Statement of Work

and

Technical Document

Group 2

{{ Redacted }}

{{ Redacted }}

{{ Redacted }}

{{ Redacted }}

Darin Roberts

{{ Redacted }}

Contents

[**Business Problem**](#_gjdgxs) **3**

[**Proposed Project**](#_gdsbc96dsiqp) **4**

[**Risk Analysis**](#_1fob9te) **5**

[**Discussion of Chosen SDLC**](#_3znysh7) **6**

[**Team Composition**](#_2et92p0) **6**

[**Quality & Assurance Standards and Measures**](#_tyjcwt) **7**

[**Project Management Plan**](#_3dy6vkm) **8**

[**Works Cited**](#_1t3h5sf) **8**

[**Appendix A —Technical Problem Description**](#_45ywzjvxbdqr) **9**

[**Appendix B: Project Management Detail 7**](#_2s8eyo1) **13**

[**Appendix C: UML System Use Case**](#_17dp8vu) **13**

[**Appendix D: UML Class Diagram of Proposed Solution**](#_3rdcrjn) **14**

[**Appendix E: UML Sequence Diagram of Proposed Solution**](#_26in1rg) **15**

[**Appendix F: UML State Diagram of Proposed Solution**](#_lnxbz9) **22**

# Business Problem

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.

Due to our rapid growth, we have exceeded the capabilities of our current infrastructure and processes. We need to have a scalable solution that will allow our team members to work jointly on projects, from across multiple regions of the globe. Team members should be able to download the code concurrently and work on it simultaneously. This type of infrastructure will not only lead to a more stable environment, but will be also be more efficient.

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 not completely compliant 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.

Even though there will be an upfront cost, there is not just going to be an outflow of money. EveryDay will to reap huge benefits by implementing this system. 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%.

To fulfill the needs of the IT Department, we are 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.

Below are just some of the advantages that our new system will have over the old 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

# 

# Proposed Project

**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
* 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. The features that will be prioritized in the system include a new graphical user interface, accessibility anywhere in the world through VPN, and most importantly the safe and secure storage of code artifacts created by engineers through git “like” version control.

**Version Control:**

The [EveryDay Software solution] will provide backups and rollback features on three distinct levels of software projects.

* 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.

**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
* Name

# 

# Risk Analysis

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Risk Description | Consequence/Impact | Probability Score | Impact Score | Action/Mitigation |
| 1 | Unsecure Connection | Unauthorized access could provide hackers with the inner workings of the system and code, possibile compromise of data. | Medium | High | Ensure connection between servers is done internally and via HTTPS connections. Implement firewalls to prevent outside access. |
| 2 | Scope Creep | Additional development request continue to come in after the initial build has been released. | Medium | Medium | Project Manager shall judge the scope of the request, any large or altering change request will be pushed out to later build cycles. |
| 3 | Inadequate staff available from external stakeholders until later in build cycle. | Inability to provide pre-release Alpha builds. | High | Low | Work with external stakeholders to free up resources as quickly as possible. Initial development team will focus on framework and design. Heavy coding will commence as team grows. |

# Discussion of Chosen SDLC

To implement this project and improve efficiency of future projects, we will employ the Agile method. The plan is to create a system that developers can use to finish the project in the first month, and then use the next to month to implement non-core features.

We will aim to have the first release of the project after two weeks with further iterations made every two weeks after that 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, etc.) (2 Weeks)

**Release 1.1:** Implement User Interface and cross developer communication system (2 Weeks)

**Release 1.2:** Implement automatic backup feature, https communication and database encryption (2 Weeks)

Total Time to reach currently outlined features: **6 Weeks**.

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

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

**2 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.

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

**2 Quality Assurance Testers** - 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 Testers** -Hire outside company to review and test the application throughout the development of the application.

**1 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.

With these team members we will form two separate teams to work on features concurrently consisting of 1 Network Engineer, 3 Software Engineers and 1 QA Tester.

# 

# Quality & Assurance Standards and Measures

Quality assurance will be used all throughout the software life cycles until the release of the software product. The following are the quality assurance tasks for each phase of the software development:

**Requirements phase:** When the software is being developed, the developer needs to ensure that it elucidates the proposed functionality of the product and to keep refining until the requirements are clearly stated and understood.  
  
**Specification and Design phase:** Because is the importance for accuracy and completeness in this phase, the developers and managers will hold weekly developmental meetings. These reviews are to test and identify defects, bugs, or changes in specifics or design and correct those are necessary.  
  
**Implementation phase:** Throughout the development and implementation phase the developer team will continue to review and tidy up code. Various implementations of resources and tools to simplify and made the code more accessible, secure and robust.   
  
**Software testing phase:** The developer shall test each unit. The final software solution shall be verified with the functionality of the software as specified in the Proposed Project section of this document. Throughout these phases of the software development, the following shall also be performed to improve the software quality:

* **Communication and Feedback:** The developers are encouraged to freely express disagreements, suggestions and opinions about all aspects of the weekly process of software development.
* **Internal audits and evaluations:** Those listed in the Team Composition section of this document will perform evaluations to ensure that the software is functioning as proposed in the project plan and that functionality, regulations and requirements are met before the release of the software.

# 

# Project Management Plan

This project will be worked on by 2 separate groups. Each group will have different assignments to complete. The groups are designed to be as small as possible to make the work easier to coordinate.

The groups will be working part time on this project and part time on their other scheduled projects as well. This way we don’t have to completely give up work on their regularly scheduled assignments for this project.

In order to access the databases externally, we will create a VPN client that will allow the users to access the servers. The VPN client will have two-factor authentication to help protect the internal infrastructure from malicious intruders. We understand that having a VPN does open the door to outsiders, but this is a necessary risk that we are willing to take. We will be monitoring the connection to ensure that only those who are authorized are able to enter.

Each team will have a QA specialist assigned to them. This QA specialist will work with developers and engineers to make sure that the project is as complete and functional as possible. Before the projects are deployed, every possible measure will be taken to ensure the project works as needed and desired.

After the project is complete, we do expect there to be maintenance that needs to be done. However, we project that the maintenance on the upgraded project will actually be less than is currently needed with the system we have now. This is going to reduce running costs further for EveryDay.

# 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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | |
| **Use Case ID / Link:** | 1230 | **Use Case Name:** | New Project |
| **Description** | | *In order to get started a new project must be made. This will enable users to continue to creating projects for EveryDay, not just work on one project at a time.* | |
| **Primary Actor or Persona** | | *Developer* | |
| **Pre-Conditions** | | *The user must have access to the database and have the program installed on their local machine. This local machine will be able to talk to the database and create the project space.* | |
| **Post-Conditions** | | 1. *Upon starting a project, space will be created on both the local machine and the database.* 2. *User names the project* 3. *The Project will be opened on the local machine.* | |
| **Triggers** | | *Pushing the “New Project” button.* | |
| **Main/Happy Path** | | 1. *User starts new project* 2. *User gives name for the project* 3. *Project created on Database* 4. *Project created on local machine* 5. *Project is auto-saved* | |
| **Alternative / Exception Flows** | | *User cancels prior to creation, either manually or network interruptions. Project is not saved to database and system returned to prior state.* | |
| **Use Case Frequency** | | *~ 1 per day* | |
| **Non-functional Requirements** | | *None* | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | |
| **Use Case ID / Link:** | 1234 | **Use Case Name:** | Commit |
| **Description** | | *The core functionality of the EveryDay solutions platform is the ability to have multiple developers share one master code base. Core to that is the ability to commit (save) a local copy of the code before pushing back to the master code base any changes.* | |
| **Primary Actor or Persona** | | *Developer* | |
| **Pre-Conditions** | | 1. *Code base has been pulled down to local machine* 2. *Modifications have been made to the code* | |
| **Post-Conditions** | | 1. *On successful completion, code will be saved to local repository.* 2. *Commit button will be grayed out* | |
| **Triggers** | | *None* | |
| **Main/Happy Path** | | 1. *Developer will launch main program* 2. *Developer will pull latest master code* 3. *Developer will modify code* 4. *Program will identify a change was made and enable the Commit button* 5. *Developer will Commit button* 6. *Program will launch Commit dialog box* 7. *Developer will enter in a comment* 8. *Developer will click submit button* 9. *Program will commit code changes to local repository* 10. *Program will grey out (disable) Commit button* | |
| **Alternative / Exception Flows** | | 1. *No other flow exists* | |
| **Use Case Frequency** | | 1. *No Use Case frequency is defined.* | |
| **Non-functional Requirements** | | 1. *Execute latest version of the program* | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | |
| **Use Case ID / Link:** | 1235 | **Use Case Name:** | Comment |
| **Description** | | *The core functionality of the EveryDay solutions platform is the ability to have multiple developers share one master code base. Core to that is the ability to commit (save) a local copy of the code before pushing back to the master code base any changes. Allowing commit comments allows the developer to explain their changes.* | |
| **Primary Actor or Persona** | | *Developer* | |
| **Pre-Conditions** | | 1. *Code base has been pulled down to local machine* 2. *Modifications have been made to the code* | |
| **Post-Conditions** | | 1. *On successful completion, code will be saved to local repository.* 2. *Commit button will be grayed out* | |
| **Triggers** | | *None* | |
| **Main/Happy Path** | | 1. *Developer will launch main program* 2. *Developer will pull latest master code* 3. *Developer will modify code* 4. *Program will identify a change was made and enable the Commit button* 5. *Developer will Commit button* 6. *Program will launch Commit dialog box* 7. *Developer will enter in a comment* 8. *Developer will click submit button* 9. *Program will commit code changes to local repository* 10. *Program will grey out (disable) Commit button* | |
| **Alternative / Exception Flows** | | 1. *No other flow exists* | |
| **Use Case Frequency** | | 1. *No Use Case frequency is defined.* | |
| **Non-functional Requirements** | | 1. *Execute latest version of the program* | |

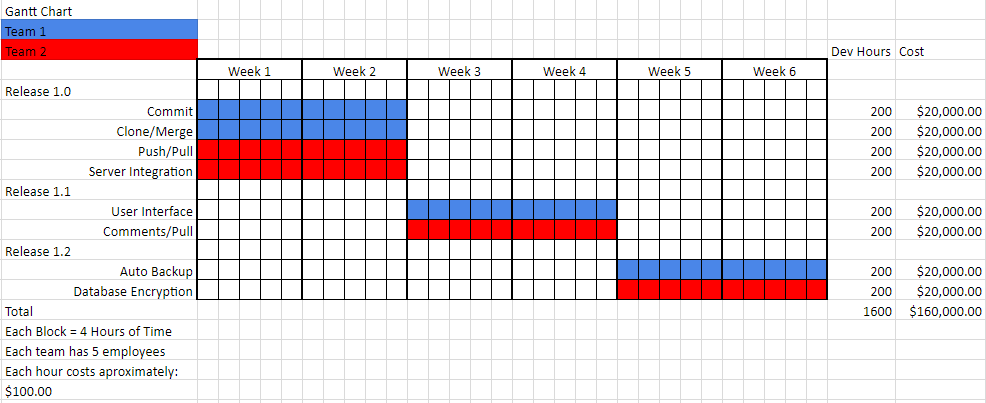
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | |
| **Use Case ID / Link:** | 1236 | **Use Case Name:** | Clone |
| **Description** | | *The core functionality of the EveryDay solutions platform is the ability to have multiple developers share one master code base. Core to that is the ability to create a local copy of the code(clone) code that has already been written by a developer.* | |
| **Primary Actor or Persona** | | *Developer* | |
| **Pre-Conditions** | | 1. *Navigate to the main page of the repository* 2. *Change the current working directory to the location where you want the cloned directory to be made.* | |
| **Post-Conditions** | | 1. *On successful completion, code will be saved to local repository.* | |
| **Triggers** | | *None* | |
| **Main/Happy Path** | | 1. *Developer will navigate to the main page of the repository* 2. *Developer will click clone or download under the repository name* 3. *Developer will change the current working directory to the location where you want the cloned directory to be made* 4. *Developer will type clone and then paste the URL* 5. *Developer will press Enter. Your local clone will be created* | |
| **Alternative / Exception Flows** | | 1. *No other flow exists* | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | |
| **Use Case ID / Link:** | 1237 | **Use Case Name:** | Merge |
| **Description** | | *One function of the EveryDay solutions platform is the ability to merge changes in one branch of code to the main branch. The developer will be able to clone code from the main repository, work on it and save it locally and then merge the changes back to the main repository when the feature they are working on is ready.* | |
| **Primary Actor or Persona** | | *Developer* | |
| **Pre-Conditions** | | * *The user must have a branch of code developed separately* | |
| **Post-Conditions** | | * *If successfully merged the branch of code merged will become the new master branch* | |
| **Triggers** | | *The user clicks the “Merge” button* | |
| **Main/Happy Path** | | 1. *The user clones an existing project (see “Clone” use case).* 2. *The user makes changes to the project* 3. *These changes are committed and saved locally (see “Commit” use case)* 4. *The user clicks “Merge” and selects the branch and the main code base being merged to.* 5. *The user resolves conflicts manually between the branch and original code* 6. *The user clicks “Done” to finish resolving conflicts* 7. *These changes become part of the main project* | |
| **Alternative / Exception Flows** | | 1. *The user clones an existing project (see “Clone” use case).* 2. *The user makes changes to the project* 3. *These changes are committed and saved locally (see “Commit” use case)* 4. *The user clicks “Merge” and selects the branch and the main code base being merged to.* 5. *The user decides the code is not ready to be merged due to conflicts.* 6. *The user clicks “Cancel” to cancel the merge.* 7. *These changes don’t become part of the main project* | |
| **Use Case Frequency** | | *Everytime a new feature needs to be worked on and added to the main code base* | |
| **Non-functional Requirements** | | *The user will need to have menus and pop ups that prompt them to select the folder containing the code to be merged and the project that it is being merged into,* | |

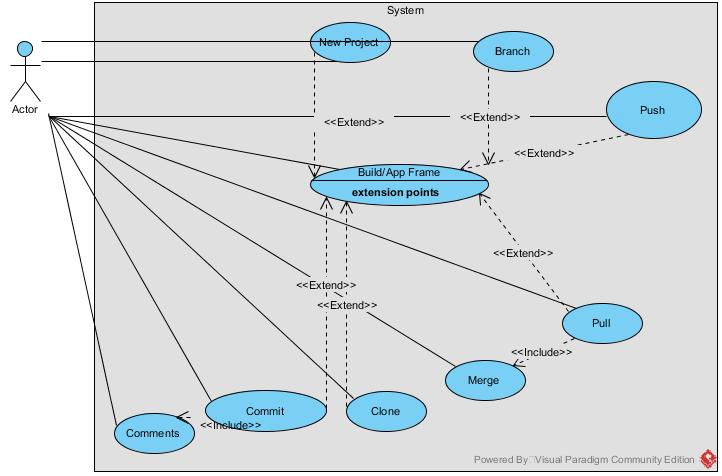
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | |
| **Use Case ID / Link:** | 1238 | **Use Case Name:** | Push |
| **Description** | | *Once code has been committed the Everyday software solution comes with the function to push code to a branch* | |
| **Primary Actor or Persona** | | *Developer* | |
| **Pre-Conditions** | | 1. *Code has been committed(saved)* | |
| **Post-Conditions** | | 1. *On successful completion, code will be saved to remote secure server.* 2. *push button will be grayed out* | |
| **Triggers** | | *None* | |
| **Main/Happy Path** | | 1. *Developer will have made changes and staged all commits necessary for a push.* 2. *Developer will select a branch* 3. *Developer will click the push button* 4. *All staged commits will be added to the selected branch* | |
| **Alternative / Exception Flows** | | 1. *No other flow exists* | |
| **Use Case Frequency** | | 1. *No Use Case frequency is defined.* | |
| **Non-functional Requirements** | | 1. *Execute latest version of the program* | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | |
| **Use Case ID / Link:** | 1239 | **Use Case Name:** | Pull |
| **Description** | | *Contacts repository and pulls down the requested information.* | |
| **Primary Actor or Persona** | | *Developer* | |
| **Pre-Conditions** | | *Repository itself must be made with a master branch which can be pulled down.* | |
| **Post-Conditions** | | *If completed successfully new information on the master branch will be pulled down to the local branch.* | |
| **Triggers** | | *User clicks the “Pull” button* | |
| **Main/Happy Path** | | 1. *User pulls down the master branch from the repository.* 2. *User creates a local branch* 3. *Changes are finished and the code is pushed back to the repository. (See “Push” use case).* 4. *Users will then pull to get the new changes updated on their local system.* 5. *Press “Pull” button and local branch is updated with the latest information.* | |
| **Alternative / Exception Flows** | | *No other flow exists* | |
| **Use Case Frequency** | | *Anytime new information has been pushed to the repository* | |
| **Non-functional Requirements** | | *User will need to have a user interface with buttons to select the “Pull” button or they will need a terminal where they can type “Pull”.* | |

# Appendix B: Project Management Detail 7



# Appendix C: UML System Use Case

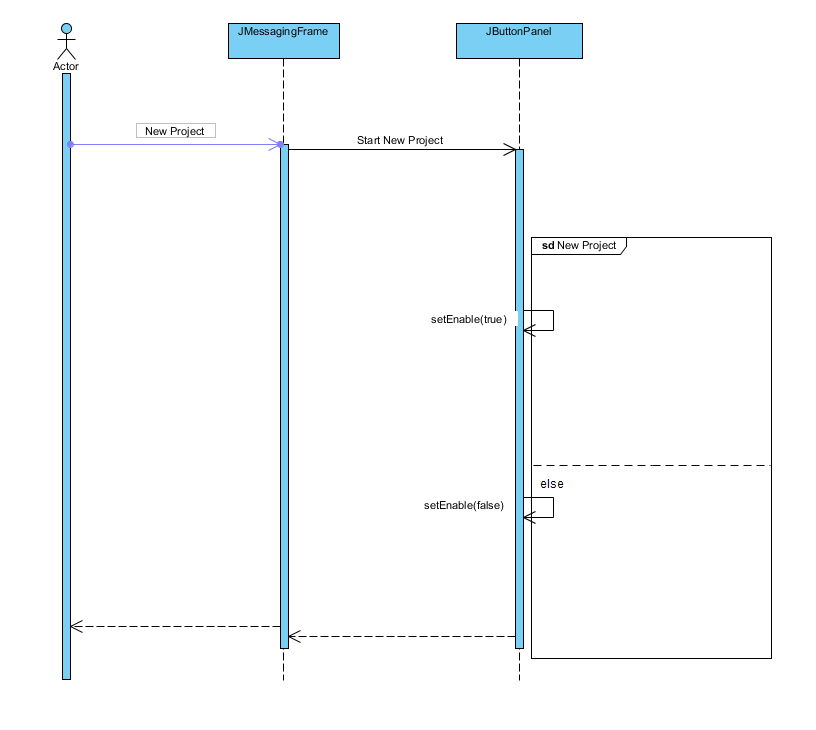
****

# Appendix D: UML Class Diagram of Proposed Solution

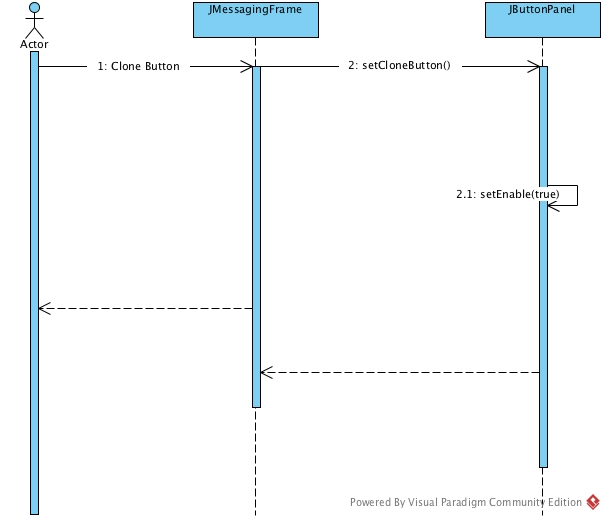
# 

# Appendix E: UML Sequence Diagram of Proposed Solution

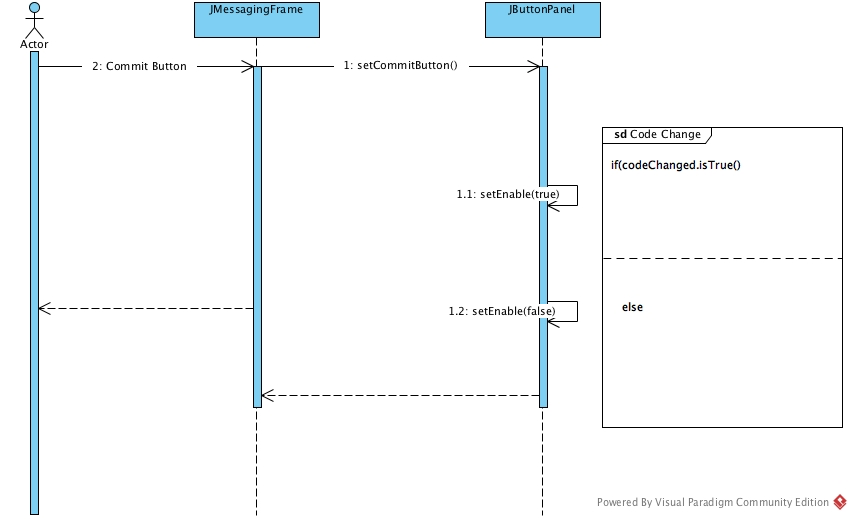
New Project



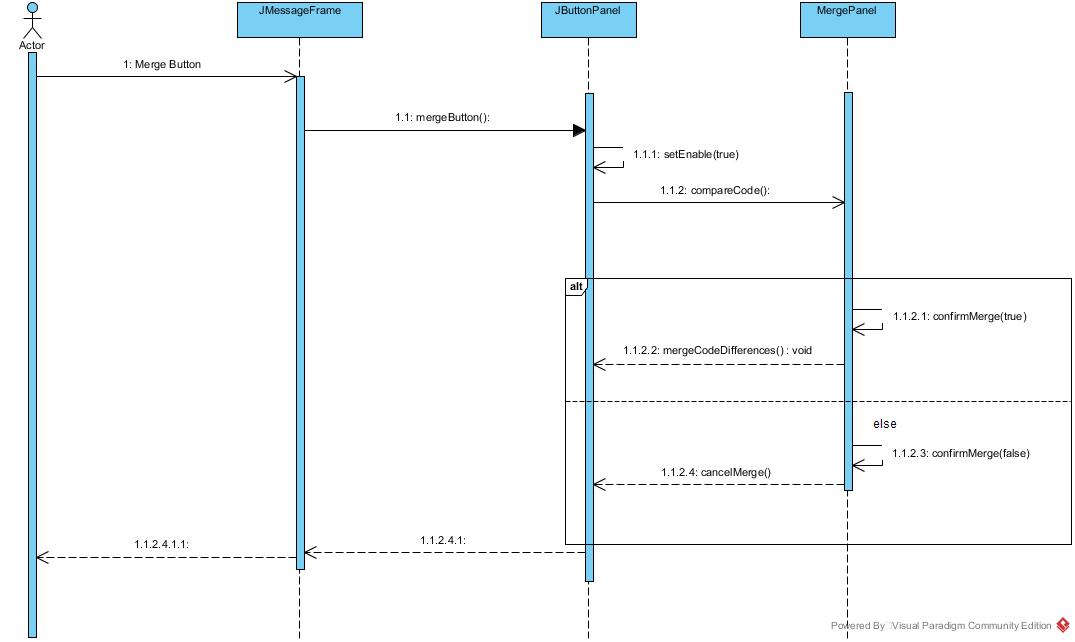
Clone Button



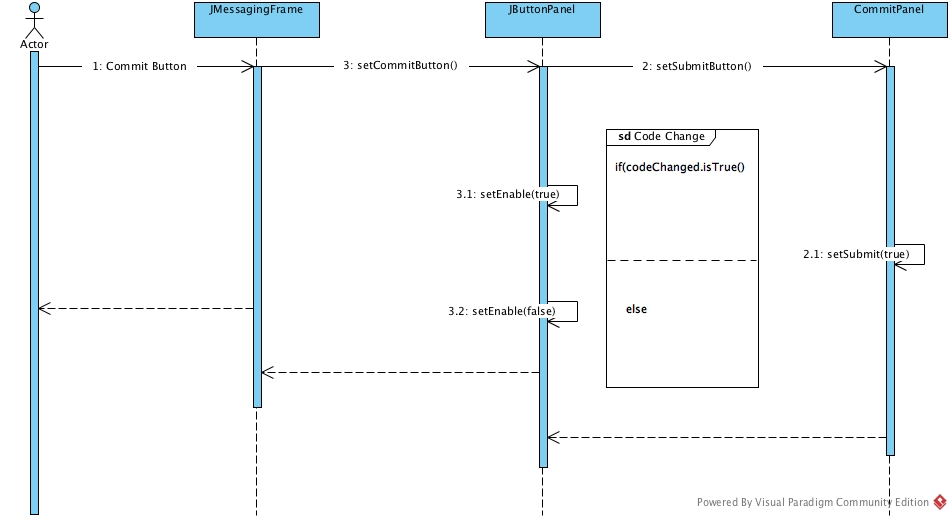
Commit Button



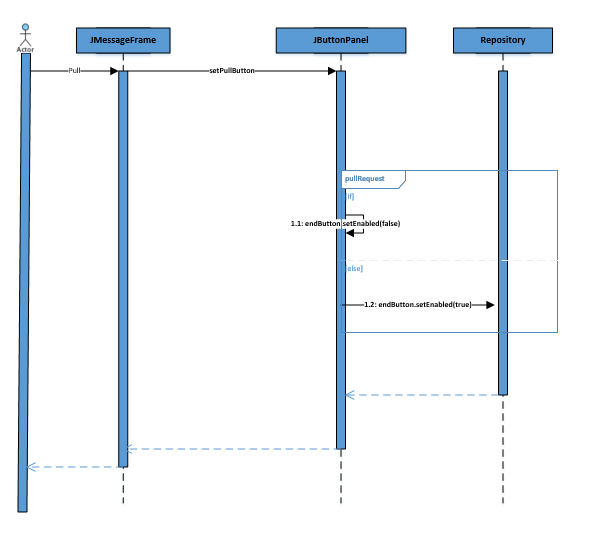
Merge Button



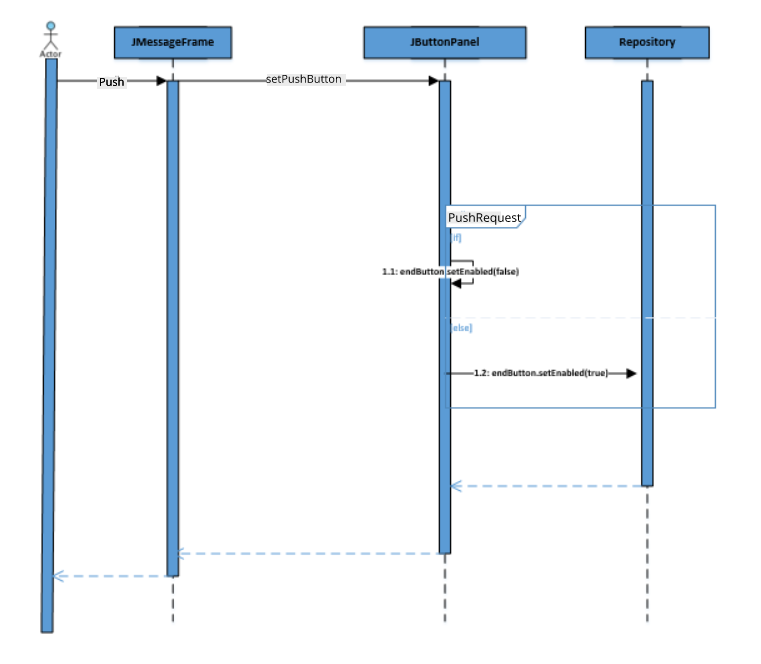
Comment Button



Pull Request Sequence Diagram



Push Request Diagram



# Appendix F: UML State Diagram of Proposed Solution

