MedCode Inc.

**APPROVAL**

Gerald Caissy, Client

MedCode Inc.

Client:

Signature Date

Project Manager:

Signature Date

Contents

[**PROJECT SUMMARY** 3](#_Toc437209342)

[**PROJECT OBJECTIVE** 4](#_Toc437209343)

[**PROJECT DELIVERABLES** 5](#_Toc437209344)

[**ASSUMPTIONS & CONSTRAINTS** 8](#_Toc437209345)

[**PROJECT PLAN** 9](#_Toc437209346)

[**PROJECT ORGANIZATION CHART** 10](#_Toc437209347)

[BUDGET 11](#_Toc437209348)

[**CHANGE MANAGEMENT** 12](#_Toc437209349)

[The Change: 13](#_Toc437209350)

[The Hand-off: 13](#_Toc437209351)

[The Change Request Form: 13](#_Toc437209352)

[The notification: 13](#_Toc437209353)

[The Documentation: 14](#_Toc437209354)

[**PROGRESS/STATUS REPORTING** 15](#_Toc437209355)

[**DECISION MAKING** 16](#_Toc437209356)

[**Step 1: Identification of the purpose of the decision** 16](#_Toc437209357)

[**Step 2: Information gathering** 17](#_Toc437209358)

[**Step 3: Principles for judging the alternatives** 17](#_Toc437209359)

[**Step 4: Brainstorm and analyze the different choices** 17](#_Toc437209360)

[**Step 5: Evaluation of alternatives** 18](#_Toc437209361)

[**Step 6: Select the best alternative** 18](#_Toc437209362)

[**Step 7: Execute the decision** 18](#_Toc437209363)

[**Step 8: Evaluate the results** 18](#_Toc437209364)

[**Standards & Conventions** 19](#_Toc437209365)

[**Appendix:** 24](#_Toc437209366)

[CHANGE REQUEST FORM 24](#_Toc437209367)

[TEAM CONTRACT 24](#_Toc437209368)

# **PROJECT SUMMARY**

Gerald Caissy, Instructor

MedCode Inc.

(MCI)

Christopher Sigouin, Project Manager

Start Date: November 29th, 2015

End Date: December 16th, 2015

Budget: $40,448.00

Definitions:

MCI: MedCode Inc.

Client a person or organization using the services of a lawyer or other professional person or company.

Database: a collection of information that is organized so that it can easily be accessed, managed, and updated.

Consultant: a person who provides expert advice professionally.

Resources: Microsoft Office software

Email software

# **PROJECT OBJECTIVE**

The objective of this project to analyze, design and develop a MedCode Inc. system which is intended for healthcare clinic use. The end result will be able to have the system all computerized with an easier task of keeping track of patients, appointments, charges, and insurance claim processing and reducing paperwork without having to add another office clerk to the payroll/budget. We hope to provide an easy system that allows healthcare clinks and other form of business to replace out of date systems/methods without much hassle of learning the new system and help with fast learning and implantation with our system we have presented.

# **PROJECT DELIVERABLES**

|  |  |  |
| --- | --- | --- |
| Phase | Deliverable | Description |
| 1 | **Introduction to System Analysis and Design** |  |
|  | Create organization chart | Chart composed up all the staff and their roles |
|  | Identify three business processes | A transaction processing system, a business support  system, and a user productivity system |
|  | Explain System Types | B2B, vertical and horizontal system packages, or Internet-based solutions |
|  | Considerations on packages |  |
| 2 | **Analyzing the Business Case** |  |
|  | Introductory meeting with Dr. Jones | Computerizing New Century's office systems |
|  | Analyze proposed system |  |
|  | Feasibility types and determinations |  |
|  | Preliminary Investigation |  |
| 3 | **Managing System Projects** |  |
|  | Power point Presentation |  |
|  | MS Word Handout |  |
|  | Project management example |  |
|  | Same as above but PERT/CPM |  |
| 4 | **Requirements Modeling** |  |
|  | List interview individuals |  |
|  | Prepare objectives |  |
|  | Prepare questions |  |
|  | Conduct interviews |  |
|  | Prepare a written summary |  |
|  | Design questionnaire |  |
| 5 & 6 | **Data and Process / Object Modeling** |  |
|  | Context diagram |  |
|  | Prepare UML diagram |  |
|  | Prepare list of data stores and data flows |  |
|  | Prepare data dictionary |  |
|  | Creation/ review of diagrams |  |
| 7 | **Development Strategies** |  |
|  | Overview of proposed system |  |
|  | Economic feasibility analysis |  |
|  | Alternatives and other material |  |
| 8 | User Interface Design |  |
|  | Monthly claim status summary report |  |
|  | Design appointment list & monthly statement |  |
|  | Data validation checks |  |
| 9 | **Data Design** |  |
|  | Initial entity-relationship diagram |  |
|  | Normalize to 3NF |  |
|  | If new entities emerge |  |
|  | Memo of recommendations |  |
| 10 | **System Architecture** |  |
|  | Advantage of internet based architecture |  |
|  | File-server or client/server architechture |  |
|  | Use both online and batch processing |  |
|  | Outline for a system design specification |  |
| 11 | **Managing Systems Implementations** |  |
|  | Plan the testing required |  |
|  | Design FAQ |  |
|  | Recommend a changeover method |  |
|  | Post-implementation evaluation |  |
| 12 | **Managing Systems Support and Security** |  |
|  | Prepare written procedure |  |
|  | Periodic slowdowns |  |
|  | Monitor benefits and costs |  |
|  | Prepare a security issue list |  |
|  |  |  |

**Non-Deliverables**

* Database software. Company will be required to install their own Database software but will be guided and taught how to use it with an expert help them
* Hardware: The company must have able to provide suitable hardware/computers that complies with the system requirements so they won’t run into performance issues

# **ASSUMPTIONS & CONSTRAINTS**

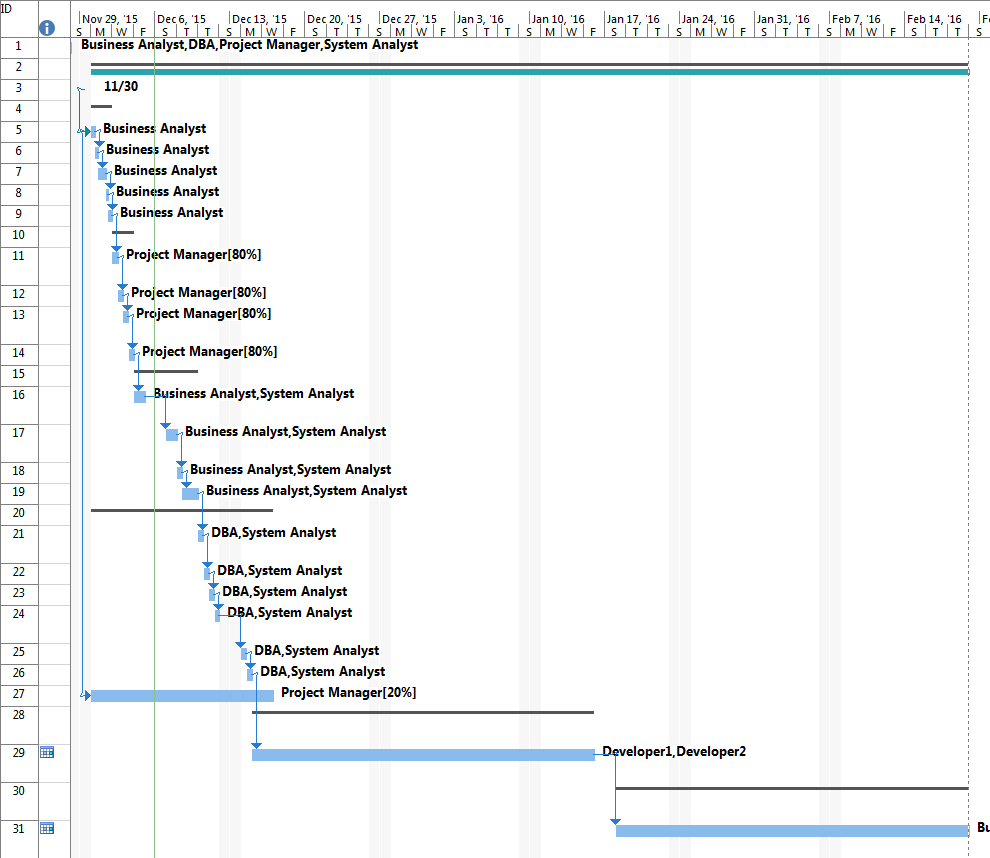
*ASSUMPTIONS*:

* Company has decided to computerize its office systems.
* Staff would be trained, by being able to handle routine maintenance tasks without your assistance.
* Regular routine maintenance will be needed in which it consists of file backups, and updating. Tasks will require about four hours per week and can be performed by a clinic staff member. In both cases, the necessary hardware and network installation will cost about $5,000
* A consultant would be hired by Dr. Jones to study the current office systems and recommend a course of action

*CONSTRAINTS*:

* If needed the problem with work overload would be solved with additional office clerk
* Company's current workload requires three hours of office staff overtime per week at a base rate of $8.50 per hour.
* The current manual system also causes an average of three errors per day, and each error takes about 20 minutes to correct.
* If designed the new system as a database application, you can expect to spend about $2,500 for a networked commercial package.
* New Century probably will need you to provide about 10 hours of initial training and support each week for the first three months of operation

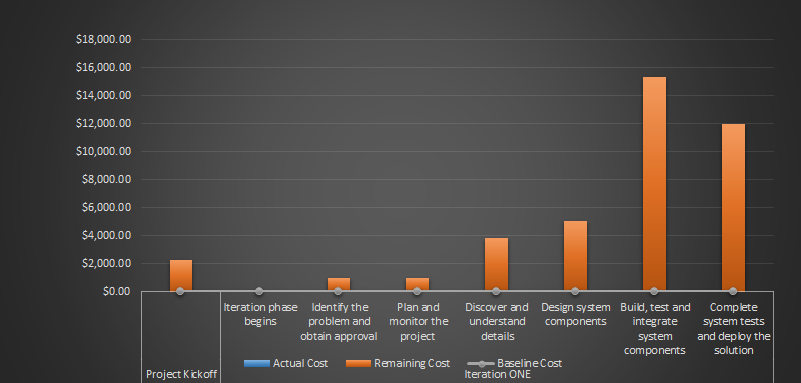
# **PROJECT PLAN**

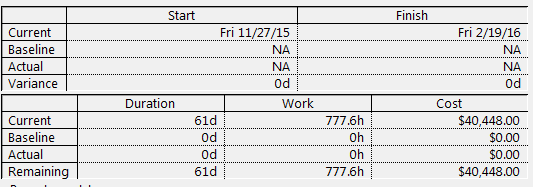


# **PROJECT ORGANIZATION CHART**



# BUDGET





# **CHANGE MANAGEMENT**

In order to ensure high efficiency and integrity, change management will ensure that any and all project alterations proceed smoothly, integrate seamlessly, and are documented correctly (or as much as humanly possible). A change management form is at the center of the intended change, and will hold all pertinent information regarding it. The form alone will not be enough to maintain this process. The following guide will outline all necessary steps to take, so that any transformation in project direction can be handled.

This table represents the five phases of a change request. They are in listed in order of execution and should be executed in his manner. Any deviation from this guide will result in a failed attempt to implement a change successfully and may place the project itself at risk.

|  |  |
| --- | --- |
| The Change: | A single change is the starting point in the change management process. This can represent a number of different items such as:   * New operation or feature requested by the client * Existing operation for feature requires an alteration * Removal of an operation for feature   A change that alters the project in a range of 50 percent or greater will nullify the project completely causing the entire development process to reset. |
| The Hand-off: | Once a project representative has been notified of said change, the next step would be to call a meeting that will include at minimum the following members of the current project:   * Project Manager(s) * Business Analyst(s)   The meeting will be to review the described change and discuss a course of action. All reviewed information should be aimed at how it will impact the project in the areas of time and cost. Also a resolution should be agreed upon as well. This phase can take more than one meeting to complete and as many times as necessary. |
| The Change Request Form: | The form is documented with all information gained during the prior phase under each heading. This responsibility belongs to the project manager. Once completed, the document is added to the project documentation, so that the change can be tracked if the need arises. An ID number is generated for the document starting at 001. This will increment sequentially as more change requests are placed. |
| The notification: | The project manager(s) will integrate the change into the current project iterations. All team members will be notified of the change via email with the subject line of the email as follows:  CHANGE REQUEST INTEGRATION – REQUEST ID # - Please review the following items contained  This will be the responsibility of the project manager as well. |
| The Documentation: | At this phase, the change request should be integrated completely into the project iteration cycle and all members notified. Any other documentation required in regards to the change may be handled by the business analyst, systems analyst or project manager of the project team. |

# **PROGRESS/STATUS REPORTING**

All progress and status reports will be mandatory and done on a weekly basis. Templates will be created using Microsoft Word or similar product. Alternate sourced word-processing software should be Microsoft Word compatible. All individuals that are required to submit the report will be asked to fill out the Progress/Status report document, and then emailed to the project manager.

The progress report will **only** be submitted by the project manager. The document will include an updated budget, any change request forms, approved and denied list of changes and any outstanding or current issues which arise during the said week.

All document templates will be attached to this document.

# **DECISION MAKING**



## **Step 1: Identification of the purpose of the decision**

In this step, the problem is thoroughly analyzed. There are a couple of questions one should be asked when it comes to identifying the purpose of the decision.

* What exactly is the problem?
* Why should the problem be solved?
* Who are the affected associates of the problem?
* Does the problem involve a deadline or a specific time-line?

## **Step 2: Information gathering**

In the process of solving the problem, you will have to gather as much as information related to the factors and associates involved in the problem. For the process of information gathering, tools such as 'Check Sheets' can be effectively used.

## **Step 3: Principles for judging the alternatives**

In this step, the baseline criteria for judging the alternatives should be set up. When it comes to defining the criteria, organizational goals as well as the corporate culture should be taken into consideration.

## **Step 4: Brainstorm and analyze the different choices**

For this step, brainstorming and listing down all the ideas is the best option. Before the idea generation step, it is vital to understand the causes of the problem and prioritization of causes.

For this, you can make use of Cause-and-Effect diagrams. Cause-and-Effect diagram helps you to identify all possible causes of the problem.

Then, you can move on generating all possible solutions (alternatives) for the problem in hand.

## **Step 5: Evaluation of alternatives**

Use your judgement and decision-making criteria to evaluate each alternative. In this step, experience and effectiveness of the judgement principles come into play. You need to compare each alternative for their positives and negatives.

## **Step 6: Select the best alternative**

Once you go through from Step 1 to Step 5, this step is easy. The selection of the best alternative is an informed decision since you have already followed a methodology to derive and select the best alternative.

## **Step 7: Execute the decision**

Convert your decision into a plan or a sequence of activities. Execute your plan by yourself or with the help of subordinates.

## **Step 8: Evaluate the results**

Evaluate the outcome of your decision. See whether there is anything you should learn and then correct in future decision making. This is one of the best practices that will improve your decision-making skills.

# **Standards & Conventions**

1. **Executable & Source code file naming**

Software development languages have their own restrictions on naming conventions for source code files. As we will ultimately have to abide by those rules, naming conventions for both executable and source code, will be generalized.

Any and all filenames will match the content contained within them and will represent that content as close as humanly possible. Filenames will have the following defined constraints (although subjective to the technology being implemented)

* Limited to 20 characters maximum
* Filenames will be in lowercase formatted
* There will be no spaces in any filenames
* Filenames will be required to start with an alphanumeric character
* Filenames must have an extension ( IE .cpp, .java, .py )

1. **Source code internal module header documentation content**

For each source code document produced within the scope of this project, a specific license header shall be applied to the beginning of each file (as much as possible based on syntax requirements and application execution needs). The following header will be placed on line one of each document produced:

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**MedCode Incorporated**

**2015**

**All rights reserved**

**This file is subject to the terms and conditions defined in file 'license.txt', which is part of this source code package.**

**Program name: < Program name here >**

**Version: < Current version here >**

**Dependencies: < Dependencies of current file here >**

**Database file: < Database files required here >**

**Change History: < See programChangeHistory.log >**

**\*/**

|  |  |
| --- | --- |
| Program Name: | The codename of the project will be listed here. This will show the most updated version of the codename that was last used. |
| Version: | Version will be the most recent version number extracted from the programChangeVersion.log file located within the source code package. |
| Dependencies: | Any dependencies that this source code document requires will be listed here in a readable format as follows:  Dependency name – Reason for dependency – What uses this dependency |
| Database files: | Any files that are required for this document that are related to a database technology in any form. ( IE. Configuration, Connection, etc ) |
| Change History: | All change history information to this document can be found within the filename ' programChangeHistory.log ' which can be located within the source code package. All changes to this document must also be recorded within this file if any are completed. |

Function or method standard headers shall utilize the information below before each declaration:

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**MedCode Incorporated**

**2015**

**All rights reserved**

**This file is subject to the terms and conditions defined in file 'license.txt', which is part of this source code package.**

**Function name: < Function or method name here >**

**Version: < Current version listed here >**

**Description: < Description placed here >**

**Inputs: < Inputs listed here >**

**Outputs: < Outputs listed here >**

**Return value: < Return values listed here >**

**Change History: < See functionChangeHistory.log >**

**\*/**

|  |  |
| --- | --- |
| Function name: | The codename of the project will be listed here. This will show the most updated version of the codename that was last used. |
| Version: | Version will be the most recent version number extracted from the programChangeVersion.log file located within the source code package. |
| Description: | A brief description of the function can be placed here. It must be descriptive enough so that, if a developer with no knowledge of the application were to review it, they would be able to utilize it with minimal effort and research. |
| Dependencies: | Any dependencies that this source code document requires will be listed here in a readable format as follows:  Dependency name – Reason for dependency – What uses this dependency |
| Inputs: | Inputs are parameters, modules or related dependency injection that affects the documented function or method. |
| Outputs: | Outputs are all sources of output contained within the function itself (including the return values). |
| Return values: | Return values are documented under this heading, including the given data type that will be returned. |
| Change History: | All change history information to this document can be found within the filename ' functionChangeHistory.log ' which can be located within the source code package. All changes to this function must also be recorded within this file if any are completed. |

1. **Data base & table naming**

Database conventions for the database itself including all table and field values will follow an ORACLE MySQL standard which can be found at this link:

<https://docs.oracle.com/cd/E18727_01/doc.121/e12897/T302934T458266.htm>

If this technology is not used within the scope of the project, then it is the responsibility of the project manager to ensure that this is notified to all members and the convention link is submitted for that particular technology type.

1. **Variable / function / class naming**

The table below addresses each source code type and how conventions should be applied to each type.

|  |  |  |
| --- | --- | --- |
| **TYPE** | **EXAMPLE** | **CONVENTION** |
| Class | class User | Class names should be considered as nouns and in singular form (unless there is a specific reason that plural is required ). Format is in ' UpperCamelCase ' with the first letter being capitalized. Avoid acronyms and abbreviations (unless it is widely used such as URL or HTML ). |
| Methods | isVerified() | A method name should be considered and thought of as a verb. Format will be kept in ' lowerCamelCase '. Methods can be either singular or plural depending on their desired purpose. Readability and specificity of high importance when choosing the name of a method (IE. getStuff vs. getUserList ). |
| Variable | String password | Variable names should be in ' lowerCamelCase ' format and designed to be mnemonic. One character variables should only be used if the variable will be discarded within its current scope. |
| Constants | PATTERN\_TO\_MATCH | These variable types should be completely uppercase in format. Underscores can act as spacing between words to improve readability. May also contain digits but not as the first character. |

1. **Document naming**

This convention will follow the same guidelines as section B.

# **Appendix:**

# CHANGE REQUEST FORM



# TEAM CONTRACT

