Statement of Work

and

Technical Document

Group 2

Robbie Dodenbier

Matthew LeFevre

Elizabeth Lim

Taylor Posivio

Darin Roberts

Brandon Sommer

Contents

[1. Business Problem 3](#_gjdgxs)

[2. Proposed Project](#_35nkun2) 5

[3. Risk Analysis 3](#_1fob9te)

[4. Discussion of Chosen SDLC 3](#_3znysh7)

[5. Team Composition 3](#_2et92p0)

[6. Quality & Assurance Standards and Measures 4](#_tyjcwt)

[7. Project Management Plan 4](#_3dy6vkm)

[8. Works Cited 4](#_1t3h5sf)

[Appendix A —Technical Problem Description 5](#_4d34og8)

[Appendix B: Project Management Detail 5](#_2s8eyo1)

[Appendix C: UML System Use Case 5](#_17dp8vu)

[Appendix D: UML Class Diagram of Proposed Solution 6](#_3rdcrjn)

[Appendix E: UML Sequence Diagram of Proposed Solution 6](#_26in1rg)

[Appendix F: UML State Diagram of Proposed Solution 6](#_lnxbz9)

# Business Problem

This is where you provide the business context and background as to why a systematic solution is required. Similar to the Economics lab, you want to identify the positive financial impact of this system. Is there a business need that this system addresses? Include that here.

Company:

* Software Development Department of a multinational retail corporation

Current problem:

System is not currently performing as it should

Requires connection to internal resources via office or VPN connection

Security Audit revealed that the servers don’t house/transmit the code in a safe manner

Cost of current system is: $blah

No version control on code base

Internal systems require highly skilled employees to maintain

Benefits of new system:

* Monetary benefits
  + Save on internal costs
  + Less people to maintain and upgrade current system
  + No fines for breaking the law
* Productivity benefits
  + Ease of use
    - Ability to share code within divisions/teams
    - Ability to branch master code to work on little sections of the code
    - Industry standard, so new employees can be caught up faster
  + No down time
  + Can be accessed anywhere in the world
  + Version control

EveryDay is a multinational retail corporation that has stores throughout the world. The last couple of years EveryDay has been trying to gain a larger online presence to accompany its physical locations. The Tech Departments has seen massive growth to allow this to happen. We have expanded from 5 software engineers to now over 20. We are expecting to see similar growth in the months/years ahead.

Because we were small when we started, we didn’t have a very established method of sharing code and working on it jointly. This led to our team not working very well together. We had one, possibly 2, members working on any given project at a time, and if we needed more, it started to become very cumbersome. We had a local server housing our data, but there wasn’t anything set up to check out the data and monitor for revisions. Whatever the latest version uploaded was, that was what you should work on. Of course, this led to many problems.

The IT Department is proposing that we set up a repository in order to design the code and work on projects more efficiently. There will be a large upfront cost. These costs will include new hardware to support servers and hold the data. It will also include a large investment in Software Engineering work hours. There will also be ongoing costs associated with this investment. Time must be allocated for maintenance of the servers. Time will also need to be spent on system upgrades that will need to be implemented as projects and requirements change.

One of the biggest reasons that we need to implement this new software is that it is going to be compliant with the new online security laws. Our company currently is having a hard time working with the Sarbanes-Oxley (SOX) Act of 2002. This law says that EveryDay must keep a record of all electronic communications for a certain length of time. This will allow us to comply more fully with this law and avoid the possibility of future monetary damages, as well as damages to the reputation to EveryDay.

However, there is not just going to be an outflow of money. This are going to reap huge benefits within EveryDay. After implementation, we expect Software Engineers to be working above a 25% increase in productivity. We will be able to get more engineers to work on a project at a time, which will reduce the length of time for projects, especially those that have been designated as a high priority for upper management. This will reduce the overall costs of projects by 25%.

We expect that, in addition to becoming more efficient, we are going to be able to deploy this system over a Virtual Private Network (VPN). This VPN will allow our engineers to work from anywhere across the world. We can use some engineers from areas that have a lower wage, saving money on salaries. We can also hire engineers on a part time, short term, or project basis. If there is a project that we need to complete in a short time span, it will much simpler hiring engineers to work on a specific project.

The system we have designed is simple and easy to understand. It is going to be a relatively easy system for engineers to understand so they can start adding and modifying code without much instruction. This is going to drastically change our current system that takes new engineers doing ride-alongs with more experienced engineers before they can really help out the team. Bringing on new engineers mid project has never really helped, but with the new system they should be able to contribute in a short time span.

The new system will also allow us to take branches of the main code and work on the branches without affecting the greater whole. This has always been a problem with our current system. If one person is working on the code, even if it is a small part, then it locks up the rest of the code for everyone else. With the new system, multiple people can write on the code at any given time.

Along with using branches of code, the new system is going to be able to do versions. Just because I upload a new version, it doesn’t mean we can’t go back to an earlier time with the code. The versions will be tracked with who has written the code. This will allow us to more easily find out who implemented a part of the code and we can backtrack any problems or improvements that were made and analyze them more fully.

While we do expect there to be times when the service might be down due to unforeseen circumstances, we to expect an uptime of greater than 99%. Currently, our system is experiencing an uptime of just about 90%. This does hinder our engineers in completing their tasks on time. It is possible to download and work with the code offline, but collaborating with the team is greatly impacted when the system is offline. The new system will help alleviate this burden.

# Proposed Project

Scope explains what your project/system will cover. For example, for complex systems, some project teams choose to break the delivery of those systems in chunks and only the specified chunk is what is addressed in the Proposed Project section.

**Terminology:**

* Branch - A branch is a complete or alternate version of the code base in a project. The main/master branch of the project is the priority branch of which most other branches are merged into once declared stable.
* Commit - A commit could be the creation of a new file/directory, an update to a current file/directory, or the removal or deletion of a file/directory to the codebase of a particular branch of a project.
* [EveryDay Software Solution] - reference to the version control system for which the project proposal is created.

**Feature/scope outline:**

* Version control
* VPN
* Intuitive Graphical User Interface
* Analytics

**General Purpose:**

[EveryDay Software solution] will improve how engineers collaborate as well as how internal software projects are managed and created through a series of modern features.

**Version Control:**

The [EveryDay Software solution] will provide backups and rollback features on three distinct levels of software projects. **\*This will require an immense and secure storage server.\***

* System - Backed up daily
* Project - Every 30 minutes a backup is created or updated on recently modified projects. These granular 30 minute updates will make it very easy to pinpoint the exact rollback time we want.
* File - Following a git style of version control changes made to project files must be committed and pushed to a project branch. Each commit is stored and recorded with the potential to roll back.
  + Commit messages: Each commit has an optional message parameter that allows software engineers to explain their commited code

Code is written on company computers and can be used to write code in an offline environment. Once connected to the internet code updates can be committed and pushed to a project branch.

**VPN:**

To convenience engineers a corporate VPN will be provided to access the [EveryDay Software solution] allowing for work to be conducted on public networks in any location with connectivity to the world wide web.

**Intuitive Graphical User Interface:**

Simple intuitive interface that allows developers to collaborate over code.

1. Project chat channels- Each project will have its own chat channel for use of developers
2. Project Documentation/Readme- Each project created will have code documentation locations
3. File/Project Commit Log - per file view the commit log
4. Easily copy or download entire projects to your local machine through the project GUI
5. GUI - The [EveryDay Software solution] client should have a built in text editor allowing engineers to commit small changes directly on the server without downloading the project to their local machine.
6. GUI - Pair programming in live collaboration mode. Use the Built in text editor to develop software with other engineers to bring new engineers up to speed quickly. Help engineers collaborate on best practices.

**Analytics/Team Collaboration:**

Individual engineers will be registered and metrics will be taken on activity.

**Analytical Metrics Recorded Per Engineer:**

* Contributing projects
* Commits per project
* Lines of code contributed per project
* Documentation written
* Favorite ice cream
* Birthdate
* Name
* SSN

# Risk Analysis

With any project, problems may arise. The risk section identifies the likely areas that the project team should watch out for and the approach the team should take to avoid/absorb the potential problem.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Risk Description | Consequence/Impact | Probability Score | Impact Score | Action/Mitigation |
| 1 | Insecure Connection | Unauthorized access could provide hackers with the inner works of the system and code, possibile compromise of data. | Medium | High | Insure connection between servers is done internally and via HTTPS connections. |
| 2 | Scope Creep | Additional development request continue to come in after the initial build has been released. | Medium | Medium | Judge the scope of the request, any large or altering change request will be pushed out to later build cycles. |
| 3 | Not Enough Good Code Writers | Poor developers will lengthen a timeline and possibly cause multiple bugs and delayed release. | Medium | High | Formulate development teams with mixed skill levels and capabilities. |
| 4 | Inaccurate Timeline | Delayed release and increase in project budget. | Small | Medium | Break up deliverable into manageable pieces and evaluate timeline after each deliverable. |
| 5 | Poor Project Management | Project team is not kept on task and project timelines will be missed. | Small | Medium | Weekly status calls with client to report on status, keeping team accountable. |
| 6 | Faulty DB Schema | Inability to report on data accurately | Small | High | Document system as you go and work with QA to properly test |
| 7 | Overly Aggressive Deliverable Date | Inability to hit promised deliverable date. Possibility of losing resources to other projects. | Small | Medium | Continue to monitor project status on a weekly basis and make adjustments when needed. |

# Discussion of Chosen SDLC

This section explains the delivery approach and schedule. Will your project be delivered with Agile? Waterfall? Do you need a few months or a year to implement the system? What types of checkpoints (milestones) will you need?

Since our current system for maintaining and sharing code is very ad hoc and unofficial, any other system that we can get implemented will benefit the company greatly. This is why we have decided to use the Agile method to deliver this system. The plan is to create a working system as soon as possible and then use that system to finish the project. We will aim to have the first release of the project within 2 months with further iterations made every two weeks to add additional features. Each two week cycle will consist of two one week parts. The first week will be implementing the new features for that release and the second week will be reserved for QA to find and test for bugs in preparation for the release. Stand up meetings will be conducted every Wednesday with the entire team to bring up major issues and roadblocks in the development process.

The release plan is as follows:

Release 1.0: Release of basic code repository software (Commit/Merge/Branching features) (2 Months)

Release 1.1: Implement cross developer communication system (2 Weeks)

Release 1.2: Implement automatic backup feature (2 Weeks)

Release 1.3: Implement https communication/database encryption (2 Weeks)

Release 1.4: Add in-client VPN integration (2 Weeks)

Total Time to reach currently outlined features: 4 Months:

As we receive feedback from the developers who are working on and using the system, more releases with additional features will be planned.

# Team Composition

Do you need any particular expertise to successfully complete this project? This section identifies all those qualifications.

3 Project Managers - Their role is to guide and direct the team in making sure that meet their goals with the project.

3 Network Engineer - Their role is to make sure that the network infrastructure is sufficient to handle all of the traffic that will be generated with the new project.

9 Software Engineers - Their roles is to build and compile the code to make this new project possible for the company.

QA Team - This team will be comprised of all the project managers as well as a member from the software and network engineer teams. Their job is to test the system and make sure that it is ready for deployment of if it needs improvement.

Penetration Testing Firm - Their job will be to test the security of our infrastructure and project to make sure it meets our security needs.

Business Analyst - There job is to make sure that the need of the business are met and to make sure that the project remains an economic asset to the company.

# Quality & Assurance Standards and Measures

There are various standards for quality. Select one you think applies and how you will measure how those standards are met.

ISO 9000-3 - http://www.mhhe.com/engcs/compsci/pressman/information/olc/ISO9000.html

Configuration Management

Change Control

Development Planning

Quality Planning

Design and Implementation

Testing and Validation

Maintenance

# Project Management Plan

This section would seem redundant since a good plan already covers the previous sections. That said, there are other components that exist in a project management plan that you can include here as well (that weren’t covered previously).  
This would be a good area to summarize the overall plan and the critical components that need to be in place in order to move forward.

How Group is structured

Group Meetings

Infrastructure

Development

QA

Deployment

Maintenance

# Works Cited

Reference section. Post URLs as needed.

<http://www.mhhe.com/engcs/compsci/pressman/information/olc/ISO9000.html>

<http://proquestcombo.safaribooksonline.com.byui.idm.oclc.org/book/software-engineering-and-development/uml/9781118037423>

https://linchpinseo.com/the-agile-method/

# Appendix A —Technical Problem Description

This is where you’ll have your use case and/or user stories and share how the user(s) are expected to interact with the system so they can accomplish their requirements.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | |
| **Use Case ID / Link:** |  | **Use Case Name:** | Insert Use Case Name |
| **Description** | |  | |
| **Primary Actor or Persona** | |  | |
| **Pre-Conditions** | | *Identifies pre-conditions that are required to execute the use case* | |
| **Post-Conditions** | | *Aka - Acceptance Criteria (part of narrative approach) are represented here.* | |
| **Triggers** | | *A trigger is the initiator of a use case. It is what causes the use case to start. There isn’t a promise that this event happens – only an indication that this event triggers the start of a use case.* | |
| **Main/Happy Path** | | *Enter the main flow of events or steps narrating interactions between processes/personas. Should always end with a success end state.* | |
| **Alternative / Exception Flows** | | *Any alternate/exception flow of events are captured here.* | |
| **Use Case Frequency** | | *Helps inform scalability. E.g. 50 per hour, 20 per day)* | |
| **Non-functional Requirements** | | *[optional]To capture UI related, Accessibility, Response time, localization, some business rules that were not covered as part of the steps in the path or pre/post conditions above.* | |

Use Case:

* Version Control
* Code Submission
* Comments

# Appendix B: Project Management Detail

Includes GANNT Chart with estimate of time, resources, hours of labor, and assigned cost.

You’ll want to tie this to the primary information listed in Sections 4-7. For example, if you need developers and testers, you’ll need to determine if they can accomplish the work in the pre-defined timeline or will the timeline need to be expanded based on the amount of work.

For example: Management OKs that the development phase is 5 weeks long. You estimate that the work will require 400 dev hours. Can you complete it with 1 developer in those 5 weeks? How many more resources might actually be needed (assuming per person works 40 hours a week)? GANTT charts are a good way to show that you are not planning to overburden your resources. MS Project is the best way to create GANTT charts, but can also be created in Excel. You can download MS Project from the On The Hub site using your BYUI credentials.

# Appendix C: UML System Use Case

If your diagram does not fit in this space, shrink it and then reference the file that has the full version.

For more detail, please refer to <usecase.pdf>

# Appendix D: UML Class Diagram of Proposed Solution

If your diagram does not fit in this space, shrink it and then reference the file that has the full version.

For more detail, please refer to <classdiagram.pdf>

# Appendix E: UML Sequence Diagram of Proposed Solution

If your diagram does not fit in this space, shrink it and then reference the file that has the full version.

For more detail, please refer to <sequence.pdf>

# Appendix F: UML State Diagram of Proposed Solution

If your diagram does not fit in this space, shrink it and then reference the file that has the full version.

For more detail, please refer to <state.pdf>