*Florida International University*

*School of Computing and Information Sciences*

CIS 4911 Senior Project

Test Case Management System Ver-2.0

Feasibility Study and Project Plan

January 2015

**Group Members**

Artiom Tiurin

**Mentors**Justin Phillips

Tariq King  
Dionny Santiago

**Instructors**Masoud Sadjadi  
Juan Caraballo

**Abstract**

The present work constitutes the overview description, base requirements, planning and initial project organization of a Test Case Management System. This deliverable will provide a general picture of the necessary context and background to understand and define the problems and challenges that emerge from managing large sets of test cases at corporations such as Ultimate Software. The feasibility study covers the positive and negative aspects of the current solution Ultimate Software has implemented to manage test case suites in addition to other potential solutions. The listing of clear definitions of Ultimate Software’s requirements is an essential guide of this document towards the implementation of a secure and user friendly application. The elaborated solutions as well as the studied alternatives presented are focused on improving the overall experience of documenting and managing all the tasks related to software testing at Ultimate Software. The human, hardware, software and other resources estimated on the planning section of this deliverable are in proportion to the system’s functional and nonfunctional requirements.

·

Table of Contents

[**1.** **Introduction** 4](#_Toc406071511)

[**1.1. Problem definition** 4](#_Toc406071512)

[**1.2. Background** 5](#_Toc406071513)

[**1.3. Definitions, Acronyms, and Abbreviations** 5](#_Toc406071514)

[**1.4. Overview of document** 6](#_Toc406071515)

[**2.**  **Feasibility Study** 6](#_Toc406071516)

[**2.1. Description of Current System** 6](#_Toc406071517)

[**2.2. Purpose of New System** 7](#_Toc406071518)

[**2.3. High-level Definition of User Requirements** 7](#_Toc406071519)

[**2.4. Alternative Solutions** 8](#_Toc406071520)

[**2.4.1. Description of Alternatives** 9](#_Toc406071521)

[**2.4.2. Selection Criteria** 9](#_Toc406071522)

[**2.4.3. Analysis of Alternatives** 9](#_Toc406071523)

[**2.5. Recommendations** 10](#_Toc406071524)

[**3.**   **Project Plan** 11](#_Toc406071525)

[**3.1. Project Organization** 11](#_Toc406071526)

[**3.1.2. Hardware and Software Resources** 12](#_Toc406071527)

[**3.2. Identification of Tasks, Milestones and Deliverables** 12](#_Toc406071528)

[**4. Appendix** 14](#_Toc406071529)

[**4.1. Appendix A - Project schedule (Gantt chart or PERT Chart)** 14](#_Toc406071530)

[**4.2. Appendix B – Feasibility Matrix** 15](#_Toc406071531)

[**4.3. Appendix C – Cost Matrix** 16](#_Toc406071532)

[**4.4. Appendix D - Diary of Meetings** 18](#_Toc406071533)

[**5.** **References** 18](#_Toc406071534)

# **1.** **Introduction**

The current chapter presents a general overview of the Test Case and Automation Management software. Initially, the problem definition of the project is depicted in order to provide a context that leads to the explanation, purpose and scope of the proposed system. Prominent definitions, acronyms and abbreviations used in this document are defined in section 1.3. In conclusion, the final section encapsulates a complete overview and main points discussed in this document.

## **1.1. Problem definition**

Software testing is a core decision stage within a functional software’s development and operation life cycle. Testing determines the correctness, completeness, and overall compliance with a client’s requested features and requirements. Companies such as Ultimate Software depend heavily on efficient and practical management of software testing in order to confidently validate their software products with their customer requirements. While Ultimate Software’s testing team manages large sets of test cases using Microsoft Test Manager (MTM) in conjunction with Microsoft’s Team Foundation Server (TFS), several shortcomings have been found in MTM. Additionally, MTM cannot connect to 3rd party systems so Ultimate test engineers must used multiple user interfaces in order to maintain their test plans. The current software testers’ experience when creating, editing and managing the tasks related to testing in general could improve in order to significantly and directly enhance the time it takes to fully test software as well as the overall assertiveness and ease of spotting blocking issues that ultimately affect the customer’s satisfaction with a given software product.

## **1.2. Background**

Software testing, at its core, is a well defined investigation into a given software to provide stakeholders with information about the level of quality of the software. Good software testing incorporates not only a series of test cases, but a sound testing strategy which includes a well defined test plan and efficient test case management. Software testing is usually supported by test case management and project management software tools which help ensure that every feature is tested and maximum code coverage is achieved within tests.

TFS is a collaboration platform at the center of Microsoft’s application lifecycle management solution. TFS provides agile project management, source code management (SCM), build; deployment; and test services, bug tracking, and many other life cycle management tools. TFS features may be accessed via Visual Studio, MTM, web GUI, or APIs. MTM extends TFS’s test capabilities providing additional methods for creating test suites, cases, and steps; manual test step-by-step guides and exploratory testing; rich media capturing tools; and feedback. Test cases are individual tests on specific functionality or feature and consist of a set of test steps which a manual tester follows or automation executes. These test cases are organized under test cases within test suites which can be tied to features, functionality, sprints, among others.

In cohesion with the TFS test plan, Ultimate Software’s internal automation software named Phoenix in conjunction with TeamCity integrate in order to automate test scripts and safe the test results. Automation files may be viewed via Phoenix as well as logs in order to acquire further details into the test case automation processes as well as errors, pass and test results.

## **1.3. Definitions, Acronyms, and Abbreviations**

**COCOMO:** Constructive Cost Model

**MTM:** Microsoft Test Manager

**SCM:** Source Code Management

**SW**: Software

**TFS:** Team Foundation Server

**WA:** Work Activity

## **1.4. Overview of document**

This document incorporates the early definitions of the project necessary to complete an accurate feasibility study for the development of the Test Case Management System. The primary objective of this document is to present the project’s overview found on this chapter as well as its constraints and alternatives. The second chapter of this document provides a feasibility analysis inquiring all the constraints and alternatives to consider before the implementation of a Test Case Management System for the client: Ultimate Software. On chapter 3, the software project’s development and management organization are also described in order to analyze dependencies and interrelations through a hierarchical chart. To conclude the appendix provides a Gantt chart of activities, costs and feasibility matrices as well as an estimation of a budget obtained by utilizing the algorithmic software cost estimation model named COCOMO.

# **2. Feasibility Study**

This chapter presents the feasibility analysis including the current system, its limitations and constraints, the purpose of the new system, and proposed alternatives. The purpose of the new system includes high level requirements for the proposed new system which were elicited from the client. The alternative solution was developed by analyzing the limitations and constraints of the current system in regard to the problem provided by the client. The chapter closes providing recommendations as to which proposed system would be the best investment for the client.

## **2.1. Description of Current System**

The current system, within the scope of this project, is centered around MTM utilized in conjunction with TFS, in addition to the Phoenix test automation system. These applications and services are used to create test suites, cases, and steps in addition to providing autonomous build, test, and deployment services; as well as reporting services. Also, MTM supports manual and exploratory testing via reporting and rich media capturing capabilities. The system provides for the ability to then query reports and tests.

The most major limitations of the current system are the usability governing creation or cloning of test suites, cases, the accessibility of test metrics data, and steps and the duplication of effort required in test planning. The usability limitations exist within the structural limitations of test cases, drill-down navigation, and the complexity/limited feature set of the user interface, and having multiple user interfaces. The client would like the ability to create two levels of test cases, one that is high level and one at an implementation details level; which the current system does not support. MTM provides a drill- down navigation of test suites and cases that allow for attachments to be added. However, viewing these attachments using the drill-down navigation creates for an arduous process. Creating similar test plans requires much duplication of effor, especially across multiple systems. Phoenix test result data, specifically in the area of test failure causes, is not readily accessible from the test planning and maintenance interfaces and not currently stored in a human readable format. Lastly, in many areas test case creation is too complex, with too busy of a user interface while also not providing necessary features such as copying of cases and steps.

Another limitation of the current system is that MTM cannot associate test suites, cases, or steps with the code repository (SVN, Git, CVS, etc…). Thus, there does not exist a method for which the tester can be notified about file changes associated with a given test element.

When it comes to automating and storing test case results, Phoenix and TeamCity softwares at Ultimate are very efficient; yet the display of this results are not as direct and efficient for a Software Tester as it could be. Since a software tester runs dozens of test cases a day, he or she relies heavily on endless amounts of logs scattered across different applications to accurately keep track of failures and its regular tendencies. Common causes of legitimate or false failures are often overlooked due to the large amount of details required on logs provided by the current system. In effect, to analyze test results and conclude its resolution with the correct form of action on a timely matter turns to be difficult.

In addition to viewing the test results in a quick and easy to comprehend format; testers require the ability to very quickly and easily create a new test project, link test cases to their associated code repository files, and notify the tester to changes in these files. Currently, the previous tasks may not be accomplished within one single user friendly application. Finally, MTM does not provide a mechanism in which the results of automated tests may be displayed. This provides for a fragmented testing system which requires multiple applications to determine the state of the tests and displaying test associations.

## **2.2. Purpose of New System**

The new system shall provide the client with the ability to create test plans; that is the creation of test suites, cases, and steps, efficiently. The system shall interface with TFS and be configurable for both TFS communication and code repository linking, in addition to publishing test plans to a stand alone database. Furthermore, the system shall allow test elements to be associated with a code repository. The system shall also support adding attachments to test elements. The system shall support a human readable format of the test result data using a user friendly system of charts. This entire system shall be secured behind a single sign-on login which is linked to domain accounts. Each feature within the proposed system shall provide an easy to use, simple interface to the system actors.

## **2.3. High-level Definition of User Requirements**

The implementation of the user stories of Test Management System will include folowing main and modified functionalities that the system shall perform:

Ver-2.0:

* System shall use a Bayesian Network Analysis on the test results; this will be required to determine the cause of failure for the test
* System shall establish building out a platform to aggregate automation results, visualize these results across multiple test cases, identify test failures over time, and build a report of the most common test failures.
* System shall create notifications or some form of callout whenever an automated test found a bug
* System shall create a visual comparison of test priorities vs. the number of bugs found by the test, and any other statistics which may be useful for a tester

Ver-1.0:

1. The system shall allow for testers within the configured Windows domain to have single sign-on access using the account the tester is signed into a Windows machine with.
2. The system shall provide a mechanism to configure the test management backend system.
3. The system shall provide an interface for creation of test suites, cases, and steps with fluid, intuitive authoring experience.
4. The system shall provide testers with the ability to add attachments and associate them with test suites, cases, and steps. The attachments should then be viewable with ease and with as few clicks as possible.
5. The system shall provide an interface to configure the integration with a code repository.
6. The system shall provide a mechanism to associate test suites, cases, and steps with a file in the configured code repository or automation associated with the project.
7. The system shall display test suites, cases, and steps in a clean explorer or tree view throughout the user navigation within the system.
8. The system shall provide a means to display data categorically, organically, and in a logical view.
9. The system shall provide a simple search interface for searching the test suites, cases, and steps.
10. The system shall provide a logout mechanism so that the current user may be logged out and a user other than the user logged into the current Windows machine may login to the system.
11. The system shall be secured behind the login mechanism. Guests will only be able to visit the login feature.
12. The system shall provide an access table with a list of approved users from within the domain user group, this access table shall be configurable by pre-defined users on an individual basis.
13. The system shall provide functionality for extending the test element model to other 3rd party management systems.
14. The system shall provide a set of metrics which track test case failures, failure types, and source code attached to failures.
15. The system shall provide a visual representation of the failure metrics.

## **2.4. Alternative Solutions**

This section presents the alternative solutions proposed and the research performed in order to determine the best solution investment for the client. By providing alternative solutions the feasibility of the solution was compared in order to minimize the client’s risk with the knowledge that all options were explored.

## **2.4.1. Description of Alternatives**

**Web Application**

A web application would be developed which would communicate with TFS, Phoenix, and the code repository. This web application would provide a web accessible interface for creating test suites, cases, and steps as well as the other required features.

**Visual Studio Plug-in**

Visual Studio provides a plug-in API which would leverage the Microsoft platform, communicating with the TFS server. The plug-in would expose the TFS test suites, cases, and steps as well as other requested features within the Visual Studio UI.

**ClearQuest Synchronization**

A simple Visual Studio plug-in would be developed which would shadow copy test suites, cases, and steps between ClearQuest and TFS. This would allow for the mature ClearQuest test creation UI to be used in conjunction with TFS.

## **2.4.2. Selection Criteria**

Given that the primary requirement relies on usability, the operational feasibility was one of the most weighed portions of the feasibility analysis. Due to the time constraints on the project the technical feasibility is also very important, including the practicality of the overall solution; developer expertise with the solution implementation; and current licensed software available from the client. This time consideration coincides with measuring the scheduling feasibility. Finally, the economic feasibility was also measured using a cost-benefit analysis.

## **2.4.3. Analysis of Alternatives**

In the analysis of the operational feasibility, the current system was compared to the proposed systems’ capabilities in terms of usability and user acceptance. The Visual Studio plug-in provides an inflexible interface dependant upon the existing Visual Studio constructs, this limitation detracts from the usability of this proposed system. Additionally, many testers may not be familiar with Visual Studio which may cause adoption resistance. Conversely, the ClearQuest solution provides for a mature and well-understood user interface compared to creating a new interface for creating test elements. However, ClearQuest’s interface is dated and does not take advantage of many new usability techniques and narrow the extendability of the user interface options. The web based solution provides for a highly tailored and flexible user experience, in addition to being very extendable. These benefits make the web-based solution the most usable and least resistant to user adoption in terms of operational feasibility.

The technical feasibility was limited when researching the Visual Studio plug-in and ClearQuest solutions due to a lack of developer expertise with developing Visual Studio plug-ins and lack of pre-existing ClearQuest experience in addition to licensing. Whereas the developers both have experience with the development of web applications, there is no licensing, and the extensibility of web applications is very broad.

Scheduling feasibility of the Visual Studio plug-in and ClearQuest applications would not require as much development as they would only require extending existing applications, rather than developing a new system from the ground up, as is the case of the web application. In contrast, there would be a greater period of time required for the developers to become familiarized with developing Visual Studio plug-ins. In addition, the Visual Studio plug-in and ClearQuest applications would require more training time for testers than the more usable web application.

Economic feasibility was determined by the sum of the development cost, future benefit, and licensing costs. Both the Visual Studio plug-in and ClearQuest application incur licensing cost, however the development costs are lower due to less development time being required. The future benefit is inflexible with these applications due to their tight technology coupling. While the web application requires more development cost, it does not have the direct licensing cost and provides the most future-proof solution being loosely coupled and very extensible.

Appendix C presents a feasibility matrix which utilizes the feasibility analysis to score each application based on the four mentioned areas of feasibility. This matrix clearly indicates the best investment for the client in regard to the three solutions proposed.

**2.5. Recommendations**

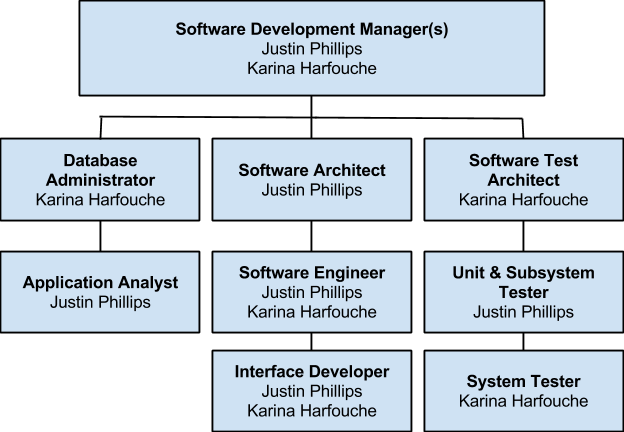
Based on the feasibility analysis and feasibility matrix scoring, the project would be best suited with a web application solution. The web application provides the ability to create a much richer and simplified user experience for testers, is less tightly coupled to one technology, and is far more extensible than the alternative solutions. This provides for a more operational and economically feasible solution while also incorporating the client’s need for a highly usable system. The feasibility matrix in Appendix C presents the solution with the best investment benefit as being the web application, scoring 6 points more than the next solution. Moving forward, it is recommended that the web application solution be chosen as the path forward.

# **3. Project Plan**

The project organization section focuses on the visualization and analysis of the general software organization as well as the software support and service organization. A hierarchical chart is presented in order to illustrate the proposed structure for the Test Case Management System’s working personnel.

## **3.1. Project Organization**

This is the software development organization for the Test Case Management System:



## **3.1.2. Hardware and Software Resources**

The following tentative table specifies all the hardware and software resources needed throughout the planning, development, implementation and testing of the Test Case Management system:

|  |
| --- |
| **Tools** |
| Server with public IP or domain, minimum of 4GB of RAM and 100GB free of hard disk space. |
| Visual Studio 2013 Premium or better |
| GIT Client |
| Microsoft Office Home |
| Team Foundation Server 2013 |
| Microsoft Windows Server or Windows 7 |
| StarUML |
| Microsoft Test Manager |
| Microsoft SQL Server Standard 2012 |

## **3.2. Identification of Tasks, Milestones and Deliverables**

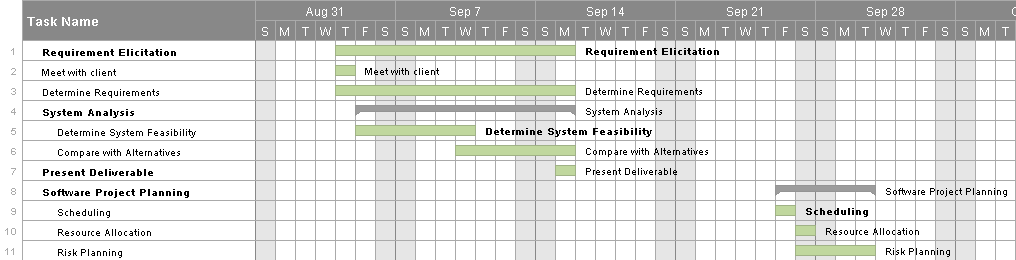
The following table identifies the major tasks, milestones and deliverables along with each respective duration dates visually defined on the Gantt Chart presented in Appendix A:

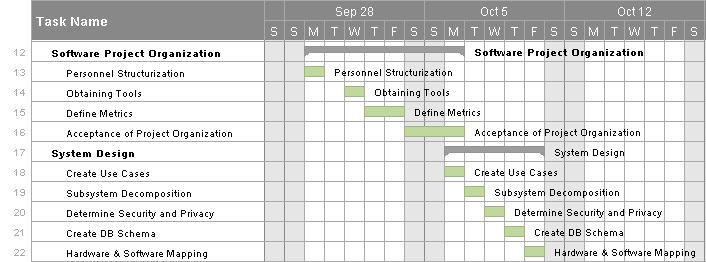
|  |  |  |  |
| --- | --- | --- | --- |
| **Task Name** | **Start Date** | **End Date** | **Duration (days)** |
| **Requirement Elicitation** | **09/04/14** | **09/15/14** | **8** |
| **Meet with client** | **09/04/14** | **09/04/14** | **1** |
| **Determine Requirements** | **09/04/14** | **09/15/14** | **8** |
| **System Analysis** | **09/05/14** | **09/15/14** | **7** |
| **Determine System Feasibility** | **09/05/14** | **09/10/14** | **4** |
| **Compare with Alternatives** | **09/10/14** | **09/15/14** | **4** |
| **Present Deliverable** | **09/15/14** | **09/15/14** | **1** |
| **Software Project Planning** | **09/26/14** | **09/30/14** | **3** |
| **Scheduling** | **09/26/14** | **09/26/14** | **1** |
| **Resource Allocation** | **09/27/14** | **09/27/14** | **1** |
| **Risk Plan** | **09/27/14** | **09/30/14** | **3** |
| **Software Project Organization** | **09/29/14** | **10/06/14** | **6** |
| **Personnel Structurization** | **09/29/14** | **09/29/14** | **1** |
| **Obtaining Tools** | **10/01/14** | **10/01/14** | **1** |
| **Define Metrics** | **10/02/14** | **10/03/14** | **2** |
| **Acceptance of Project Organization** | **10/04/14** | **10/06/14** | **2** |
| **System Design** | **10/06/14** | **10/10/14** | **5** |
| **Create Use Cases** | **10/06/14** | **10/06/14** | **1** |
| **Subsystem Decomposition** | **10/07/14** | **10/07/14** | **1** |
| **Determine Security and Privacy** | **10/08/14** | **10/08/14** | **1** |
| **Create DB Schema** | **10/09/14** | **10/09/14** | **1** |
| **Hardware & Software Mapping** | **10/10/14** | **10/10/14** | **1** |
| **Object Design** | **10/12/14** | **10/24/14** | **11** |
| **Create Class Diagram** | **10/12/14** | **10/12/14** | **1** |
| **Create OCL statements** | **10/13/14** | **10/13/14** | **1** |
| **Create Sequence Diagram** | **10/24/14** | **10/24/14** | **1** |
| **System Implementation** | **10/10/14** | **11/13/14** | **25** |
| **Create/Configure Database** | **10/10/14** | **10/14/14** | **3** |
| **Implement User Interface** | **10/14/14** | **10/24/14** | **9** |
| **Implement System Logic** | **10/25/14** | **11/13/14** | **15** |
| **Present Deliverable** | **10/28/14** | **10/29/14** | **2** |
| **Testing** | **11/13/14** | **12/02/14** | **14** |
| **Create & Implement Test Cases** | **11/13/14** | **11/17/14** | **3** |
| **Run test suites** | **11/13/14** | **11/19/14** | **5** |
| **Evaluate & Fix Defects** | **11/19/14** | **12/02/14** | **10** |
| **Present Final Deliverable** | **12/15/14** | **12/16/14** | **2** |
| **Present Deliverable to Client** | **12/15/14** | **12/15/14** | **1** |
| **System Release** | **12/16/14** | **12/16/14** | **1** |

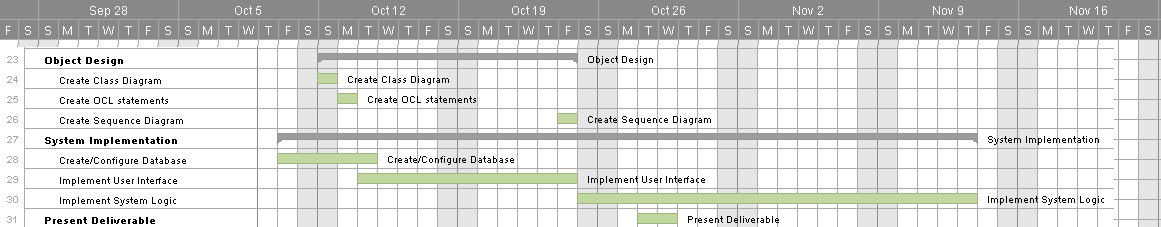
# **4. Appendix**

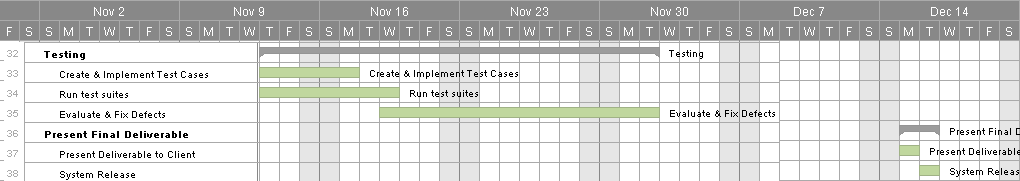
The appendix chapter will visually cover the project schedule, feasibility and cost matrix including tools and software costs of the Test Case Management System. A diary of meetings and tasks is specified on Appendix D followed by section 5 which lists the references utilized during the elaboration of the current document.

## **4.1. Appendix A - Project schedule (Gantt chart or PERT Chart)**









## **4.2. Appendix B – Feasibility Matrix**

|  |  |  |  |
| --- | --- | --- | --- |
| **Characteristics** | **Option 1**  **Web Application** | **Option 2**  **VS plug-in** | **Option 3 IBM ClearQuest** |
| **Technical** | Low Risk, Flexible, Future enhancement possible  *Score: 5* | High Risk, Inflexible, Personalized Enhancements not possible  *Score: 3* | Low Risk, Inflexible, Personalized Enhancements not possible, redundant data  *Score: 2* |
| **Operational** | No User Resistance  *Score: 5* | User Resistance  *Score: 2* | User Resistance  *Score: 3* |
| **Economic** | $31,560 | $35,560.00 | $20,160 |
| **Expiration Time** | Indefinite  Score: 5 | Dependent on Visual Studio Licensing  Score: 3 | 12 Months  *Score: 1* |
| **Estimated Configuration & Completion Time** | 3 to 4 Months  *Score: 3* | 2.5 Months  *Score: 4* | 2 Months  *Score: 5* |
| **Total Score** | **18/20** | **12/20** | **11/20** |

## **4.3. Appendix C – Cost Matrix**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Task Name** | **Work** | **Cost Per Person** | **Total Cost** |
| 0 | Planning | 20 hrs | $600.00 | $1,200.00 |
| 1 | Requirement Elicitation | 5 hrs | $150.00 | $300.00 |
| 2 | Meet with the client | 4 hrs | $120.00 | $240.00 |
| 3 | Determine Functional Requirements | 4 hrs | $140.00 | $280.00 |
| 4 | Determine Non-functional Requirements | 4 hrs | $140.00 | $280.00 |
| 5 | Create use cases | 24 hrs | $720.00 | $1,440.00 |
| 6 | System Analysis | 24 hrs | $720.00 | $1,440.00 |
| 7 | Determine system feasibility | 16 hrs | $480.00 | $960.00 |
| 8 | Compare to current system | 8 hrs | $240.00 | $480.00 |
| 9 | Present Deliverable | 1 hrs | $30.00 | $60.00 |
| 10 | Present deliverable to client | 1 hrs | $30.00 | $60.00 |
| 11 | System Design | 40 hrs | $1,200.00 | $2,400.00 |
| 12 | Object Design | 40 hrs | $1,200.00 | $2,400.00 |
| 13 | System Implementation | 301 hrs | $3,750.00 | $7,500.00 |
| 14 | Present Second Deliverable | 4 hrs | $120.00 | $240.00 |
| 15 | Present second deliverable to client | 4 hrs | $120.00 | $240.00 |
| 16 | Unit Testing | 40 hrs | $1,200.00 | $2,400.00 |
| 17 | Integration Testing | 80 hrs | $2,400.00 | $2,800.00 |
| 18 | Fix defects | 16 hrs | $450.00 | $900.00 |
| 19 | System Testing | 80 hrs | $2,400.00 | $4,800.00 |
| 20 | Fix defects | 16 hrs | $450.00 | $900.00 |
| 21 | Present Final Deliverable | 4 hrs | $120.00 | $240.00 |
| 22 | Present final deliverable to client | 4 hrs | $120.00 | $240.00 |
|  | **Total** | 620 hrs | $14,050.00 | **$31,560.00** |

Below are the tools that will be needed in order to successfully complete the project:

|  |  |  |
| --- | --- | --- |
| **Tools** | **Number of Units** | **Budget** |
| Server  Public IP  4GB Ram  100GB HD free space | 1 | $2,199.99 |
| GIT Client | 2 | $0.00 |
| Microsoft Office Home | 2 | $139.99 |
| Team Foundation Server 2013 | 1 | $499.00 |
| Visual Studio 2013 Premium or better | 2 | $2,569.00 |
| StarUML | 2 | $0.00 |
| Microsoft Test Manager | 2 | $0.00 |
| Microsoft Windows Server 2012 Dedicated Hosting (3 months) | 1 | $600.00 |
| Microsoft SQL Server Standard 2012 | 1 | $2,499.99 |
| **TOTAL** |  | **$11,417.94** |

## **4.4. Appendix D - Diary of Meetings**

**Diary Entry 1:**

**Date:** Friday, Jan 5, 2015

**Location:** Virtual(SKype)

**Start time**: 4:15 pm

**End time:** 5:00 pm

**In Attendance:** Dionny Santiago, Justin Phillips, Artiom Tiurin

**Late:** N/A

**Diary Entry 2:**

**Date:** Monday, Feb 02, 2015

**Location:** Virtual (Skype)

**Start time**: 11:00 am

**End time:** 11:30am

**In Attendance:** Karina Harfouche, Justin Phillips, Artiom Tiurin

**Late:** N/A

**Diary Entry 2:**

**Date:** Monday, Feb 02, 2015

**Location:** Virtual (Skype)

**Start time**: 2:45 pm

**End time:** 3:30pm

**In Attendance:** Tariq King, Dionny Santiago, Justin Phillips, Artiom Tiurin

**Late:** N/A

# **5.** **References**

1. COCOMO II:  
   <http://csse.usc.edu/tools/COCOMOII.php>
2. IBM Rational ClearQuest:  
   <http://www-03.ibm.com/software/products/en/clearquest>
3. Microsoft Test Manager  
   <http://msdn.microsoft.com/en-us/library/jj635157.aspx>
4. Microsoft Team Foundation Server  
   <http://msdn.microsoft.com/en-us/library/ms181238(v=vs.90).aspx>
5. Wikipedia: Software Testing  
   <http://en.wikipedia.org/wiki/Software_testing>

6. Test planning using IBM Rational Quality Manager  
 <http://www.ibm.com/developerworks/rational/library/09/0203_kelly/>