**Title of the Capstone Project**

A Manuscript

Submitted to

the Department of Computer Science

and the Faculty of the

University of Wisconsin-La Crosse

La Crosse, Wisconsin

by

**Student’s Full Name**

in Partial Fulfillment of the

Requirements for the Degree of

**Master of Software Engineering**

Month in Full (e.g. January), Year (4 digits)

**Title of the capstone project**

By Student’s Full name

We recommend acceptance of this manuscript in partial fulfillment of this candidate’s requirements for the degree of Master of Software Engineering in Computer Science. The candidate has completed the oral examination requirement of the capstone project for the degree.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Adviser’s name Date

Examination Committee Chairperson

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Examiner’s name Date

Examination Committee Member

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Examiner’s name Date

Examination Committee Member

# Abstract

Last name, First name, Middle Initial (if any)., “Title of the capstone project”, Master of Software Engineering, Month in full, Year (4 digits), Adviser’s name.

This is where the abstract goes. It is generally around 350-500 words long. You can write it as one paragraph or multiple paragraphs. If it has more than three paragraphs, it does not sound like an abstract.

# Acknowledgements

This page itself is optional. You can skip it if you would like to. There is no formatting requirements or words limit for this section. But it is preferable to limit the acknowledgements within this page.

# Table of Contents

[Abstract iii](#_Toc374036093)

[Acknowledgements iv](#_Toc374036094)

[Table of Contents v](#_Toc374036095)

[List of Figures vi](#_Toc374036096)

[List of Tables vii](#_Toc374036097)

[Glossary viii](#_Toc374036098)

[1. Introduction 1](#_Toc374036099)

[1.1 Background 1](#_Toc374036100)

[1.2 Motivation 2](#_Toc374036101)

[2. Requirements and Assumptions 4](#_Toc374036102)

[2.1 Requirements 4](#_Toc374036103)

[2.2 Assumptions 5](#_Toc374036104)

[3. Design 6](#_Toc374036105)

[3.1 Approach 6](#_Toc374036106)

[3.2 Database Design 6](#_Toc374036107)

[3.3 Interface Design 8](#_Toc374036108)

[3.4 Architecture 13](#_Toc374036109)

[3.5 Question Types 15](#_Toc374036110)

[4. Implementation 19](#_Toc374036111)

[4.1 Technology 19](#_Toc374036112)

[4.2 Difficulties 20](#_Toc374036113)

[4.3 Validation 20](#_Toc374036114)

[5. Testing 26](#_Toc374036115)

[6. Deployment 28](#_Toc374036116)

[7. Future Work 30](#_Toc374036117)

[8. Bibliography 32](#_Toc374036118)

[Appendix 33](#_Toc374036119)

# List of Figures

[Figure 1: Iterative Development Model 6](#_Toc374036120)

[Figure 2: ER Diagram for the Toolkit 7](#_Toc374036121)

[Figure 3: Toolkit Interface 8](#_Toc374036122)

[Figure 4: Select Project (Optional) 9](#_Toc374036123)

[Figure 5: Rename Project 10](#_Toc374036124)

[Figure 6: Page Preview 11](#_Toc374036125)

[Figure 7: Navigation Tree 12](#_Toc374036126)

[Figure 8: Generated Survey Tool Interface 13](#_Toc374036127)

[Figure 9: Class Diagram of the Toolkit 14](#_Toc374036128)

[Figure 10: Question Type – Text 16](#_Toc374036129)

[Figure 11: Question Type - Decimal Number 17](#_Toc374036130)

[Figure 12: Question Type - Multiple Choice 18](#_Toc374036131)

[Figure 13: Project name Validation 22](#_Toc374036132)

[Figure 14: Navigation Tree Errors 23](#_Toc374036133)

[Figure 15: Question Setup Errors 23](#_Toc374036134)

[Figure 16: Page Preview Errors 24](#_Toc374036135)

[Figure 17: Generate Survey Errors 24](#_Toc374036136)

[Figure 18: Generated Survey Tool Errors 25](#_Toc374036137)

[Figure 19: Physician Station - Male Patient 33](#_Toc374036138)

[Figure 20: Physician Station - Female Patient 34](#_Toc374036139)

[Figure 21: Question Class Diagram 35](#_Toc374036140)

[Figure 22: Page Class Diagram 36](#_Toc374036141)

[Figure 23: DatabaseAccess Class Diagram 36](#_Toc374036142)

[Figure 24: CodeGenerator Class Diagram 37](#_Toc374036143)

[Figure 25: Servlet Class Diagram 37](#_Toc374036144)

# List of Tables

[Table 1: Test Cases 27](#_Toc374036145)

# Glossary

Items in this section must be alphabetically arranged. Check with your adviser on what to be included in the glossary. It is highly recommended **not to** include general, well-known and well-defined terms such as “Java”, “Testing”, “Verification” and so on. Tools, software frameworks and utilities that are used in the capstone project can be listed here. There is no limit on the number of items to be included in the glossary but it is recommended not to exceed three to four pages. See some sample items below.

**JavaServer Faces (JSF)**

A specification for constructing component-based user interfaces for Java web applications.

**JavaServer Pages (JSP)**

A technology that uses Java to assist in the creation of dynamic web pages.

**jQuery**

A cross-browser JavaScript library that aims to simplify client-side scripting for HTML.

**Modal Window**

A child window (often called a popup or dialog box) that requires user interaction before focus can be returned to the parent window.

This is the beginning of the main text for the report. Some sample pages are included here to show the fonts for chapter, section and subsection headings as well as how to include figures and tables. Do not change the fonts and style; just edit the headings and the contents.

#

# Introduction

Global Partners is a non-profit organization associated with Gundersen Health System, a comprehensive health care network whose headquarters are located in La Crosse, Wisconsin [1]. Created in 2008, Global Partners’ goal is to develop long-term, sustainable relationships and community-to-community partnerships that expand beyond Gundersen Health System’s typical borders [2]. One such partner community is the Matagalpa district of Nicaragua. Nicaragua is the second poorest country in the Western Hemisphere. Its mountainous terrain and rural disposition make it difficult for its citizens to access necessary services and opportunities. Global Partners-Nicaragua aims to improve the overall health of the people and their communities by helping communities secure adequate nutrition, a clean water supply, and safe housing.

In 2011, Global Partners-Nicaragua collected data on the health needs and strengths of a community named Santa Celia [2]. Information pertaining to current living conditions was gathered and a baseline community health assessment was performed to quantify health-related issues. This information was used to prioritize volunteer efforts and will be compared with future assessments as a way to measure the effectiveness of Global Partners’ efforts in Nicaragua.

The health assessment was done with paper and pencil due to the lack of internet availability in rural Nicaragua. The answers to the assessment questions were then manually input into a database. With over 200 villagers assessed, data input took a substantial amount of time. Manually entering data into a database also introduces a risk of human error. Moreover, the quality of data was not as good as it could have been for several reasons. One such reason was that several volunteers filled out various forms regarding different aspects. Accordingly, there were chances for inconsistencies in data entry due to many human volunteers and multiple forms. Another example of an inconsistency that was encountered was from dates being entered in various forms. Since Global Partners’ volunteers were mostly from the United States, they expected dates to be entered in the American format: MM/DD/YYYY. However, some volunteers from countries other than the United States entered the date in the format DD/MM/YYYY. The person manually entering the hand-written data into the database had to make an educated guess as to which format was used for each form. Furthermore, some questions had unreadable or incomplete answers while other questions were not answered at all.

## 1.2 Motivation

In 2012, Dr. Mark Nigogosyan, a member of Global Partners, reached out to the Computer Science department at the University of Wisconsin-La Crosse for assistance in developing an iPad application for the data-collection portion of the health assessment. Dr. Kasi Periyasamy, the director of the Master of Software Engineering program, initiated an independent study with the goal of developing an application that would meet Global Partners’ needs.

As a result, a web-based data-entry application was developed that was run locally on a MacBook Air. The laptop was set up as a server that other laptops, iPhones, iPods, and iPads were able to connect to. Volunteers then used a connected device’s web browser to access the application. Submitted data was stored in a database on the MacBook Air that was hosting the application. The application had validation that ensured all questions were answered with data that met relevant criteria for those questions as specified by Dr. Nigogosyan. In addition, dependencies among the questions were also incorporated; for example, some questions were shown and some were hidden according to previously answered questions. These question dependencies saved the assessor time in not asking questions that do not make sense (e.g. asking a man, a child, or a 90 year old woman if they are currently pregnant). The appendix contains several screenshots of the application that are relevant to this document.

The survey tool worked wonderfully on Global Partners’ next trip to Nicaragua. Not only did they save a lot of time, but their data quality was also improved. However, several changes to the questions were requested which required a substantial amount of work after the independent study had been completed. Realizing that similar requests would be likely after each visit to Nicaragua, the developer was motivated to create a generic web-based toolkit for data collection. With such a generic toolkit, Global Partners, as well as other organizations, would be able to create their own data-entry survey tool.

# Requirements and Assumptions

## 2.1 Requirements

Almost all the requirements for the generic toolkit were derived from the initial prototype that Global Partners used. However, the developer paid considerable attention to ensure that the toolkit can be used by any other group as well. As a result, the following requirements were established for the project:

* The toolkit must have predefined questions types including plain text, multiple choice, and numeric type questions.
* Each question type must have appropriate validation criteria that can be defined via the toolkit while creating the survey tool.
* The toolkit must generate a fully functional data collection tool (referred to as a survey tool) that can be deployed as a web-based application.
* The toolkit must generate a new database, with a table to store the collected data, which will be utilized by the generated survey tool.
* The generated survey tool must validate the data entered against validation constraints defined by the toolkit user.
* Generation of the survey tool is considered to be an independent project within the toolkit. Accordingly, the toolkit must be able to save a user’s projects and reload them later so that the user can continue working where they left off. For the ease of reading, this document uses the word “project” that refers to the development of one instance of a survey tool.
* Both the toolkit and the generated survey tool must be easy for anyone with basic computer skills to use.

# Design

## 3.1 Approach

There are numerous development models in existence, each with its own advantages and disadvantages. Due to the complex nature of this project, the iterative development model was chosen. The iterative (incremental) development model involves designing, developing, and testing repeatedly, in a cyclical manner, until the software is fully functional and meets all requirements. Each iteration begins as soon as the previous one is completed. The diagram in Figure 1 illustrates the iterative development model used for this project.



Figure 1: Iterative Development Model

## 3.2 Database Design

MySQL was chosen for the relational database management system (RDBMS) for both the toolkit and the generated survey tool mainly because it is a free and open source product. The entity relationship (ER) diagram for the toolkit is shown in Figure 2. The database design for the toolkit consists of three tables: survey\_project, page, and question. There is a one-to-many relationship between the survey\_project and page tables and between the page and question tables. Each table’s primary key is an auto-incrementing integer. As shown in Figure 2, the primary key of survey\_project, iProjectID, is a foreign key in page. Similarly, iPageID, the primary key of page, is a foreign key in question. These constraints give the tables meaningful relationships that satisfy the requirements and assumptions specified previously.

# Testing

Because of the iterative development approach selected for this project, testing was done for each chunk of code that was written, as it was developed. When everything was implemented, the developer then tested the entire application as a whole. There were two areas of testing that were done: testing the interface of the toolkit and testing the interface of the generated survey tool. However, the same ideology was used to test both portions.

To test the interface of the toolkit, basic test cases were developed for each iteration in order to ensure all possible scenarios of the newly implemented functionality worked. For example, when the “Delete Selected” option was implemented, the test cases in Table 1 were evaluated. Additionally, some basic regression testing was performed to ensure that the addition did not break something that was implemented in a previous iteration.

After everything had been implemented and tested iteratively, the full functionality was tested. Several volunteers, as well as the developer, did full run-throughs with the toolkit and the subsequently generated survey tool. Having multiple people test an application is always a good idea since everyone thinks uniquely. Because the developer knows how the code is written, they may not think to test a specific scenario. Meanwhile, the other testers will have no preconceptions and will simply try to use the application. This testing is also a good indication of the usability of the software; if the testers are confused or become frustrated with the interface, then the application could use improvement.



Table 1: Test Cases

# Bibliography

1. (2013, Oct. 2). *Gundersen Lutheran Medical Center* [Online]. Available: <http://en.wikipedia.org/wiki/Gundersen_Lutheran_Medical_Center>
2. (2013). *Global Partners* [Online]. Available: <http://www.gundersenhealth.org/global-partners>
3. Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal, *Pattern-Oriented Software Architecture Volume 1: A System of Patterns*, Wiley, 2000.
4. Robert Futrell, Donald Shafer, Linda Isabell Shafer, *Quality Software Project Management*, Prentice Hall, 2002.
5. Erich Gamma, Ralph Johnson, Richard Helm, John Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1994.
6. David Geary, Cay Hortmann, *Core JavaServer Faces*, 2nd ed., Prentice Hall, 2007.

# Appendix



Figure 19: Physician Station - Male Patient



Figure 20: Physician Station - Female Patient

Figure 21: Question Class Diagram