|  |
| --- |
|  |
| Team Feynman |
| Team members: Cindy Samano, Alex Pope, Durwin Johnson, Erik Johnson Wenhao Zhang, Reggie Jirigesu |

|  |
| --- |
| 2-24-2017 |

***SOFTWARE PROJECT MANAGEMENT PLAN (SPMP)***

**1.0 Introduction**

* 1. **Problem statement**

The task at hand will be to develop a pharmacy system that will effectively manage a pharmacy store medicine inventory. It will allow for interaction and communication between store(s), company and warehouse. The system will use a comprehensive approach to minimize the effort between each interaction and ultimately allow the pharmacy store to deliver medicine to their customers.

The system will create a two-step application for managing and updating a pharmacy store’s medicine inventory. Maintaining inventories will be done at the warehouse, store and company level through a daily on-line system and a nightly batch system. The on-line system will interact with the pharmacy employees to perform transactions, check inventory and order items from company warehouse. The nightly batch system will be used to update the inventory of the company warehouse and stores based on the orders generated during the on-line system.

**1.2 Project scope**

**1.2.1 Inclusions**

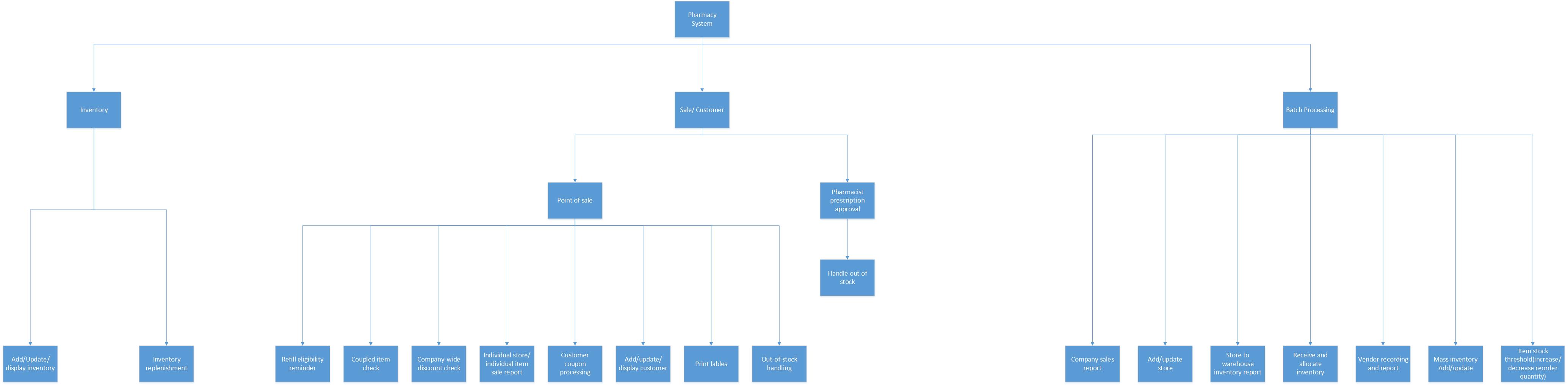
The software will act as inventory management for a specific pharmacy store or the entire pharmaceutical company. It will allow for the displaying and editing of items, inventory data and values. It will send requests to the company warehouse once per day to ensure that each store has consistent inventory values based on defined levels. If an item is consistently out of stock, the system will automatically order more than the previous amount. If the item is never out of stock, the system will order less of that item. In both cases the change will be reported. Stores will order the number of items needed to keep items in stock as frequently as possible. It will create specific historical reports for both individual stores and the company. Types of report periods will by specified by user. New items can be added and deleted from the inventory through the system. Additionally, customer transactions will be handled through a screen interface that communicates information to the company. This includes printing prescription labels, handling out of stock items, and account management. Additionally, there will be an automatic notification to prompt the cashier when a customer is checking out, if the customer is eligible for a refill the cashier can ask the customer if they would like a refill on their prescription. There will also be options for coupons for customers that provide discounts and companywide sales on certain items. Certain items will also automatically be offered to the customer at checkout when a similar item is purchased.

**1.2.2 Exclusions**

For this program, we have opted not to use a database system. We Believe our teams skill set is insufficient for the use of a database. As an alternative, we have decided to use a flat-file system as the team is much more familiar with these. We also have decided to keep the program as a single unit instead of using a client or service model.

**1.3 Major software functions**





**1.4 Performance/Behavior constraints**

Overview

This section describes the performance and behavior constraints imposed on the project by the customer's requirements, reasonable expectations of functionality and performance, and by the inherent nature of the project.

Constraints

* + Customer Constraints
    - Customer requirements will be determined during Phase 2, by completing Use Case Specifications and the Use Case Summary.
  + Reasonable Functionality and Performance Constraints
    - A query initiated online by the user should take no longer than 3 seconds to return a result and display it.
    - Offline batch processing should take no more than 3 minutes to complete.
    - If an unexpected error is encountered by the program:
      * The error and pertinent information should be output to the program log file.
      * An error message should be displayed to indicate to the user that an error occurred. The error should contain some description of the error, and an error code.
      * All possible steps should be taken to recover from the error. If recovery is impossible, all transaction steps taken in the unit of work that contains the procedure and specific parameters which incurred the error should be rolled back.
        + An unexpected error is defined as an error that occurs in lieu of an expected action taking place successfully. Example: During the batch process, the file containing the incoming batch items is not formatted properly. The software would both log an error and display a message indicating that the file was unreadable, specifying which file it was, and providing a specific error code.
    - If an expected error is encountered by the program:
      * An error message should be displayed to indicate to the user that an error occurred. The error should contain some description of the error, and an error code.
        + An expected error is defined as an error that occurs because of normal program functionality. Example: A customer tries to place an order but there is no inventory remaining. The user (cashier) would receive an error message and be told that the store is out of stock and prompted to initiate a transfer or stock replenishment for the customer, and would be provided with an ETA for the item to be available for pickup.

**1.5 Management and technical constraints**

**1.5.1 Management constraints**

This project is operating under the constraint of a drop dead delivery date. Therefore, the due date of this project is final. The absolute last day this project must be completed by has been set as 4/28/17. Because all the team members are students, there is minimal overlap in our schedules for meetings and discussion. However, after looking at all our general availabilities, the team has agreed to have a weekly meeting every Sunday evening. An updated availability schedule and calendar will be used to pick meeting times outside of the weekly Sunday meeting. After the first phase of the SPMP process, when submitting and editing final versions of documents, Alex and Durwin will look over and approve final documents before adding them to the repository. Once submitted, Erik or Cindy will be able to review and approve the final version of submitted documents to be dedicated to the master branch.

**1.5.2 Technical constraints**

The team has agreed to write this program in C++ as it is the language that we are all most familiar with. We will be building the program in Visual Studios 2015. The program will be limited to machines that are compatible with Visual Studio to prevent any compiling errors. The program will use a command window as its input/output source due to our skill set limitations as well.

**2.0 Project Estimates**

**2.1 Historical data used for estimates**

* 1. **Initial Estimate**

As a basis for our estimates we will draw upon on Erik’s current work experience, a lines of code per hour formula, and sampling of previous projects.

* A lines of code per hour formula will be applied to major programming projects from CIS 200 and up for each team member.
* Sampling previous projects will allow us to determine which projects are significant enough to be included in the lines of code per hour formula.

Erik’s current work experience will give us insight into how much time and effort it will actually take to complete the project.

**2.2.1 SPMP Completion Estimate**

Overview

* This document estimates the number of man-hours required to complete the SPMP.

Estimation

* Based on previous experience with similar but smaller templates from past Computer Science classes, we estimate the SPMP will take approximately 60 hours to complete.

**2.2.2 Overall project estimate**

* + - 1. **Line-of Code Estimate**

Overview

This section describes the estimated count of software lines of code of all the major functionality of the project as obtained by an initial guess for the entire project, based on previous experience with computer science curriculum projects up until this point.

Estimate

Based on prior experience with previous computer science projects, we estimate this project to require 5000 lines of code.

* + - 1. **Function Estimate**

Overview

* + This section determines the count of the functions present in 1.3, the Major Software Functions, to be used for function point estimation. The lowest-level functionality in 1.3 is counted and added up.

Estimation Breakdown

* + Pharmacy Management System - 18
    - Inventory - 2
      * Add/Update/Display inventory - 1
      * Inventory replenishment - 1
    - Sales/Customer - 9
      * POS (Point of Sale) – Cashier - 8
        + Refill & eligibility reminder - 1
        + Coupled item check - 1
        + Company-wide discount check - 1
        + Individual Store / Individual Item Sales Report - 1
        + Customer Coupon Processing - 1
        + Add/Update/Display Customer - 1
        + Print labels - 1
        + Out-of-stock Handling - 1
      * Pharmacist Prescription Approval - 1
    - Batch Processing - 7
      * Company Sales Report - 1
      * Add/Update Store - 1
      * Store to Warehouse Inventory Request - 1
      * Receive and Allocate Inventory (from Store/Vendor) - 1
      * Vendor Reordering & Report - 1
      * Mass Inventory Add/Update - 1
      * Item Stock Threshold Processing (Increase/decrease reorder quantity) – 1

Total – 18 Functions

* + - 1. **Tasks Estimate**

The high level tasks to complete for this project are:

* Creating a communication plan: 1 hour
* Planning the project: 10 hours
* Estimating the project’s time cost: 2 hours
* Gathering requirements for the software: 5 hours
* Designing algorithms for the software: 10 hours
* Creating test plans: 2 hours
* Coding the algorithms: 5 hours
* Testing the software: 3 hours
* Debugging: 3 hours
* Documentation: 5 hours

The initial estimation time in hours is 46 hours per person, for a total of 184 hours.

**2.2.2.4 Total overall project time estimate in hours of effort**

Based on the initial estimated produced from the lines of code, function, and task based estimates, the overall project is estimated to take 162 hours. This is based on the 60 expected hours to complete the SPMP, added to the lines of code estimation at 102 hours (5000 total lines divided by the team average of 49 lines of code per hour. Of the estimates produced from the estimation methods, the lines of code felt to be the most reasonable.

**2.3 Estimation techniques applied and results**

Overview

* This section describes the methodology used to calculate the estimated number of hours of the project using lines of code, based on a wider base of knowledge gained from class lectures and working on the SPMP up until this point.

Estimation Methodology

* + - Lines of Code per Function (based on 1.3)
      * An estimate for the lines of code for each piece of functionality in 1.3 is determined based on previous experience with computer science curriculum projects up until this point.
  + Total Lines of Code
    - The estimates generated in the Lines of Code per Function estimate above are summed up into a single number.
    - Formula
    - Where denotes the number of pieces of functionality and denotes the lines of code per piece of functionality.
  + Lines of Code per Hour
    - Each team member reviewed previous projects they had completed. They counted lines of code and divided by the estimated hours spent to come up with an average lines of code per hour.
    - Criteria
      * Projects Reviewed
        + Any project-type assignments completed in any college programming courses from CIS200 (or equivalent) and higher-level courses.
      * Programming Languages Considered
        + C++
      * Exclusions
        + Assignments completed prior to CIS200

Team Feynman felt these assignments were not a representative measurement of lines of code, as in CIS200 and more advanced classes, a much deeper understanding of code and programming was obtained.

* + - * + Non-project type assignments, such as labs

Team Feynman felt these were a less representative sample for the average, as the lines of code tended to be much lower and the problems were often of trivial complexity.

* + - Formula

      * Where denotes the number of projects, denotes the lines of code for a specific project, denotes the estimated amount of time spent writing the code (by the team member's own recollection) and denotes an instance of a specific project.
  + Team Average Lines of Code per Hour
    - Each team member's result from Lines of Code per Hour is summed up into a single number, and the number is divided by the number of team members, to get the average lines of code per person per hour.
    - Formula
      * Where denotes the number of team members and denotes the lines of code per hour for team member .
  + Final Estimate
    - The Total Lines of Code is divided by the Team Average Lines of Code per Hour, and this is the estimated number of hours based on lines of code.
    - Formula
      * Where denotes the Total Average Lines of Code per Hour and denotes the Team Lines of Code per Hour.

**2.3.2 Estimate for technique 1 – lines of code**

* Overview
  + This section describes the methodology used to calculate the estimated number of hours of the project using lines of code, based on a wider base of knowledge gained from class lectures and working on the SPMP up until this point.
* Estimation
  + Lines of Code per Function (based on 1.3)
    - Pharmacy Management System
      * Inventory
        + Add/Update/Display inventory – 400 LOC
        + Inventory replenishment – 600 LOC
      * Sales/Customer
        + POS (Point of Sale) – Cashier

Refill & eligibility reminder – 400 LOC

Coupled item check – 200 LOC

Company-wide discount check – 100 LOC

Individual Store / Individual Item Sales Report – 300 LOC

Customer Coupon Processing – 100 LOC

Add/Update/Display Customer – 500 LOC

Print labels – 200 LOC

Out-of-stock Handling – 400 LOC

* + - * + Pharmacist Prescription Approval – 200 LOC
      * Batch Processing
        + Company Sales Report – 300 LOC
        + Add/Update Store – 500 LOC
        + Store to Warehouse Inventory Request – 300 LOC
        + Receive and Allocate Inventory (from Store/Vendor) – 500 LOC
        + Vendor Reordering & Report – 300 LOC
        + Mass Inventory Add/Update – 100 LOC
        + Item Stock Threshold Processing (Increase/decrease reorder quantity) – 300 LOC
  + Total Lines of Code: 5700
  + Lines of Code per Hour
    - Durwin Johnson - 53
    - Erik Johnson - 63
    - Alex Pope - 40
    - Cindy Samano – 40
  + Team Average Lines of Code per Hour: 49
  + Final Estimate: 117 Hours (116.33 rounded up)

**2.3.3 Estimation technique 2 – function points**

Overview

* + - This section describes the methodology used to calculate the estimated number of hours of the project using function point estimation, using the techniques provided to us in class in Powerpoint 2, Slides 31-37.
* Estimation Methodology
  + Functionality Breakdown – Each piece of functionality specified in the project specification sheet, as well as additional functionality, is broken down into units that can be categorized as one of the following (from Slide 33):
    - **EI - external inputs**, which are the components responsible for introducing changes in system's internal data.
    - **EO - external outputs**, which are the ways system's internal data can be presented, but beware - there are a few similarities with EQ components, though.
    - **EQ - external inquiries**, which are the methods for reading system's data without modifying it.
    - **EIF - external interface files**, which are responsible for exchanging data with other systems.
    - **ILF - internal logical files**, which are files that are being used by the system itself.
  + Complexity Classification
    - Each piece of functionality from the Functionality Breakdown is assigned a complexity rating, based on a best-guess estimation of complexity, which is in turn based on prior experience with computer science projects and curriculum.
  + Function Point Valuation
    - Each piece of functionality is assigned a point value based on its Complexity Classification, utilizing the following table (Slide 36):

|  |  |  |  |
| --- | --- | --- | --- |
| **Component:** | **Complexity:** | | |
| **Low** | **Average** | **High** |
| EI | 3 | 4 | 6 |
| EO | 4 | 5 | 7 |
| EQ | 3 | 5 | 6 |
| EIF | 5 | 7 | 10 |
| ILF | 7 | 10 | 15 |

* + Function Point Summation
    - The Function Point Valuation of each piece of functionality is summed to generate the Total Function Point Valuation (Slide 37)
      * Where denotes the number of pieces of functionality and denotes the function points for the th piece of functionality.
  + Total Hours Estimation
    - The Total Function Point Valuation is multiplied by 8, which Slide 37 explains is the number of hours equivalent to 1 Function Point.

**2.3.4 Estimate for technique 2 – function points**

Overview

* + - This section describes the methodology used to calculate the estimated number of hours of the project using Function Point Estimation, based on a wider base of knowledge gained from class lectures and working on the SPMP up until this point, as well as the techniques detailed in CIS375 Powerpoint 2 - Software Engineering Project Management, slides 31-37.
* Estimation
  + Functionality Breakdown
    - Overall Functionality
      * Inventory
        + Store / Warehouse data updates (store info, item stock, etc)
        + Item-level data updates
      * Customer
        + Point of sale purchasing/ordering
        + Point of sale customer data updates
      * Transactions
        + Batch transactions (overnight)
        + Online transactions (point of sale)
    - Category Breakdown
      * External Inputs (EI)
        + Store / Warehouse screen - Average
        + Inventory screen - Average
        + Point of sale screen - High
        + Customer account screen - Low
        + Pharmacy prescription approval / fill screen - Low
        + Batch file processing input - High
      * External Outputs (EO)
        + Store / Warehouse screen - Average
        + Inventory screen - Average
        + Point of sale screen - High
        + Customer account screen - Low
        + Pharmacy prescription approval / fill screen - Low
        + Sales screen (reports) - Average
      * External Inquiries (EQ)
        + Sales screen (reports) - Average
      * External Interface Files (EIF)
        + Vendor order file - Average
      * Internal Logical Files (ILF)
        + Transaction table - High
        + Customer table - Average
        + Store table – Average
        + Store/Warehouse Inventory table - Average
        + Item table - Average
  + Function Point Valuation

|  |  |  |
| --- | --- | --- |
| * Category | * How Many | * Complexity |
| * EI | * 2 | * 3 |
| * EI | * 2 | * 4 |
| * EI | * 1 | * 6 |
| * EO | * 2 | * 4 |
| * EO | * 3 | * 5 |
| * EO | * 1 | * 7 |
| * EQ | * 1 | * 5 |
| * EIF | * 1 | * 7 |
| * ILF | * 4 | * 10 |
| * ILF | * 1 | * 15 |

* + Function Point Summation: 117 Total
  + Total Hours Estimation: 936 Total Hours

**2.3.5 Estimation technique 3 – process/task**

Tables or equations associated with lines of code estimation technique m are presented. Re-estimate effort based on actual tasks defined in completed SPMP.

**2.3.6 Estimate for technique 3 – process/task**

Estimate in hours generated for process/task technique.

**2.4 Reconciled Estimate**

* Overview
  + This section will use the previous estimation results to select a final estimation for how long our team believes this project will take to complete. The previous techniques used were Lines of Code, Function Point Estimation, and Task/Process estimation.
* Estimation Data
  + Results
    - Lines of Code Estimate: 117 Hours
    - Function Point Estimate: 936 Hours
    - Task Estimate: 399 Hours
  + Person-Hours Spent So Far
    - Total Task Time from Toggl: 116 Hours
* Estimation Selection
  + We rejected all the estimates, and went with our own estimate of 525.5 person-hours.
* Estimation Selection Rationale
  + We determined that we had spent 116 hours thus far within the time that we began working together up until today's date (2/6/2017 – 2/24/2017). We then determined the number of days left until the final project is due – 4 days left in February, 31 days in March, due on the 28th of April = 63 days. We then divided the total number of days left by the number of days we had been working on the project: 63 / 18, to arrive at 3.5. We then multiplied this number by 117, to arrive at 409.5 hours. We added 116 to this number (the amount of time spent so far) and arrived at 525.5 hours. The closest to this number was the Task Estimate, which is still off by 126.5 hours. Therefore, we rejected the other estimates and went with 525.5 person-hours.

**2.5 Project Resources**

**People**  
There are 6 people to be involved in this project, with Erik Johnson acting as the team leader. Team members consist of Durwin Johnson, Alex Pope, and Cindy Samano. The assigned project managers are Wenhao Zhang, and Fnu Jirigesi.

**Software**There are various software programs that will be used in the process of completing this project.

* Toggl, a time tracking program will be used to document the time each member has spent on the project, as well as keeping times for meetings and other project related efforts. It has a browser app, a desktop app, and a mobile app. Each of these will be used based on preference by individual members.
* Wrike will be used to facilitate the project planning coordination and its associated time tables.
* GitHub will be used for the documentation and coding pertaining to the project, as it allows for the tracking of various versions developed in the process of the project as well as providing easy access for all involved in the project as necessary.
* Slack is to be used as the primary program for general communication between team members, leaders, and managers. There are desktop and mobile applications available, both of which will be used.
* Microsoft Word will be used for drafting and generating the final versions of the artifacts/documents.
* Microsoft Visual Studio 2015 will be used for coding, interface working, and debugging the program.
* Microsoft Excel will be used to plan and draft the task diagram and time tables.
* Microsoft Visio will be used to construct the decomposition diagram.
* Microsoft Project will be used to construct the task network diagram.

**Hardware**:

* Windows machines for each member will be used to produce artifacts and the code for the project.
* An iPad
* Phones for away from computer communication between members

**Tools**No physical hand tools will be required for this project.

**Other**Various miscellaneous supplies will be used throughout this project for planning and hand drafting purposes.

* Pens
* Pencils
* Paper
* Notebooks

**3.0 Risk Management**

This section discusses project risks and the approach to managing them.

**3.1 Project Risk Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Risk** | **Description** | **Probability** | **Impact** | **Mitigation** | **Contingency** |
| **Dangerous Travel Conditions** | This project takes place during the winter semester of school. At some point the roads may become dangerous or un-drivable. | Medium | Medium | All team members will monitor weather reports online and in the news. | Multiple sources of mobile communication have been put in place to allow meetings and discussion to take place without being face to face. An alternative meeting location has also been chosen. |
| **External Obligations** | All team members are students and it is likely we will all have other assignments or obligations throughout the semester. | High | Medium | A google calendar will be used to keep track of updated availability in case a team member is given an important assignment in another class. | A team member may pass on some work to another volunteering member under the agreement that they will pick up extra slack at another point of the project. All work must be even by the end of the project cycle. |
| **Loss of Work/Data** | Since we are working with computers, there is always a chance of random failure that results in lost work. | Low | High | Weekly backups will be kept of our GitHub repository. In addition, team lead Erik will keep an additional backup of the repository on google drive. | In the event of data loss, we will resort to our last weekly backup of the GitHub repository. If that source is unusable, we will use the secondary back up provided by Erik. |
| **Team Switching/Loss of Member** | It has been hinted that team members will be switched at some point during the project. In addition, there is also a chance that a team member may be lost. | High | High | All team members will keep track of each member’s responsibilities as the project progresses. Team members will also be sure that their work is clear and easy to understand. | If a team member is lost, their assignments and documents will help their replacement understand their work and quickly continue it. If the team lead is lost, Cindy will take over administrative position. |
| **Unfavorable Outsourcing** | Part of our code will be outsourced. There is a chance that the code we receive will be poor and unusable. | Low | High | The team will make sure the algorithm we supply to us outsource will be clear and easy to understand. | If the code we receive is unusable, then we will use the algorithm we wrote to produce a version ourselves to save time rather than trying to fix a code that is beyond repair. |

**3.2 Overview of Risk Mitigation, Monitoring, Management**

**Hazardous Travel**

Overview:

Since it is the winter semester of school, the likelihood that travel will become difficult and dangerous due to inclement weather is very high. Weekly meetings are planned for Sunday evenings, but we cannot always be sure everyone can safely arrive. To prevent meeting disruption, weather will be monitored regularly through online resources and news stations and an alternative meeting location has been established.

**In the Event of Hazardous Travel**

Alternative means of communication will be established in case an in-person meeting cannot occur. Slack has already been established as a main mobile source of communication by all team members. If necessary, visual means such as skype can also be utilized to add visual communication. In the event of library closure, the team has agreed on an alternative meeting location at Biggby Coffee at 25421 Ford Rd. This location is conveniently located for all team members. Task assignments will be discussed in person and reiterated on Slack and also on Wrike.

**School and Other Obligations**

Overview:

Our team consists entirely of students. That means that all of us at some point will have other work that must be dealt with. Exams, projects, and other tasks will be assigned to all of us in other classes throughout the semester. We cannot expect every team member to be able to contribute all their time to this project the entire semester. Therefore, tasks must be distributed to accommodate each team member's schedule and obligations. To accommodate, we have created a Google Calendar that all team members have access to. It is here that they may list dates they cannot attend meetings in their normal availability. This will be updated every week before Sunday. Additionally, we must make sure that this distribution stays even amongst the team throughout the semester. Any slack given must be made up at another point in the semester by that team member.

**In the Event of External Obligations**

If a team member has another obligation that must be attended to, they may have some of their tasks picked up by another team member. However, by passing tasks on to another member, this team member agrees to take on extra responsibility on the next set of tasks assigned. This will keep the distribution of work fair and even throughout the project. Under no circumstances should one team member be doing more work than the rest involuntarily and without future compensation from the teammate he is taking extra work from.

**Loss of Work or Data**

Overview:

When working with any form of technology, sudden loss of data is always a possibility. Careful measures should be taken throughout the project cycle to minimize the chances of this occurring. A weekly backup off all uploads on GitHub will be downloaded once per week via the GitHub desktop app while the project is underway. Additionally, a secondary backup will be kept on Google Drive by team lead Erik in case GitHub becomes unavailable. This will minimize the amount of work that could be lost under catastrophic failure. If backups are kept diligently, a worst-case scenario would only produce a week's worth or less of lost data. Keeping contents shared among team members will also mitigate losses and make lost information easier to reproduce.

**In the Event of Loss of Data**

In the event of data loss, the team will roll back to the latest version available of the document that was lost from our most recent backup. The GitHub recovery will be used primarily, however the secondary backup will also be prepared in case the first is lost or unusable. A worst-case scenario should be a week's worth of work (assuming weekly backups are kept). It is also advised that documents be shared amongst the team so that more than one person knows the contents of documents. This way, the information is more spread amongst the team and makes it easier to replicate the lost information.

**Team Switching or Loss of Member**

It has been made apparent that at some point during the project cycle we will lose one or more members of our team. Team members should do their best to make sure that their work is done cleanly, clearly, and is well documented. Additionally, everyone should know and understand what each member of the team is doing and is responsible for throughout the project.

**In the Event of Loss of Member**

If all the mitigation strategies are followed and if all team members understand the tasks each member takes then there should be a smooth transition for any new team members. We can use previous documents and notes from each task to better clarify and complete any unfinished work from a lost member. Maintaining traceability is also critical, as it will make it much easier to track progress and process, making understanding others work much easier. Ideally, any new member should be able to understand the process we have followed quickly and can contribute to the project.

**In the Event of Loss of the Team Lead**

Admin privileges will be given to team lead Erik and to Cindy. Dispersal of the documents ensures no information is left to only one person to have access too. This will allow a smoother transition in the event we lose our team lead. However, the team lead still holds all administrative powers unless stated otherwise. In the event that the team lead is lost, his administrative privileges will be handed off to Cindy to take over. This ensures that no unnecessary modifications or edits are made without the consent of the team. This prevents any hard to find errors or code breaking changes from slipping past the teams vetting process.

**Outsourcing**

Overview:

Outsourcing refers to giving work to an outside source to be completed for us. It has been inferred that we may be outsourcing some of our coding to a CIS 200 student. To minimize the risk of this code being unsatisfactory, steps will be taken to ensure the code of sufficient quality. We must be sure our algorithm is detailed and as clear as possible. The easier it is to understand, the more likely we will receive a favorable result.

**In the Event of Unfavorable Outsourcing**

It has been agreed that while our code is being outsourced, we will closely monitor its progress as its being written. If the code seems favorable and well done, then we will continue to use that code and integrate it with our plans. However, if we find that the code we receive is poorly done or incomplete beyond repair, we will use the same algorithm we supplied the external source to write the code ourselves. We believe that this is much more cost effective than spending too much time trying to repair the code we received. This will be done only if we believe the code received is unsalvageable.

**4.0 Project Schedule**

**4.1 Project task set**

For this project, the decision to use the Waterfall process flow was made. The scale of the project overall is relatively small, making a linear approach to the process the simplest.

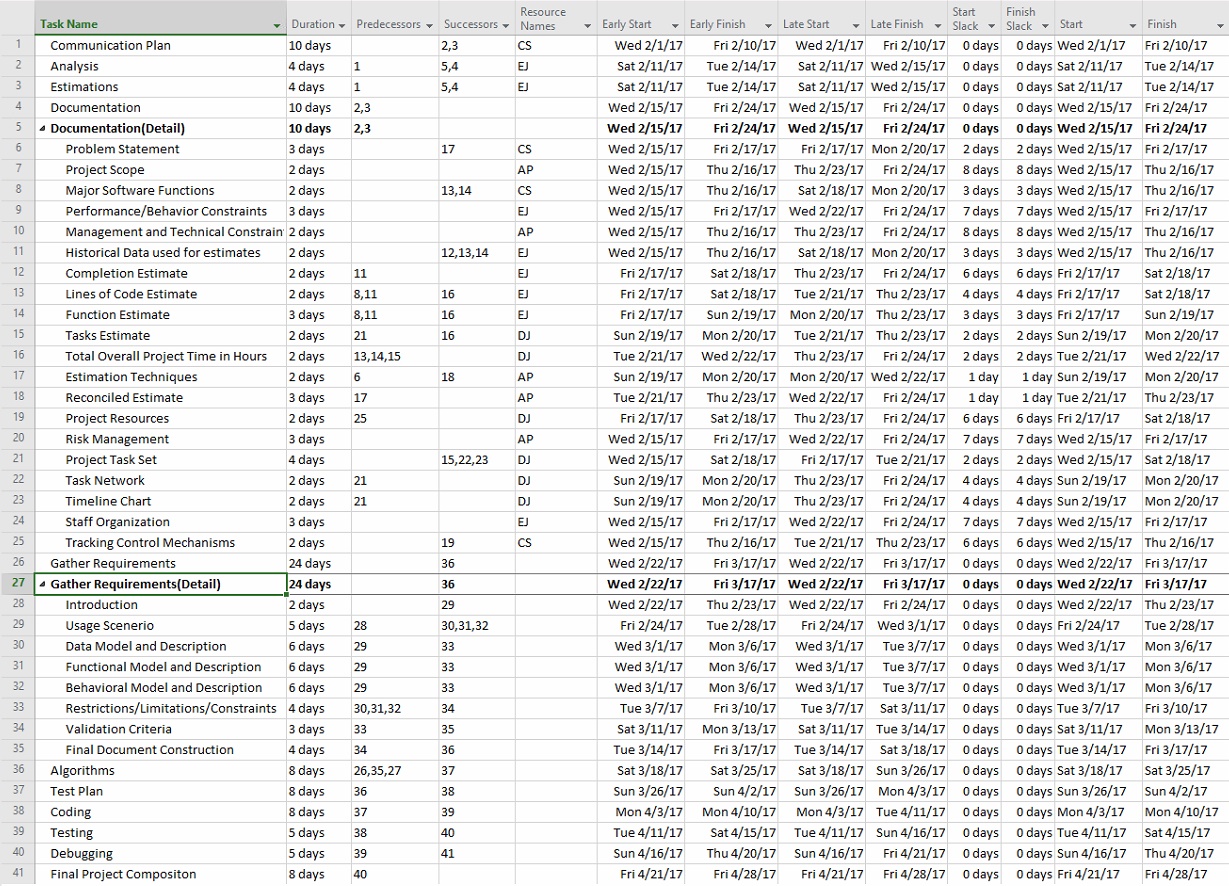
**Framework Tasks:**

* **Communication**:  
  The first step in the process is to establish the means of communications between members. This includes selecting tools to facilitate formal and informal communication, as well as laying out the availability of each member to select best meeting times and working schedules. This will reduce as many information related delays as possible by providing a steady flow to all members regularly. Adjustments will be made as necessary throughout the entire process.  
  Included tasks:
  + Create general availability calendar
  + Create an exception calendar
  + Create a document laying out communication plan
* **Planning:**  
  This phase in the process will consist of gathering preliminary estimates and an analysis of the general problem. This allows time to figure out both the general tasks that must be completed, and how much effort will be involved in those tasks to complete the project on time. This will allow planning the proper scheduling of tasks based on their estimated times to reduce tasks related delays in later stages of the process. Adjustments will be made as necessary throughout the entire process.  
  Included tasks:
  + Analysis of the problem
  + Estimations for effort to solve the problem
  + Create a document laying out the software project management plan
* **Modeling(Design):**  
  Towards the end of the planning phase, the modeling and design of the software can begin. This includes gathering a definitive set of requirements for the program, the design of the correlating algorithms, and the development of the testing plan. This is to ensure all requirements of the program are met, and reduces the time spent on the coding of the software.  
  Included tasks:
  + Gather software requirements
    - Develop Use Cases to develop a list of requirements
    - Develop Test Expected outcomes
  + Develop algorithms corresponding to the requirements
    - Create diagrams to visually describe relations and flows
  + Ensure each algorithm function matches a requirement
  + Ensure each requirement fits into an algorithm
  + Develop test cases for the algorithms to find possible errors
* **Construction:**  
  After the modeling and design of the software is completed, the algorithms will be converted into code. Testing on pieces of the software using the test cases is expected to take place both throughout the coding process, as well as upon its completion.  
  Included tasks:
  + Convert algorithms into code
  + Ensure all code matches an algorithm
  + Ensure each algorithm has a coded representation
  + Apply test cases to the code
  + Debug any discovered errors and correct them then re-apply test cases to the code

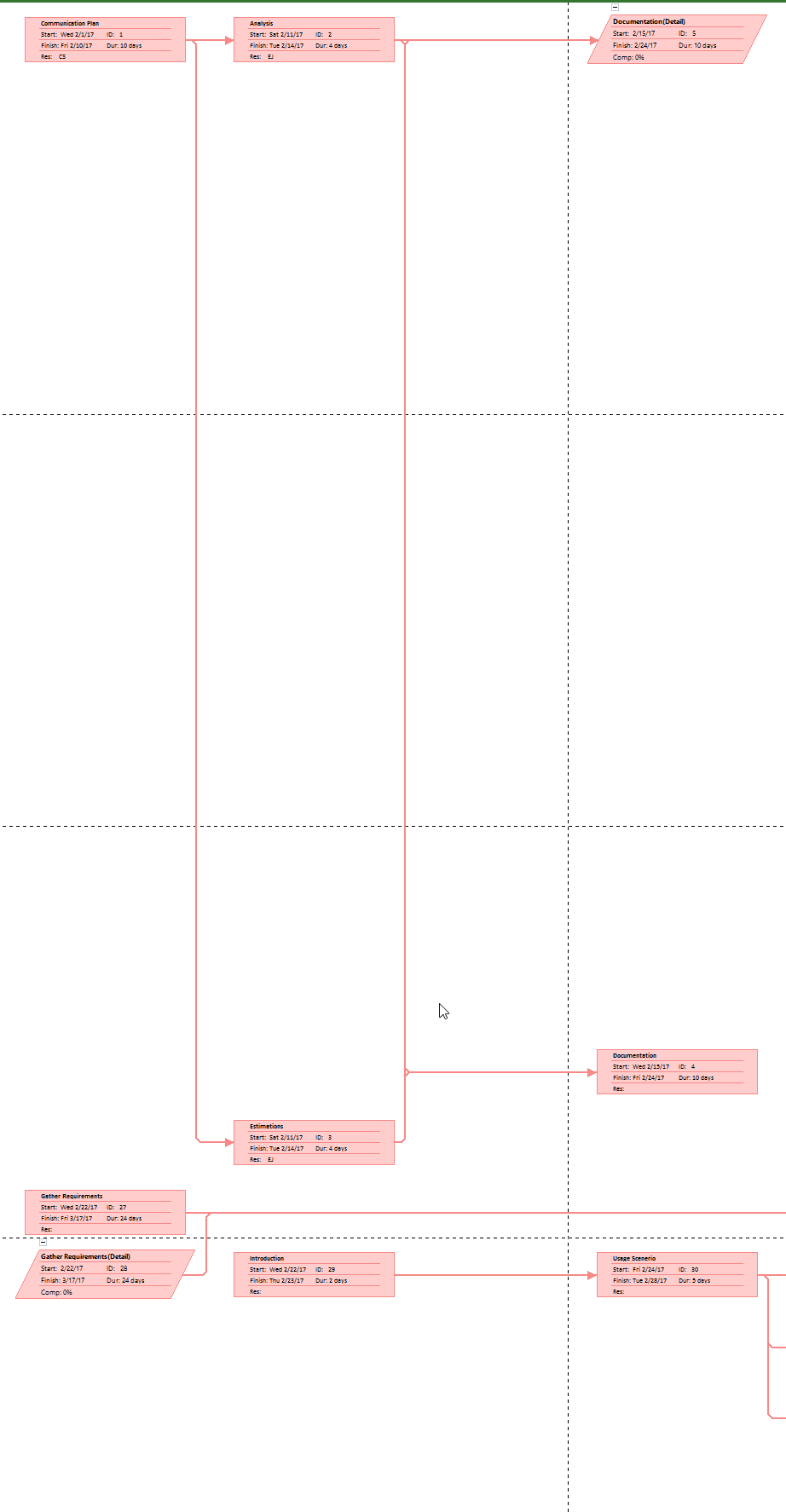
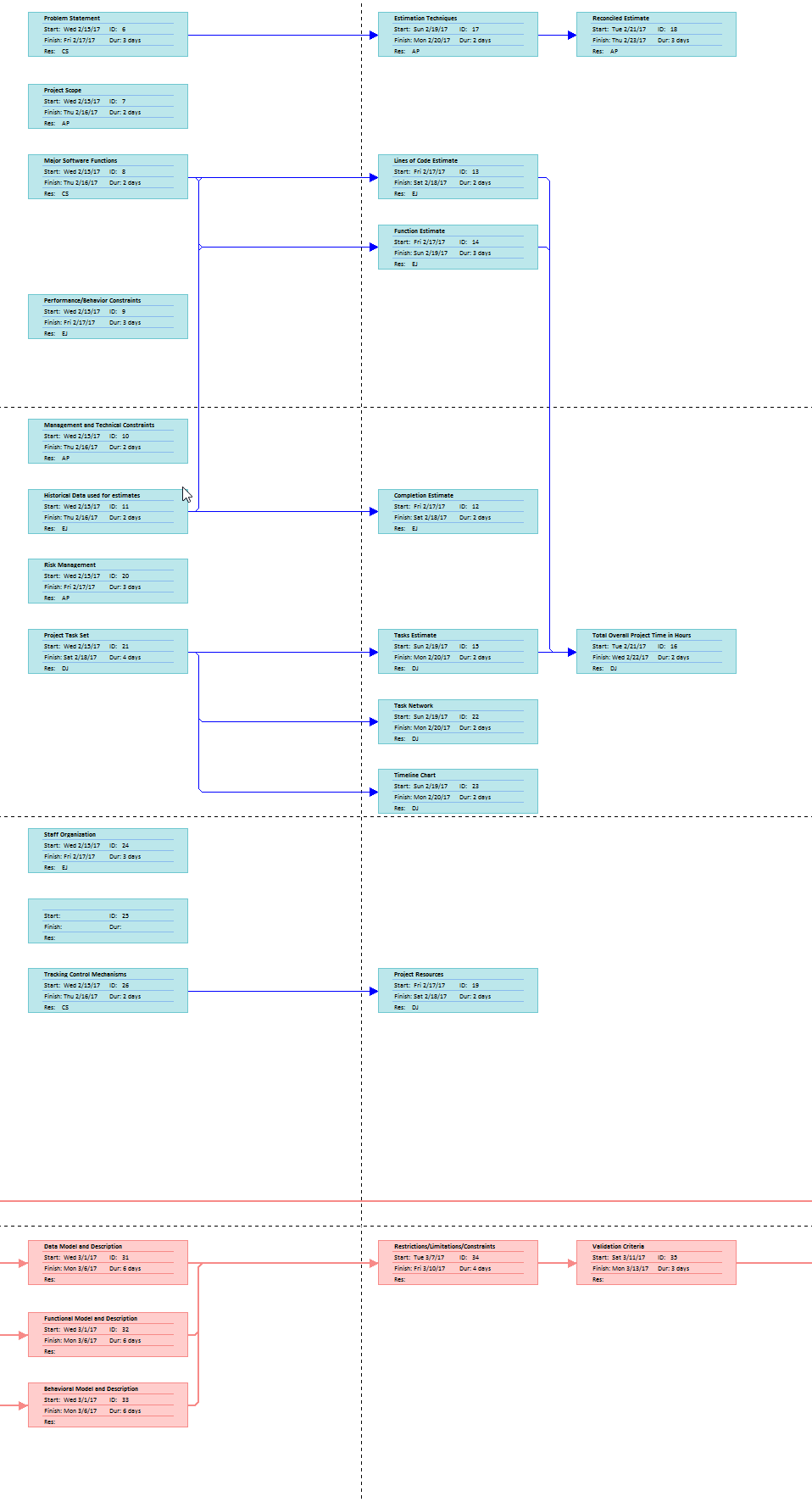
* **Deployment:**  
  This step takes the place of a presentation and demonstration of the software. This is to occur upon the expected completion of the software on April 28, 2017.  
  **Included tasks:**
  + Prepare final document
    - Prepare instructions to run code
* Prepare presentation

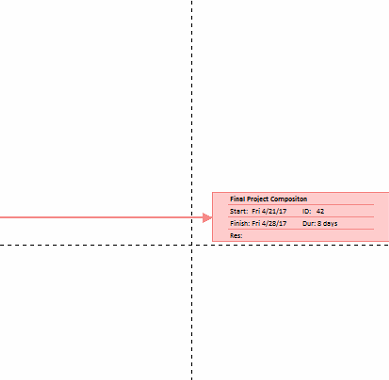
**4.2 Task network**

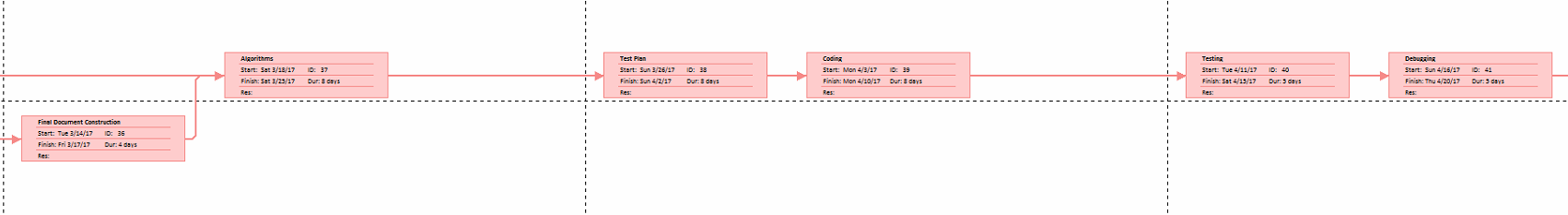
****



**Task Network Diagram:**





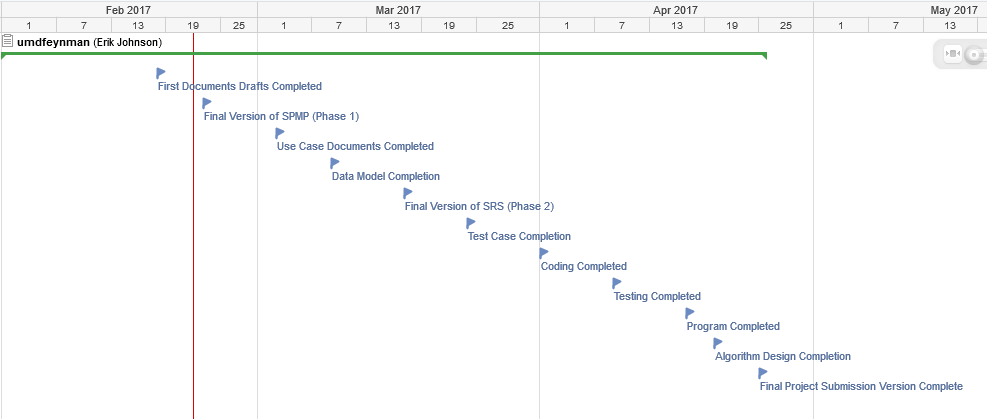
****

**4.3 Timeline chart**

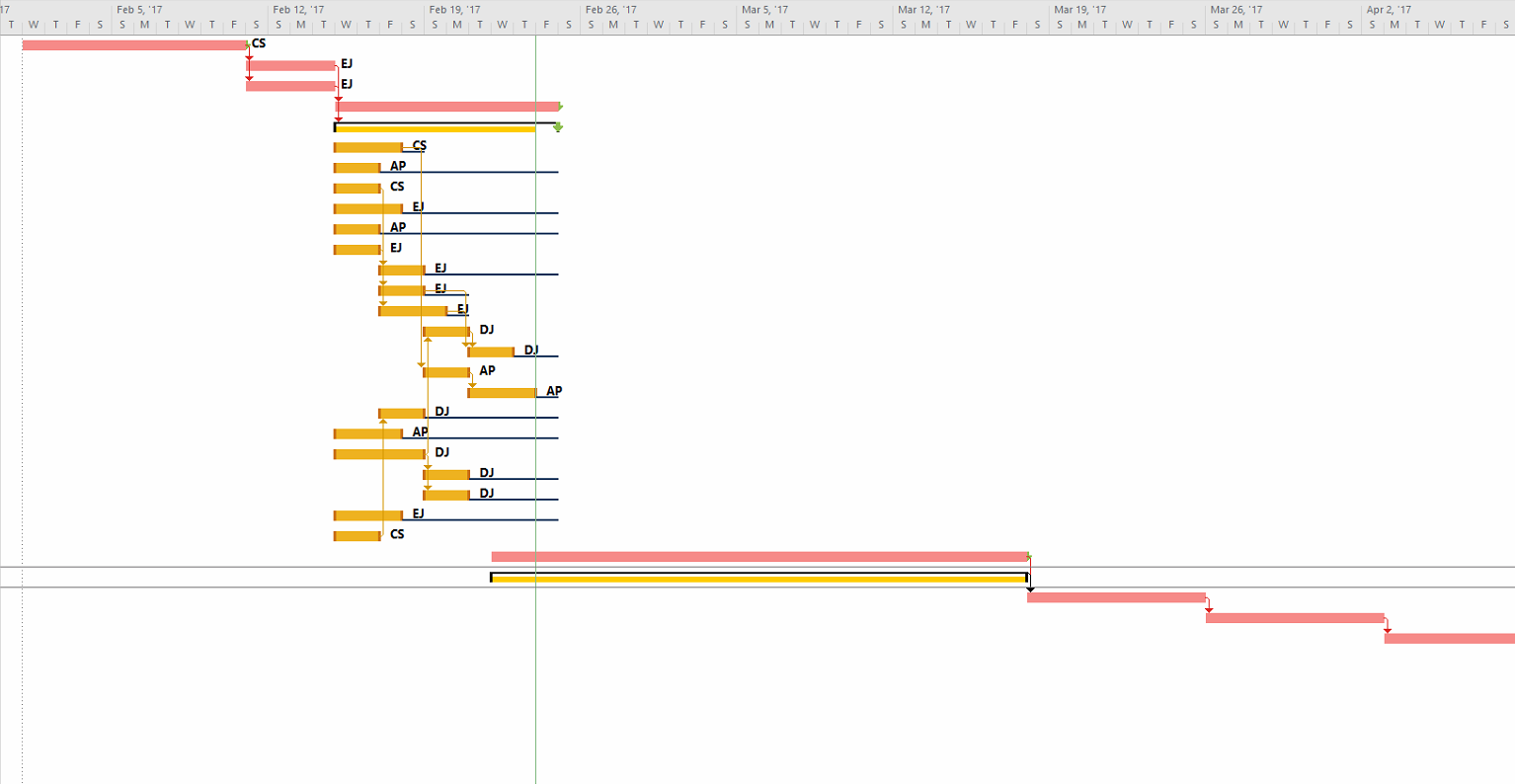
To say on target and ensure the project is completed on time, the milestones have been set to a date before the estimated latest the task can be completed without taking time from other tasks. By setting the milestones ahead of the scheduled time, the mental goal will be to finish early instead of on time. This also allows for a clear indication when the team is on track, before it becomes too late to make a change in the working process for that task.

|  |  |
| --- | --- |
| **Milestone** | **Date** |
| First Documents Drafts Completed | 2-18-2017 |
| Final Version of SPMP Completed (Phase 1) | 2-23-2017 |
| Use Case Documents Completed | 3-3-2017 |
| Data Model Completion | 3-9-2017 |
| Second Document Drafts Completed | 3-14-2017 |
| Final Version of SRS Completed (Phase 2) | 3-17-2017 |
| Algorithm Design Completed | 3-20-2017 |
| Test Case Completion | 3-24-2017 |
| Coding Completed | 4-01-2017 |
| Testing Completed | 4-09-2017 |
| Program Completed | 4-17-2017 |
| Final Project Submission Version Complete | 4-25-2017 |

**Visual Timeline:**



**Overall project timeline:**



**5.0 Staff Organization**

**5.1 Team structure**

* Business / Customer
  + Professor Steiner
* Project Management Team
  + The Project Management team is responsible for:
    - Signing off on artifacts and code produced by the Development Team
    - Signing off on progress reports produced by the Team Lead
    - Assisting with reviewing project artifacts to ensure they meet sufficient quality
    - Providing recommendations and guidance with regard to all facets of the project
    - Presenting a finalized copy of the project materials at the end of each project phase to the Customer
  + Team Members:
    - Reggie Jirigesu
    - Wenhao Zhang
* Development Team
  + The Development Team is responsible for:
    - Producing the artifacts involved in the project
    - Writing the source code for the project
    - Communicating with the Project Management Team
    - Presenting a finalized copy of the project materials at the end of each project phase to the Project Management Team, and to the Customer
    - Providing an evaluation of other Developers, the Team Lead and the Project Management Team at the end of each project phase
  + Team Members:
    - Erik Johnson - Team Lead
      * The team lead is responsible for:
        + Coordinating the development team's tasks and meetings

Transcribing meeting minutes and providing them to the team for signoff

* + - * + Submitting a progress report to the customer and project managers by Monday end of day each week
        + Team and project manager signoff on artifacts
    - Durwin Johnson, Alex Pope, Cindy Samano - Developers
      * Each team member is responsible for:
        + Completing tasks assigned by team lead

**5.2 Management reporting and communication**

**Selected Communication Methods**

**1.1 Communication Tools**

We will be using Slack.com as our main source of communication. Slack is a real time team communication system that will allow for instant communication between team members. Slack is accessible online and as an app on mobile devices through both the Apple Store and Google Play Store. In addition, Slack allows for the creation of multiple “channels” which are essentially other boards with a specific purpose. There will be a channel where we will discuss general things, and there will be a channel named “meeting\_minutes”, that will contain an overview of the discussions from in-person meetings.

**1.2 Communication with Project Managers**

Our designated Project Managers are: **Fnu Jirigesi** and **Wenhao Zhang**. Both Managers will be added into our group in Slack. They will be added to all channels created in the group as an effort to keep them involved and informed. There will also be a specific channel dedicated to talking specifically and directly to them whenever necessary.

**1.2.1 Informal Communication**

Slack will be used as the informal form of communication with Project Managers. Informal communication involves quick questions, additional help, and asking for opinions or ideas.

**1.2.2 Formal Communication**

All formal communication with Project Managers will be handled through email. This would be in necessary in cases such as team leader sending weekly reports and updates, or reporting a team member for lack of involvement in project.

**1.3 Expectations of Team Members and Project Managers**

Each team member will create an account on Slack and join the group created by our team leader. Team members will be required to check Slack at least once a day in order to stay up to date.

Under the Meeting\_minutes channel, team members and project managers are required to sign off on the information posted about the in-person meetings we will be having. Sending a message with your initials will be the accepted form of signing off and agreeing to the contents of the information posted. All team members and project managers are required to sign off on the channel discussions by the end of the next day after an in-person meeting (meaning you will have one day), failure to do so will symbolize that you do not agree with some of the contents in the post and you will be contacted by the team leader for follow up questions the following day. The team lead must assure that all stakeholders sign off. If they have not, it is the team lead's responsibility to contact them and make sure they do.

**Team Availability**

**2.1 In-Person Meetings**

There will be a Google Doc set up with a day of the week and time table and it will formally be named “Availability Sheet.” Each team member is required to fill in what days and during what one-hour time periods they are available on a regular basis. This will give us a general overview of what days’ team members will be available for in-person meetings. The days with the most overlap, will be the days that will be considered for in-person meetings.

Based on information that we have already gathered through the Availability sheet, in-person meetings will be held on Sunday between 4pm and 6pm on campus on the second floor of the Mardigian library at UM Dearborn unless stated otherwise and agreed upon by all team members. Risks pertaining to in-person meetings will be discussed in the mitigation plan.

**2.2 Exceptions**

If there is an in-person meeting that you know you will not be able to attend or there is a change in your general availability, there will be a Google Calendar titled “Exceptions”. In this calendar you will be able to add in a day that you will not be available. Team members are required to update the exception calendar every Sunday at 7 pm at the latest. If the calendar is blank or there are no entries for a team member it will be assumed that person has no exceptions and is fully available during the times specified in the Availability sheet. You are allowed to add in more in future exceptions.

**Contact Information**

Along with the Availability Sheet, there will be a table provided at the bottom where each team member is required to write their contact information. This includes cell phones numbers, and emails or any other form of we can get in touch with you. These forms of contact are to be used in the event that we are not able to contact each other through Slack.

**Customer Progress Report**

The Team Lead will provide the customer/management a project status report that contains a 1-2 sentence description for each of the following pieces of information:

* + - What was accomplished the prior week
    - What is to be accomplished next week
    - Issues the team is currently dealing with
    - Potential risks
    - Risks identified and being responded to in mitigation or contingency fashion
    - A color value identifying the status of the project:
      * Green - Everything OK
      * Yellow - We're having some trouble
      * Red - We're having serious trouble

The status report will be signed off on by all team members and the project managers, and delivered via Canvas.

**Team Coordination and Task Assignment**

At team meetings on Sunday, assignments and tasks will be considered, discussed, expanded on, and determinations will be made so that each team member is clear as to exactly what they should do.

The Team Lead will communicate to each team member their assignments verbally as well as listed in the meeting minutes for each week.

**6.0 Tracking and Control Mechanisms**

**6.1 Quality assurance and control**

**Introduction**

* + This Software Quality Assurance plan (SQA) will outline the processes, methodologies, standards, and procedures of how work will be performed to assure a timely and quality delivery of the pharmacy system.

**Overview**

* + The SQA will be organized as follows:
  + **Management:** Description of major organization components and the SQA tasks and relationships.
  + **Documentation:** Identify documents related to development, versioning and verification.
  + **SQA plan Requirements:** Define the SQA reporting, review and auditing procedures to ensure deliverables are on time, with quality and in accordance with the SQA plan and project requirements.
  + **Training:** The necessary skills needed to communicate with team members, document artifacts and develop code.

**Management**

Organization: There are four team members involved in the process. Work will be divided into tasks for each team member. An equal amount of tasks (effort or amount) will be distributed among team members. Team must report to their project managers about the status of the project. Project managers will be required to check on each task assigned to team members and report back as soon as possible.

**Team structure**:

* Two project managers
* One team lead
* Three other members (equal amount of work as team lead)

**Responsibilities**:

* **Project managers:** Provide feedback and guide team members. Will act as middle man between client(Professor) and team members.
* **Team lead:** Must send weekly progress reports to the Professor, submitted to canvas every Monday.
* **All team members:** Develop a SPMP that will guide the development of the project. Update the SPMP as the project progresses. Involve project managers at all stages of the process. Evaluate Project managers, themselves and other team members on their performance throughout the project.

**SQA implementation:** Throughout all the software life cycles the implementation of the quality assurance plan will be carried out until the release of the software.

* Software Requirements phase: During the development of the SRS, team members must ensure that the required functionalities are included. Required functionality is of more importance than wanted functionality. The required functionality should be refined until they are completely understood and stated.
* Planning phase: SPMP artifacts should be reviewed daily until the end of the project.
* Design phase: Code and algorithms should be reviewed daily and weekly to ensure quality.

**Documentation**

**Software Project Management Plan:** Will state the problem statement, the scope of the project, major software functions, performance/ behavior constraints, project estimates, risk management, project schedule, staff organization, and tracking and control Mechanisms.

**Software requirements specification:** Will list the essential requirements of the software. Essential requirements being: introduction, usage scenarios(s), data/functional/behavioral model description, functionalities, and constraints and validation criteria.

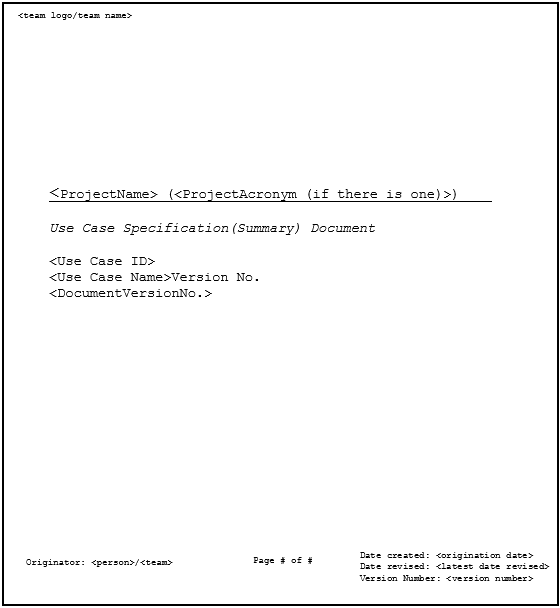
**Uses cases:**

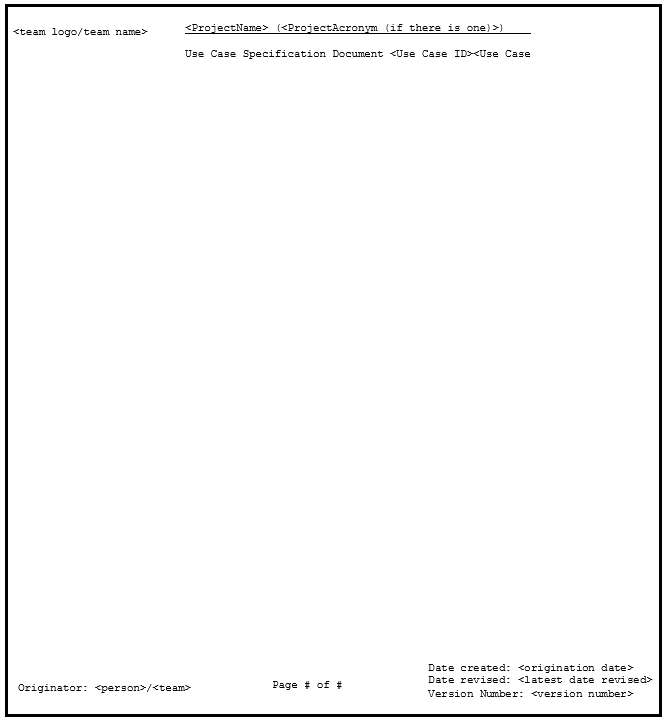
* Specification: Captures the detailed functional and non-functional business requirements. Technical or application requirements should not be detailed here.
* Summary: This document captures business requirements at a high level by identifying all the use cases in a project/application.

**SQA plan Requirements:**

**Standards:**

* Coding standards: C++
* Write-ability
  + Documents**:**
    - Software Project Management Plan(SPMP): High level language will be used in during part one of the SPMP process. As the project progress, the language will evolve to a more business based language.
    - Uses cases: Business language will strictly be used in use case specification and summary to capture business requirements.
* Read-ability
  + Artifact templates:
    - Software Project Management Plan (SPMP) template: The SPMP will follow the template provided in class.
    - Use cases: Use case summary and use case speciation will follow the template provided in class. The layout of each page of the use case specification and summary is as follows:





* Font and size**:** All documents will use Courier New font. Section headers will be in 14pt and boldface. The text that follows will be in 12pt, any titles of a major topic will be boldfaced.

**Reviews and audits:**

* Communication plan
* Software documentation
* Estimate review
  + Metrics: Lines of code per hour, function point and task estimate will be used to determine the size of the project and software.
* Traceability review
  + Every artifact has to have traceability.
* Design review
* Code review
* Test plan review
* Test cases review

**Testing:** Both testing and error/defect collection and analysis will require every line of code to be testes as a part of some test case.

* Unit testing
* Integration testing
* System testing
* Acceptance testing

**Error/defect collection and analysis:**

* Types of data
* Types of defects

**Change management:** Make sure a disciplined process is defined. Only make changes when they are needed, not wanted.

* Process method
* Change handling

**Safety:**

* Environment

**Security management:**

* Access to artifacts
* Access to tools

**Risk management:** Risks must be mitigated, monitored and managed

**Noncompliance documentation**:The hierarchy of reportingnoncompliance must be followed.

**Training**

Team members will be required to obtain or already have the skillset to use the following tools that will be used through the software development process:

* Visual studio 15
* Google docs
* Google calendar
* Microsoft word
* Wrike task manager
* Slack online group messenger
* GitHub

**6.2 Change management and control**

Communication of changes and how they will be handled will be accomplished through both Slack and GitHub.

**Source code:**

* Slack will be used as the informal method of communicating a change. On Slack a team member will notify the others of a change to a file and inform them of any pull request review.
* GitHub includes a functionality that allows for a brief description of the changes made to the file when attempting to commit to the master. Team members will be required to write an adequate and general overview of what changes have been made to the file when requesting a pull request review of a file. Doing so will allow for fluid traceability of any bugs that may arise through the merge.
* Merge-ability of pulled requests can be accomplished through protected branches. Protected branches will ensure that no irrevocable changes will be made to the master branch without the having at least one administrator review the request along with other team members not involved in the file changes. One administrator will sign off and submit detailed reviews (if any) with a unanimous team decision to either approve, defer or reject pull requests. Administrators must ensure that all branches that are created are protected branches.

**Non-source code(documents):**

* For documents such as, but not limited to, the Software Project Management Plan (SPMP), communication of a change will also be accomplished through Slack and GitHub.
* On Slack, a team member will inform the others of a changed or added document to the GitHub repository. These documents will not require review by the other team members and can be uploaded directly to the master during the first part of the SPMP process. After part one of the SPMP, all requests to commit artifacts to the master will require at least one administrator approval.
* On Slack there will be a channel labeled “meeting\_minutes” where the team lead will document the changes and post it onto Slack.

**6.3 Tools**

The tool for control access and versioning of artifacts will be GitHub.

**Source code:**

* Versions of the source code will be found on the GitHub repository through “branches”. Changes are proposed in a branch to avoid overwriting the master branch. The master branch will act as the baseline and will only contain published and approved work.
* The team leader will have admin access to the repository. With the exception of one team member who will also have admin access, all other team members will have read and write access to the repository.

**Non-Source Code(documents):**

* To differentiate versions of the documents on GitHub, an underscore “\_” in the document name will indicate the document is still in progress, a document name without underscores “\_” will indicate that that is the final version of the document.

**7.0 Appendix**

Supplementary information is provided here.