms than those listed in the BPD Definition of Terms)

8.3. Reference List

[1]. Chakraborty M. Application security vs. software security: What's the difference? [Internet]. Synopsys. 2016 [cited 28 November 2018]. Available from: https://www.synopsys.com/blogs/software-security/application-security-vs-software-security/

[2]. Keramati H, Mirian-Hosseinabadi SH. Integrating software development security activities with agile methodologies. InComputer Systems and Applications, 2008. AICCSA 2008. IEEE/ACS International Conference on 2008 Mar 31 (pp. 749-754). IEEE.

[3]. Martin RC. Agile software development: principles, patterns, and practices. Prentice Hall; 2002.