Team OC

vLearn Course Management System

Software Development Plan

CPSC 546 – Modern Software Management
Dr. Christopher Ryu
Department of Computer Science
California State University, Fullerton

Team Information

Group name: Team OC

Group leader: Alex Liao

No.	***Member Name***	***Email***	***Contribution***
1	Alex Liao	aliao@csu.fullerton.edu	100%
2	Danny Zhang	danny.zhang@csu.fullerton.edu	100%
3	Eric Lyv	lyveric@csu.fullerton.edu	100%
4	John Crisanto	jcrisanto1987@csu.fullerton.edu	100%
5	John Mai	jmai1011@csu.fullerton.edu	100%
6	Kevin Tran	kevintt@csu.fullerton.edu	100%

Executive Summary

This document is a software development plan for the vLearn Course Management System to be developed by Team OC Inc. The new course management system is intended to replace California State University, Fullerton's (CSUF) current virtual learning environment. CSUF currently uses Moodle as its virtual learning environment. Both students and school faculty members rely heavily on Moodle as a tool for course management and to communicate with each other. Recently, CSUF has passed an initiative to replace its current virtual learning environment (Moodle) with a new system that takes Moodle's most useful features and adds its own new capabilities and features.

The significance of this document is to draft out and present the vLearn Course Management System's development plan to be approved by CSUF administrators. In this document, excluding the table of contents, team introduction, and references, there are seven parts to our software development plan. These seven parts include: the introduction, project goals and scope, basic and proposed requirements, software development process model, work breakdown structure (WBS), project schedule, and time, effort and cost estimates.

We start this document with an introduction to provide the readers with background information on what a virtual learning environment is and the motivation behind building the vLearn Course Management System. We will then present the goals and scope that we have set forth and identified. Following that section is the basic and proposed requirements where all of the system features of the vLearn Course Management System will be presented. After that, we will describe the software development process model that we have selected to use in developing the system. The work breakdown structure (WBS) and project schedule will map out the work that needs to be done and will include the timeline of the overall project. The last section of the document consists of the estimates made for time, effort, and cost.

It is beneficial for anyone interested in the system and the software development plan to review this document in its entirety as it maps out the project's vision and scope, system features, and all relevant time, effort, and cost estimates. In this document, we outline our plan of how we intend to create the vLearn Course Management System.

Table of Contents

Table of Contents

1.0 Introduction

A virtual learning environment (VLE) is a web-based learning and course management system. VLEs typically group users into different roles, such as teachers, students, site administrators, etc. Users in these roles can interact with the VLE in different ways. For example, teachers will have certain permissions to upload lecture slides, collect homework assignments, and assign grades. Students will be able to review lecture slides, turn assignments in online, and view their grades. Typically, VLEs will have similar main core features, such as the ability to create courses and users, communication features, and a way to plan a course. Additional features may vary from system to system. However, these additional features typically contribute to improving the learning experience for its users.

VLEs allow students, instructors, and anyone else using the system to interact with the course outside of conventional classroom walls. Education can continue from any physical location, as long as there is an internet connection. The use of a VLE allows a university (or any other school) to extend the course beyond the classroom. Instructors (and schools) may even choose to teach a course entirely through the VLE.

California State University, Fullerton (CSUF) currently uses Moodle as its virtual learning environment. Moodle is a free and open source course management system, similar to Blackboard Learn. Moodle includes the core features of a VLE, as well as many additional features that go beyond the core features. Students can view the courses they are enrolled in at CSUF, and teachers can modify the class pages for the courses they are teaching. CSUF uses Moodle as its VLE for not only on-campus students, but off-campus distance learner students as well. It currently plays an integral role for its users, including students, professors, department chairs, and CSUF administration. However, Moodle is not a perfect system either. Because Moodle is developed by a separate company, it lacks specific customizations unique to CSUF. As a result, Moodle contains certain features that are not used at all by CSUF users. It may be laid out in a way that CSUF administration feels is not user friendly. Moodle may also be lacking features that certain CSUF academic departments would find useful.

In our analysis of Moodle, we determined that while Moodle is a useful and powerful course management system and VLE, it has its shortcomings. The shortcomings mainly relate to the user experience when using Moodle. Certain features are not displayed effectively, which results in users not being able to fully take advantage of them. Instructors can get easily overwhelmed with the many different options and features available to them as they manage their courses. Students may not have interest in clicking through every page of Moodle to identify or find a specific feature. This results in a poorer user experience when using Moodle. Moodle also lacks effective communication features, which may affect a user’s perception of the system. While it offers strong collaboration tools, such as Wikis, forum posts, and blog entries, it does not have clear and straightforward ways to communicate with other students and instructors.

CSUF has expressed an interest in developing a new course management system to replace Moodle. We believe that we can work with CSUF stakeholders to develop a new course management system that is specific to their needs. We will include key features from Moodle that CSUF stakeholders feel are important. In addition, we will develop custom features to accommodate the needs of CSUF users. CSUF has chosen to embark on this investment on fostering an improved learning environment through an improved VLE; we believe that we can be a part of that path. In working together with CSUF, we truly believe we can create a comprehensive and feature-rich system that satisfies the needs of CSUF users, ultimately improving the learning environment that a VLE provides to its users.

By our definition, an ideal learning environment will consist of several key elements. An ideal learning environment starts with efficient communication. We plan on developing a system that improves communication not only between students and professors, but students and other students as well. We believe that students communicating in an effective manner can lead to a more collaborative learning environment.

2.0 Project Goals and Scope

The goal of our project is to create a new and fully functioning course management system that provides users with a positive experience. We also hope to foster and promote student learning through the system. With a modern system that takes advantage of new technologies, we hope that students and instructors will feel that learning, studying, teaching, and communication/interaction is more efficient. In order to meet these goals, we address items such as communication, interfaces, and ease of use. We have created the following objectives listed below that we believe will accomplish this goal.

Objective 1: Improve communication between students and professors, especially for distance learning

By providing more effective and straightforward methods of communication using our system, we hope that users will feel that they are getting more value out of the system. Communication between students and instructional faculty (including professors, TAs, etc.) can be unorganized, especially for instructors, who may be receiving many emails per day from different classes they teach. By improving communication features, we hope that students and instructors feel that they are now able to communicate efficiently and effectively. When emails get answered, instead of being lost in a sea of emails, we believe that our users will see this as an improvement in communication, resulting in a more positive experience in using a course management system. In order to meet this objective, we will be refining the communication features from both the students' perspective and the instructional faculty's perspective. In meeting this objective, it is likely that we will be adding new features to assist with communication that were not previously available.

Objective 2: Create a user friendly and robust interface

We believe that a user's experience with any interactive system is quickly determined just by its ease of use and robustness. In the past, systems have used dated layouts with designs that were not as visually appealing when compared to other websites. Other systems may use dated code that does not work well on mobile devices, thereby limiting how users can access that system. In order to give our users a positive experience when using our system, we believe that creating a clean and user friendly design is important. A modern design and layout will allow users to use our system on whatever device they choose, be it a mobile device, tablet, or computer. A clean and user friendly design will make it easy for users to navigate and use the system to its fullest capability. We believe that this objective, while purely cosmetic, will lead to users having a positive experience in using our system.

Objective 3: Make it easier for users to use the system more effectively

This objective ties in with Objective 2, but takes it one step further. We understand that there are users who will use our system with varying levels of tech savviness. In creating a user friendly and robust interface, we aim to make our system straightforward to use, regardless of how tech savvy a user is. Not only will our system feature new and updated layouts, we will be adding system features to make the system more intuitive to use. When the site is clean and straightforward in its design, users can easily find features that they may not have known existed in the first place. We believe that by reducing the number of clicks and pages required to accomplish a specific task, users will have a more positive view of the system and be more inclined to use it. When features and pages are laid out in a straightforward yet visually appealing manner, users may feel more comfortable using the system, allowing them to view our system in a positive light.

Objective 4: Create a system rich in useful features

With this objective, we hope to create a system that is rich in features, but not bloated with them. Certain systems may be full of features, which is a great marketing talking point. However, there is no use having hundreds of features with only 10 or fewer actually being used. We consider systems like that to be bloated and confusing. Users, namely instructors, may feel overwhelmed with all the different features and be afraid of trying new things in the system. In order to give users a better and more positive experience, we aim to find a use for each system feature and functionality we add. This ensures that the system remains clean and relevant.

Another goal of this project is to create an optimized system from an IT and implementations perspective. It is our goal to have our system take advantage of efficient coding optimized for newer hardware. Although this is more of a backend goal, we believe implementation is equally important to the user experience. An un-optimized system will very likely lead to a poor user experience anyway, so we felt it necessary to specifically state this as a project goal.

Objective 5: The system should have a smooth and simple implementation

A smooth and simple integration should go hand-in-hand with an optimized system. If the system is truly optimized and well coded, it should be smooth and simple to implement and install; if a system is smooth and simple to install, it is possible that the system is optimized and well coded. We believe that meeting this objective will help us accomplish our goal of creating an optimized system. As developers work on the system, they will refer back to the goals and objectives of our project. Clearly stating this as an objective for accomplishing our goal will help keep it in the developers' minds as they work on the system.

Objective 6: Low cost of maintenance

A low cost of maintenance will make it easier for IT and any maintenance staff. Similar to Objective 5, we hope that explicitly stating this objective will help developers remember our goal of having an optimized system. In order to meet this objective, the system should, in theory, be optimized and efficient in its coding practices. By meeting this objective, the system should be easy to maintain and update, which would require optimized coding that may take advantage of new hardware.

Objective 7: Low resource utilization

Having a system with low resource utilization will require the system be efficient and take advantage of the hardware it's installed on. We consider resource utilization to encompass any CPU, memory, storage, and network usage. To meet this objective, the system should be coded in a way that is efficient and optimized. This should allow the system to remain quick and efficient for users, since optimized code will yield performance improvements and benefits.

End Result

The end result of our project should be to have a course management system that aligns with our goals and objectives defined above. It will provide users with a more positive experience through improvements in visual design as well as increased functionality. The system will keep some existing and useful system features from Moodle, but will also include new features that we have developed that are intended to further improve the user experience. Existing features that were not heavily used will not be included either, keeping the system bloat-free and easy to use. The resulting system should also improve communication between students and professors, fostering an improved learning environment. This will allow both students and professors to feel they are getting the most value out of our system. To branch off of communication, our resulting system will improve organization as well. Professors will be able to display information in a more straightforward fashion and students will be able to find what they are looking for more efficiently.

Ultimately, our end result will be a system that satisfies our goals and objectives while integrating key system features and requirements found in this document.

3.0 Basic and Proposed Requirements

3.1 Introduction

A critical aspect of the system is to provide an intuitive approach for instructors to easily manage and setup their courses. This course management system has all the useful features available on current version of Titanium as well as additional features we believe will improve student's academic success.

3.2 Purpose

To provide a detailed description of the functional and nonfunctional requirements of Team OC management system version 1.

This document reflects upon 7 key objectives:

1. Improve communication between students and professors
2. User friendly and robust
3. Make it easier for users to use the system more effectively
4. Create a system rich in useful features
5. Smooth and simple implementation
6. Low cost of maintenance
7. Low resource utilization

3.3 System Features

- Administrative **Features**
- Student **Features**
- Forum and Text Editing **Features**
- Interface **Features**
- Advanced Communication **Features**
- Data Management **Features**

ID	**Functional Requirements**
F1	This system shall be a workflow oriented application which consists of customizable course management features
F2	This system shall provide collaborative tools for students and instructors
F3	This system shall process user requests for content delivery
F4	This system shall provide a user interface

F5	This system shall provide multiple channels of communication
F6	This system shall provide users to easily export course data
ID	**Non Functional Requirements**
NFR01	This system must provide user support and guidance
NFR02	This system must be able to handle multiple users
NFR03	This system shall keep track of user and system activity
NFR04	This system shall update to the latest software available
NFR05	This system shall be able to integrate with third party applications
NFR06	This system shall be secured through many forms of authorizations
NFR07	This system shall offer digital storage
NFR08	This system must comply with university policies
NFR09	This system shall be portable
NFR10	This system shall perform optimally under normal conditions

3.4 Functional Requirements

F1 - Administrative

- Moodle shall allow instructors to enroll students into the course.
- Moodle shall allow instructors to remove students from the course.
- Moodle shall allow instructors to post resources for students to access.
- Moodle shall allow instructors to send emails directly to students.
- Moodle shall allow instructors to change the look-and-feel of course pages.
- Moodle shall allow instructors to “roll-back” settings.

F2 - Student Performance: Assignments, Quizzes, and Grades

- Moodle shall allow instructors to post assignments.

- Moodle shall allow students to submit assignment files during the time period set by the instructor.
- Moodle shall allow instructors to grade assignments and provide feedback.
- Moodle shall allow instructors to post quizzes.
- Quizzes shall only be accessible during the time period set by the instructor.
- Moodle shall allow instructors to grade quizzes and provide feedback.
- Moodle shall allow students to view their grades and feedback on assignments and quizzes.

F3 - Forum and Text Editing

- Moodle shall allow instructors and students to access a class forum.
- Moodle shall allow instructors and students to post a new thread on a class forum.
- Moodle shall require a title for class forum threads.
- Moodle shall allow instructors and students to post to an existing thread on a class forum.
- Moodle shall allow users to subscribe to a forum thread.
- Moodle shall allow users to unsubscribe from a subscribed forum thread.
- Moodle shall allow provide a text editor for instructors and students.
- Moodle shall provide text formatting tools to the user.
- Moodle shall allow users to modify posted text.
- Moodle shall allow teachers to disable the ability for student’s to modify their forum post after it has been published.

F4 - Interface

- The Moodle system shall provide an improved dashboard display with the use of widgets.
- The Moodle system shall provide users the ability to format displayed text and icons to improve font readability.
- The Moodle system shall display popup notifications on new assignments, deadlines, forum posts, and private messages.
- The Moodle system shall allow users the option to use quick command keys to directly access key pages; for example, hotkeys.
    - User Hotkeys
        - ‘R’ - Access recent updates page
        - ‘G’- Grades page

F5 - Communications

- Moodle shall allow instructors to broadcast live web seminars.
- Moodle shall allow instructors to send messages to their students in a single and group context.
- Moodle shall allow users to communicate through live chat.
- Moodle shall allow users to email instructors and other students.

F6 - Export User data

- Moodle shall allow the user to have the option to export previous assignments, blog post, and files.

- Moodle shall allow the user to have the option to export all user/course files.

3.5 Non-functional Requirements

NFR01 - User Accessibility

- Language support
    - Users shall be able to view and use Moodle in a number of supported languages.
- Tooltip and guided access for new users
    - Moodle shall display tooltips and guided access to users by default.
    - Moodle shall allow the user to hide the tooltips and guided access features.
- Speech support
    - Moodle shall provide speech support as an accessibility feature.

NFR02 - Scalability and Maintainability

- The Moodle system is designed to handle a large amount of active users.
- The Moodle system can maintain system functionality if there's an occurrence in instructor errors or administrative errors.

NFR03 - Monitoring

- Moodle system shall allow instructors/administrators to monitor student's activity.

NFR04 - Maintenance

- The Moodle system may undergo maintenance for software updates, security patches and system checks.

NFR05 - System Integration

- Moodle may allow instructors to easily integrate 3rd party applications.
    - I.e. Google apps, Dropbox and/or Slack.

NFR06 – Security

- Moodle shall require users to log in with a username and password.
- Moodle shall allow users to sign in to their account with a password that is sent to their mobile phone via SMS.
- Moodle usernames and passwords shall be encrypted per ISO/IEC 18033-3.
- Moodle shall only authorize users currently enrolled with the university.
- Moodle shall allow users to remain logged in for up to 14 days.
- Moodle shall require users to reset their password if it has not been reset within the past 12 months.
- Moodle shall allow users to sign off
- Moodle shall prevent an IP address from logging in for 30 minutes after 5 failed attempts.

NFR07 - Data Management

- Moodle shall provide users storage space for digital media files.

NFR08 - Standards Compliance

- The Moodle system shall be compliant with university and classroom policies.

NFR09 - Portability

- The Moodle system can be access through a mobile device or a web based application.
- The Moodle system shall allow users with offline access.
    - Offline mode grabs and caches Moodle pages with the most recent updates and course content; once back online, any posts, messages and/or files will be posted, sent/received

NFR10 - Performance

- The system shall perform optimally while multiple users are actively online.
- The system shall have a response time of no longer than 0.1 seconds.
- The system shall not be offline for longer than 30 minutes, unless it is in the event of a scheduled maintenance.

4.0 Software Development Process Model

For the purposes of this proposed project, Team OC will utilize Scrum and Extreme Programming (XP) development processes with an emphasis on well-defined requirements. Both XP and Scrum are Agile models that have the ability to adapt to unpredictability and change throughout the software development life-cycle (SDLC). This is accomplished through iterative work that focuses on the feedback loop. Team OC considers the delivery of working software and customer satisfaction as measurements for project success, which is why it decided to incorporate these two processes as part of its software development process model. Also the incremental delivery methodologies of Scrum and XP will enable the team to produce a high quality product with minimal risk. Furthermore, each process emphasizes collaboration through activities such as daily Scrum meetings and planning games, which will help keep the team aligned with the product vision throughout the entire SDLC. These processes are also ideal for handling future enhancements to the product, which may be necessary as the needs of students and instructors evolve over time. Scrum and XP will enable the team to elicit feedback from these user groups to develop optimized solutions that are more likely to be integrated smoothly.

Since the requirements for the desired system are provided at a high level, it made a great deal of sense to select Scrum and XP as the appropriate process models to utilize for this project. As mentioned, Scrum and XP processes are centered around timeboxes or iterations in which a working piece of software is delivered at the end of each iteration. For each iteration, the high-level requirements are broken down into smaller, simpler, and more manageable pieces, which in turn are prioritized by the stakeholders. The advantage of this is that the cost related to the time and effort spent drafting and breaking down the system requirements can potentially be reduced since specification of requirements is done in increments and change in requirements can sometimes be inevitable. Specifying all the requirements ahead of time does not seem prudent because if a system functionality is deemed insignificant by the stakeholders, it would place low on the priority list and no resources would been expended on its behalf. With the feedback loop in place, it also helps ensure that the functionalities with the highest priorities are delivered first and only the functionalities that are desired by the stakeholders will be implemented. With the utilization of both the Scrum and XP process models, Team OC will also be able to focus on providing high customer satisfaction.

The overall development process will be comprised of four main phases: pre-game planning and staging, development, and release. Each increment or timebox, which is also referred to as a sprint, will include these four phases and at the end of each sprint, a working piece of software will be delivered for feedback. Each subsequent release will provide additional features and functionalities to the prior releases until the final product is completed. To help produce a high-quality product and reduce technical debt, Team OC, as part of its overall software development process, will implement key practices, including, but not limited to: class, responsibilities, and collaboration (CRC) cards, pair programming, test-driven development, continuous integration, and other key practices that will be discussed in later sections. The following section describes the work products and various activities that will be performed within the four phases in the development process.

5.0 Work Breakdown Structure

Level 1	**Level 2**	**Level 3**	**Level 4**
1 Course Management System	1.1 Pregame	1.1.1 Planning	1.1.1.1 Vision, Scope, Resource Overview
			1.1.1.2 Domain Analysis
			1.1.1.3 Deconstruct Moodle
			1.1.1.4 Stakeholder Interviews
			1.1.1.5 Create User Stories
			1.1.1.6 Refine User Stories and Create Product Backlog
			1.1.1.7 Estimate and Prioritize Tasks
			1.1.1.8 Risk Analysis
			1.1.1.9 Scrum Training
			1.1.1.10 Change Management
		1.1.2 Staging	1.1.2.1 Prioritize Requirements and Refine Product Backlog (Sprint Planning Meeting)
			1.1.2.2 Create Tasks from User Stories (During Sprint Planning Meeting)
			1.1.2.3 Review Tasks + Create Sprint Backlog (During Sprint Planning Meeting)
			1.1.2.4 Architectural Design
	1.2 Development	1.2.1 Planning	1.2.1.1 Estimate Tasks
			1.2.1.2 Schedule Milestones
		1.2.2 Design	1.2.2.1 Architecture Vision
			1.2.2.2 Iteration Modeling

		1.2.3 Implementation	1.2.3.1 Pair Programming
			1.2.3.2 Sprint Burndown Chart
			1.2.3.3 Refactoring
		1.2.4 Testing	1.2.4.1 Unit Testing
			1.2.4.2 Integration Testing
			1.2.4.3 Regression Testing
		1.2.5 Review	1.2.5.1 Sprint Review
			1.2.5.2 Sprint Retrospective
			1.2.5.3 Identify More Requirements
			1.2.5.4 User Acceptance Testing
	1.3 Release	1.3.1 Deployment	1.3.1.1 Ship Deliverables
			1.3.1.2 QA Testing
		1.3.2 Documentation	1.3.2.1 System Documentation
			1.3.2.2 User Manuals
		1.3.3 Training	1.3.3.1 User Training
		1.3.4 Maintenance	1.3.4.1 Bug Tracking
			1.3.4.2 Refactoring
			1.3.4.3 Health Monitoring
			1.3.4.4 Software Support

6.0 Project Schedule

In order to meet our end results and deliver the system on time, we have created a project schedule that will guide our work. We have included a task list below, as well as a Gantt chart to provide a visual overview of our project schedule. The project is divided into three major phases: pregame, which consists of planning and development, development, which consists of three iterations, and release.

The pregame phase is set to last 6 weeks, to be divided evenly between planning (3 weeks) and staging (3 weeks). Planning will lay the groundwork for the Scrum and XP process and will include training. In the planning phase, all team members will also review the vision, scope, and resources available to solidify each team members' understanding of the project. Stakeholders will be interviewed and the current Moodle system will be deconstructed and analyzed. At the end of the planning phase, the product backlog should be complete, management should approve the project, and resources should be secured. The staging phase will begin next. The majority of the work done in this phase is designing the software architecture. Team members will also refine the product backlog and begin creating tasks based off user stories and create the sprint backlog. At the end of the staging phase, the architectural design should be created along with the sprint backlog.

The development phase is set to last 90 days, with each sprint lasting 30 days. The first iteration will focus on core functions. The second iteration will focus on the quiz and forum components of vLearn. The third iteration will focus on the look and feel, real-time communications, and settings. At the end of each sprint, working and testing software should be delivered to the customer.

The final phase is the release phase, which is set to last 30 days. During the release phase, team members will begin developing a training program. We will also gather feedback from the end users to see if there are any particular features or components that need to be added, removed, or adjusted. This is to better adjust the system to the end users' needs. Based on the feedback received, team members may or may not modify features. However, there must be compelling reason to do so. Documentation will then be created once final modifications are complete. The team will assist with implementation and setting up the vLearn system on the customer's servers. Following implementation and setup, we will provide on-site training as well. At the end of the release phase, the customer will have received a complete and working vLearn system, as well as documentation and hands-on and in-person training.

Task List

Task Name	Duration	Start	Finish	Predecessors
vLearn Project	**150 days**	**Mon 5/8/17**	**Fri 12/1/17**	
Pregame	**6 wks**	**Mon 5/8/17**	**Fri 6/16/17**	
Planning	**3 wks**	**Mon 5/8/17**	**Fri 5/26/17**	
Vision, Scope, Resource Overview	2 days	Mon 5/8/17	Tue 5/9/17	
Domain Analysis	1 day	Mon 5/8/17	Mon 5/8/17	
Deconstruct Moodle	2 days	Wed 5/10/17	Thu 5/11/17	4
Stakeholder Interviews	3 days	Wed 5/10/17	Fri 5/12/17	4
Create User Stories	4 days	Mon 5/15/17	Thu 5/18/17	7
Refine User Stories and Create Product Backlog	3 days	Fri 5/19/17	Tue 5/23/17	8
Estimate and Prioritize Tasks	2 days	Mon 5/22/17	Tue 5/23/17	9FF
Risk Analysis	5 days	Mon 5/15/17	Fri 5/19/17	
Scrum Training	2 days	Wed 5/24/17	Thu 5/25/17	10
Change Management	3 days	Wed 5/24/17	Fri 5/26/17	12FF
Product Backlog Completed	0 days	Fri 5/26/17	Fri 5/26/17	
Management Approval	0 days	Fri 5/26/17	Fri 5/26/17	
Resources Secured	0 days	Fri 5/26/17	Fri 5/26/17	
Staging	**3 wks**	**Mon 5/29/17**	**Fri 6/16/17**	**3**
Prioritize Requirements and Refine Product Backlog	5 days	Mon 5/29/17	Fri 6/2/17	
Create Tasks from User Stories	3 days	Mon 6/5/17	Wed 6/7/17	
Review Tasks and Create Sprint Backlog	4 days	Thu 6/8/17	Tue 6/13/17	19
Architectural Design	10 days	Mon 6/5/17	Fri 6/16/17	
Architectural Design Created	0 days	Fri 6/16/17	Fri 6/16/17	
Sprint Backlog Created	0 days	Fri 6/16/17	Fri 6/16/17	
Development	**90 days**	**Mon 6/19/17**	**Fri 10/20/17**	**2**
Iteration I (core functions)	30 days	Mon 6/19/17	Fri 7/28/17	
Working and tested software	0 days	Fri 7/28/17	Fri 7/28/17	
Iteration II (quizzes and forum)	30 days	Mon 7/31/17	Fri 9/8/17	25
Working and tested software	0 days	Fri 9/8/17	Fri 9/8/17	
Iteration III (look and feel, real-time communication, settings)	30 days	Mon 9/11/17	Fri 10/20/17	27
Working and tested software	0 days	Fri 10/20/17	Fri 10/20/17	
Release	**30 days**	**Mon 10/23/17**	**Fri 12/1/17**	**24**
Develop Training Program	5 days	Mon 10/23/17	Fri 10/27/17	
Gather Feedback	3 days	Mon 10/30/17	Wed 11/1/17	32

Modify features based on feedback	3 days	Thu 11/2/17	Mon 11/6/17	33
Create documentation	10 days	Tue 11/7/17	Mon 11/20/17	34
Implementation and setup	3 days	Thu 11/23/17	Mon 11/27/17	35FS+2 edays
Training	4 days	Tue 11/28/17	Fri 12/1/17	36
System delivered to customer	0 days	Fri 12/1/17	Fri 12/1/17	
Documentation complete	0 days	Fri 12/1/17	Fri 12/1/17	
Training program created	0 days	Fri 12/1/17	Fri 12/1/17	

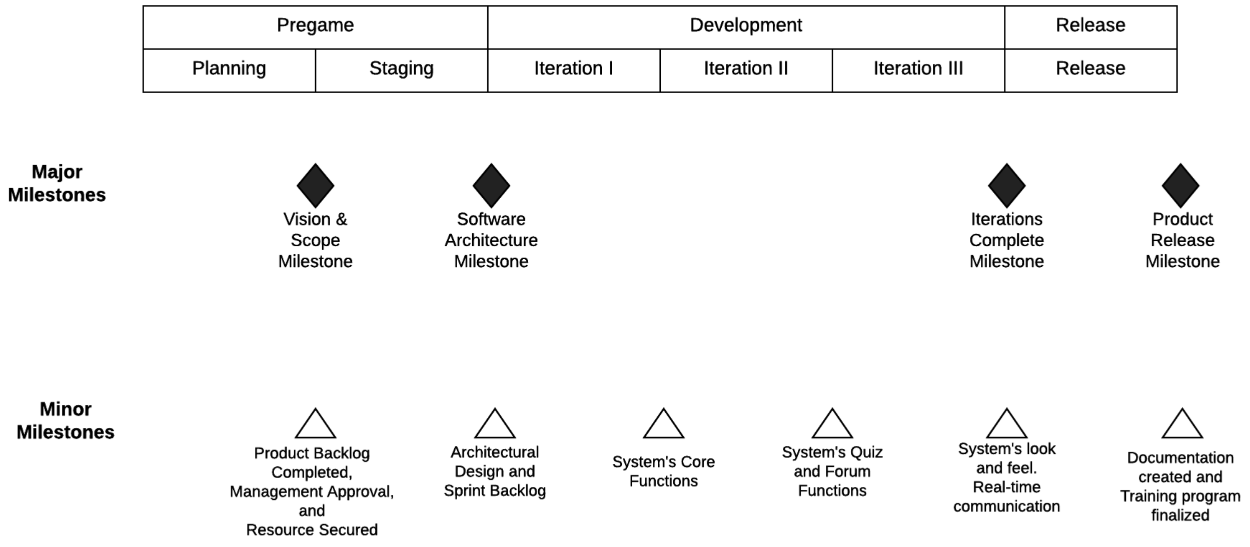
Pregame
Development
Release
Planning
Staging
Iteration I
Iteration II
Iteration III
Release
Major Milestones
Vision & Scope Milestone
Software Architecture Milestone
Iterations Complete Milestone
Product Release Milestone
Minor Milestones
Product Backlog Completed, Management Approval, and Resource Secured
Architectural Design and Sprint Backlog
System's Core Functions
System's Quiz and Forum Functions
System's look and feel. Real-time communication
Documentation created and Training program finalized

Figure 6.1: vLearn Major and Minor Milestones

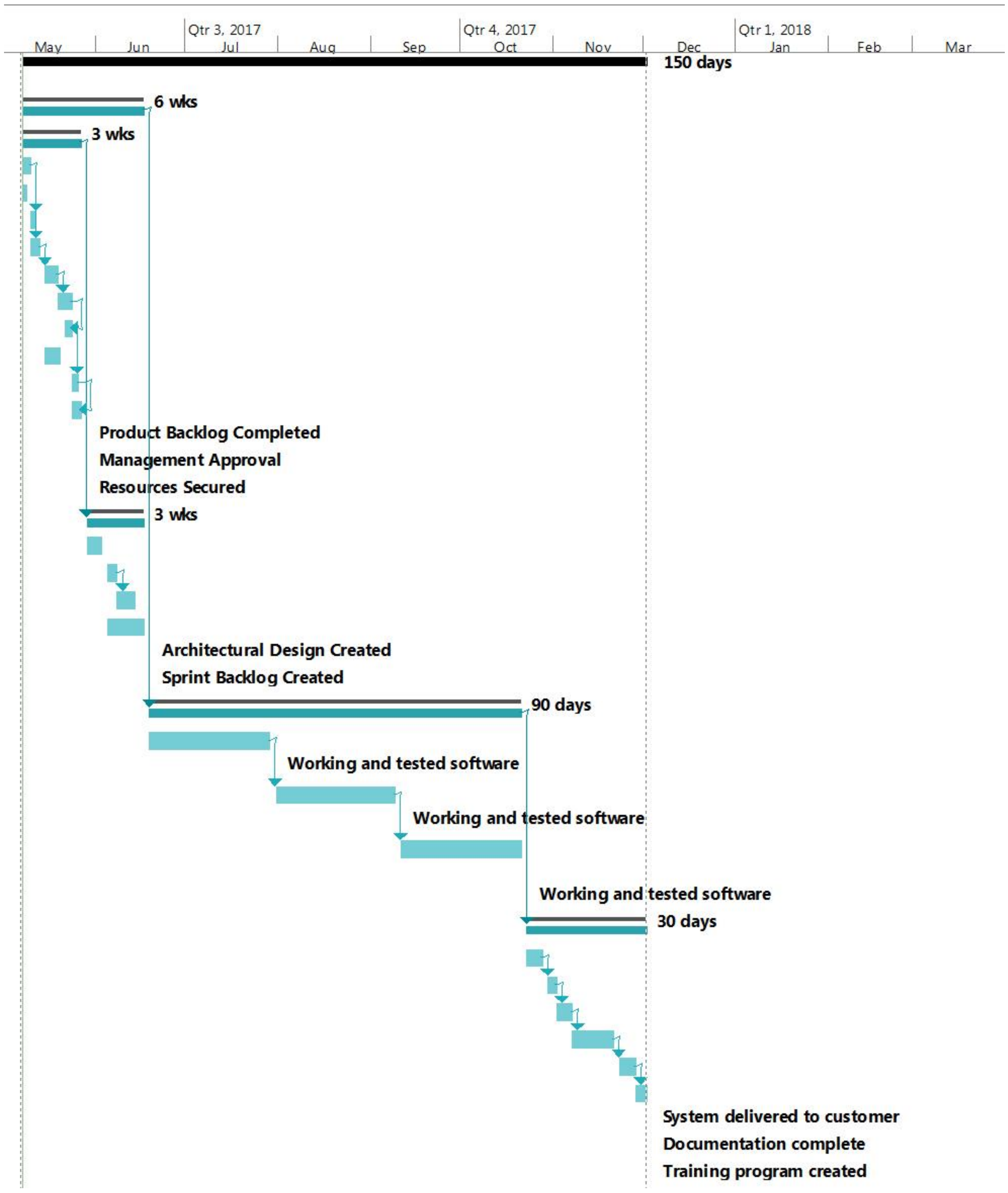

Figure 6.2: Gantt chart of vLearn Project

7.0 Time, Effort, and Cost Estimates

7.1 Overview of Time, Effort, and Cost Estimates

The team will construct a project schedule network diagram using the Precedence Diagram Method (PDM). The PDM is a project modeling technique used for scheduling activities in a project plan. We will be constructing the PDM to estimate the time, effort, and cost for the project.

The PDM is made up of nodes, which are displayed as rectangles, and the project activities are shown in those boxes. The rectangles are connected to each other with arrows in order to show dependencies. As seen in an example of a rectangle below, each node will consist of the Activity ID, Activity Description, start date, planned duration, end date, effort, and cost. The arrows show the activity dependencies depicted, usually from left to right. The PDM format that we will be using is as follows:

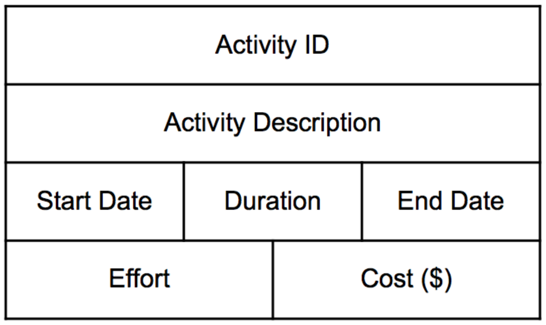

Figure 7.1: PDM Format

The PDM is filled in three phases, starting with the start date, duration, and end date in estimating the time. Then we will estimate the effort by adding in the number of engineers needed to accomplish a task into each node. Lastly, we will compute the cost based on the time and effort estimates.

7.2 Time

Process for Estimating Time

The node activities that are found in the PDM will be taken from the Work Breakdown Structure. The associated durations for each node activity is taken from the Project Schedule. To estimate the time to complete the project, we filled in the PDM with the activity id, description, duration, start date, and end date. Upon finishing this, we calculated the shortest duration in which the project can be completed using the critical path method (CPM). Since the project cannot be completed without completing the activities in the longest path, the critical path is the longest path on the PDM. We calculate our critical path by adding up all of the activity durations on the critical path. The sum of the durations on the critical path is the time estimate for the project.

Our Time Estimate

The team followed the process outlined above to estimate the time for the project. The resulting PDM after adding in the activity id, activity description, duration, start date, and end date is below.

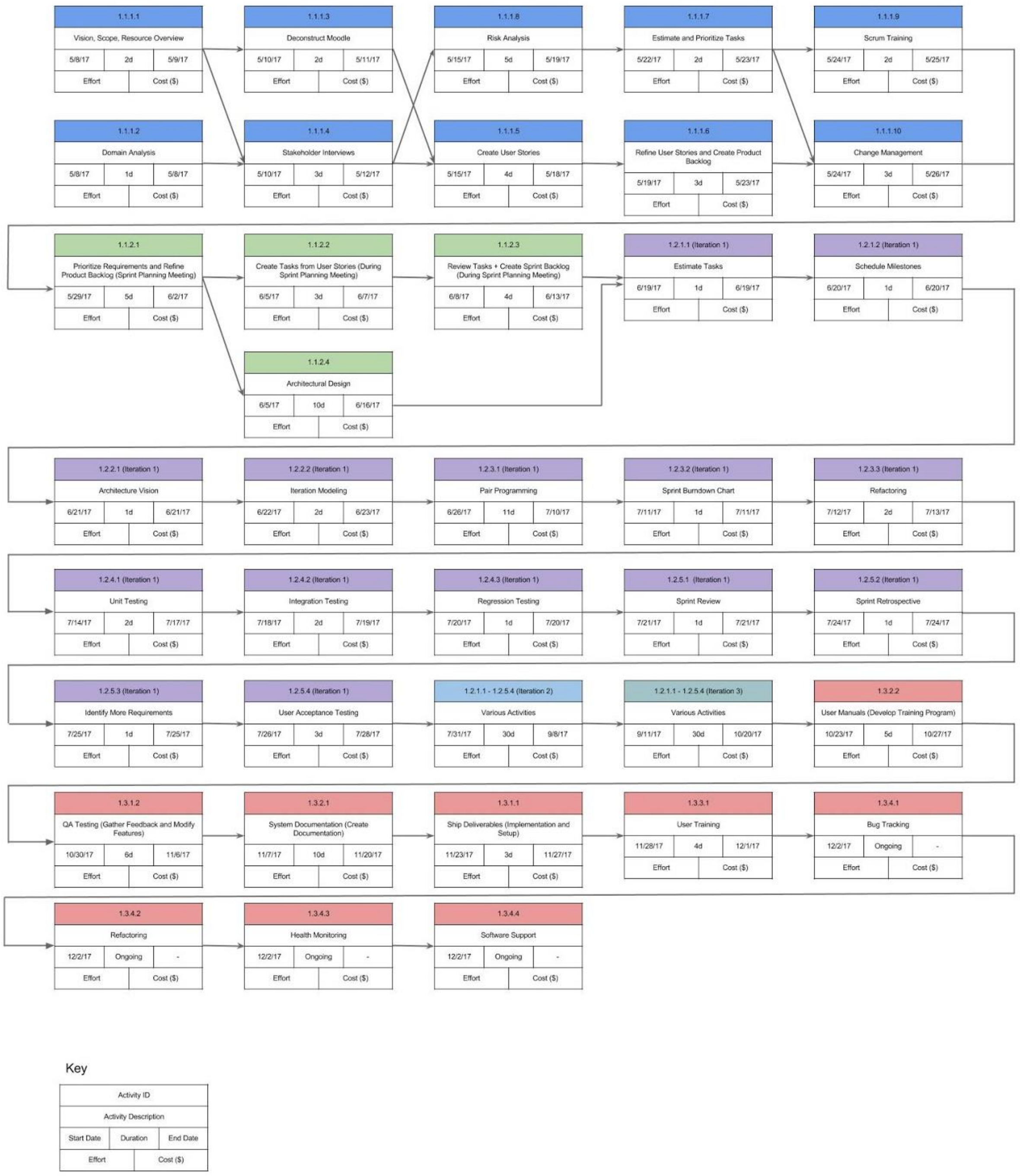

Figure 7.2: Time estimate PDM

After adding the durations to the PDM, we calculated the critical path using the CPM, which if finding the sum of the activity durations for each third level activity. Since the critical path for each iteration is the same, we performed the calculations for Iteration 1, and used the same critical path for Iterations 2 and 3. Bug tracking, refactoring, health monitoring, and software support in the maintenance activity are excluded from the time estimate as those are after the completion of the project. The sums for the third level activity durations on the critical path are below.

1.1.1	Planning	= 1.1.1.1 + 1.1.1.4 + 1.1.1.8 + 1.1.1.7 + 1.1.1.10	
		= 2d + 3d + 5d + 2d + 3d	
		= 15 days	
1.1.2	Staging	= 1.1.2.1 + 1.1.2.4	
		= 5d + 10d	
		= 15 days	
Iteration 1		**= 30 days**	
	Iteration 1 Calculations		
	1.2.1	Planning	= 1.2.1.1 + 1.2.1.2
			= 1d + 1d
			= 2 days
	1.2.2	Design	= 1.2.2.1 + 1.2.2.2
			= 1d + 2d
			= 3 days
	1.2.3	Implementation	= 1.2.3.1 + 1.2.3.2 + 1.2.3.3
			= 11d + 1d + 2d
			= 14 days
	1.2.4	Testing	= 1.2.4.1 + 1.2.4.2 + 1.2.4.3
			= 2d + 2d + 1d
			= 5 days
	1.2.5	Review	= 1.2.5.1 + 1.2.5.2 + 1.2.5.3 + 1.2.5.4
			= 1d + 1d + 1d + 3d
			= 6 days
	Iteration 1 Total	= 1.2.1 + 1.2.2 + 1.2.3 + 1.2.4 + 1.2.5	
		= 30 days	
Iteration 2		**= 30 days**	
Iteration 3		**= 30 days**	
1.3.1	Deployment	= 1.3.1.1 + 1.3.1.2	
		= 3d + 6d	
		= 9 days	
1.3.2	Documentation	= 1.3.2.1 + 1.3.2.2	
		= 10d + 5d	
		= 15 days	

1.3.3	Training	= 1.3.3.1 **= 4 days**

To calculate the critical path for the entire project, we added each third level duration estimate together. The calculation is below.

Project Time Estimate	= 1.1.1 + 1.1.2 + 3 * (1.2.1 + 1.2.2 + 1.2.3 + 1.2.4 + 1.2.5) + 1.3.1 + 1.3.2 + 1.3.3 + 1.3.4 = 15d + 15d + (3 * 30d) +9d + 15d + 4d **= 148 business days** **= 29.6 weeks** **= 7.4 months**

Based on our calculations, we estimate the time needed for our project to be 148 business days, which is 7.4 months.

7.3 Effort

Process for Estimating Effort

Team OC estimated the effort necessary to complete the project by performing the level of effort (LOE) activity to estimate the number of software engineers necessary to complete the activity. Depending on the project manager and team, some projects vary the level of effort depending on activity, while others will have the same level of effort for the activities and adjust the duration needed for the task.

Our Effort Estimate

Given the scale of the project, Team OC decided that it would be optimal to have one team of four software engineers work on developing the application. Since the effort for the activity nodes do not fluctuate in this application, the effort for all the activity nodes will all be 4 software engineers. However, for some of the activities that were done concurrently with others, we split the effort to 2 software engineers to each activity. We updated our PDM from performing the time estimate with the effort values. The resulting PDM after adding the effort values is below.

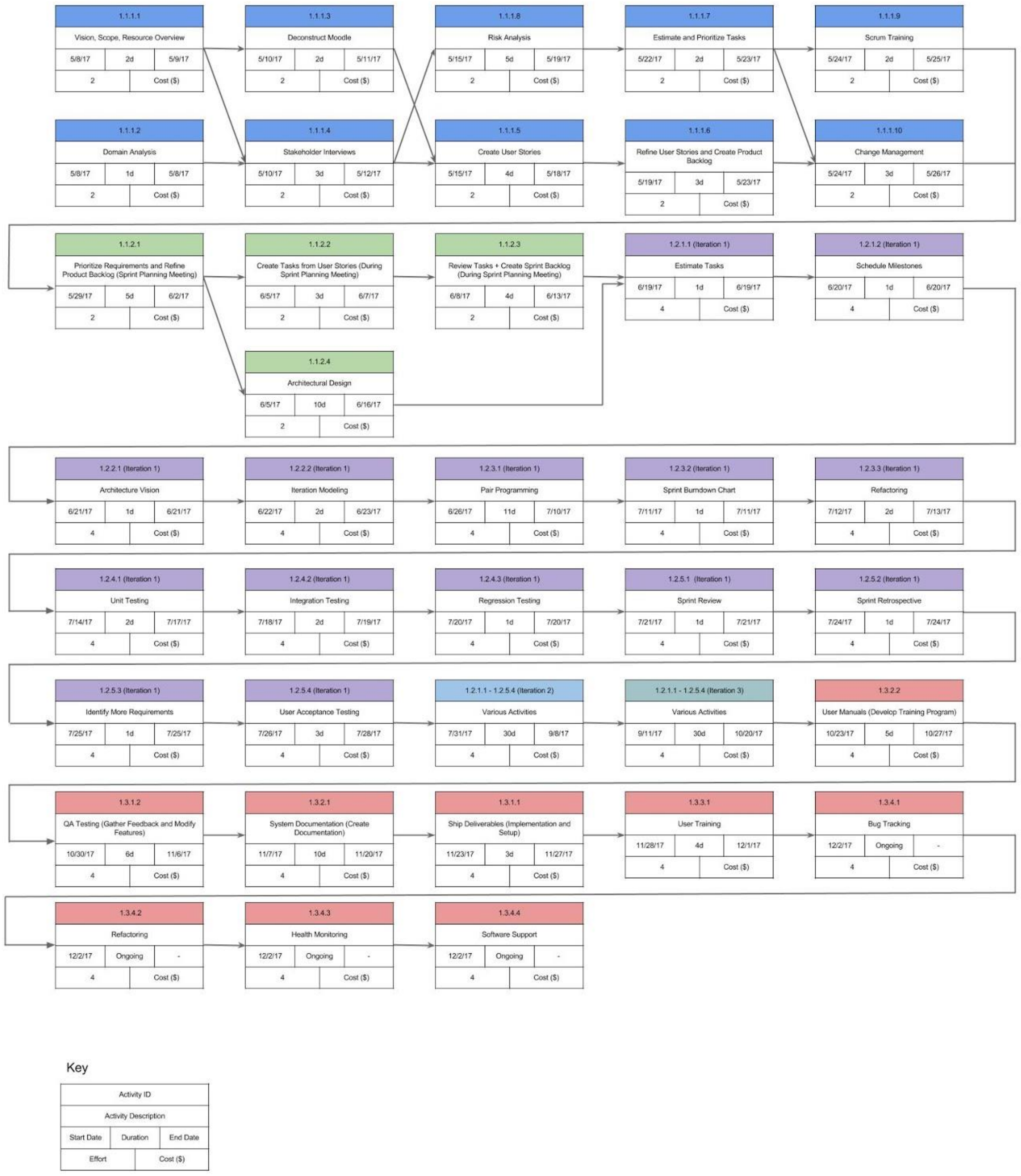

Figure 7.3: Effort estimate PDM

7.4 Cost

Process for Estimating Cost

The team has decided to estimate the cost based on the time and effort estimates. Each activity will have a cost estimate calculated by multiplying the duration, effort, and the one day labor cost for a software engineer. The estimated cost for the project is the sum of the cost estimates for all the activities.

Our Cost Estimate

For our project, we have set the price for each software engineer's labor to be $200 per day. We started by calculating the cost estimate for each activity and adding them to the PDM. The resulting PDM after adding the cost values is below.

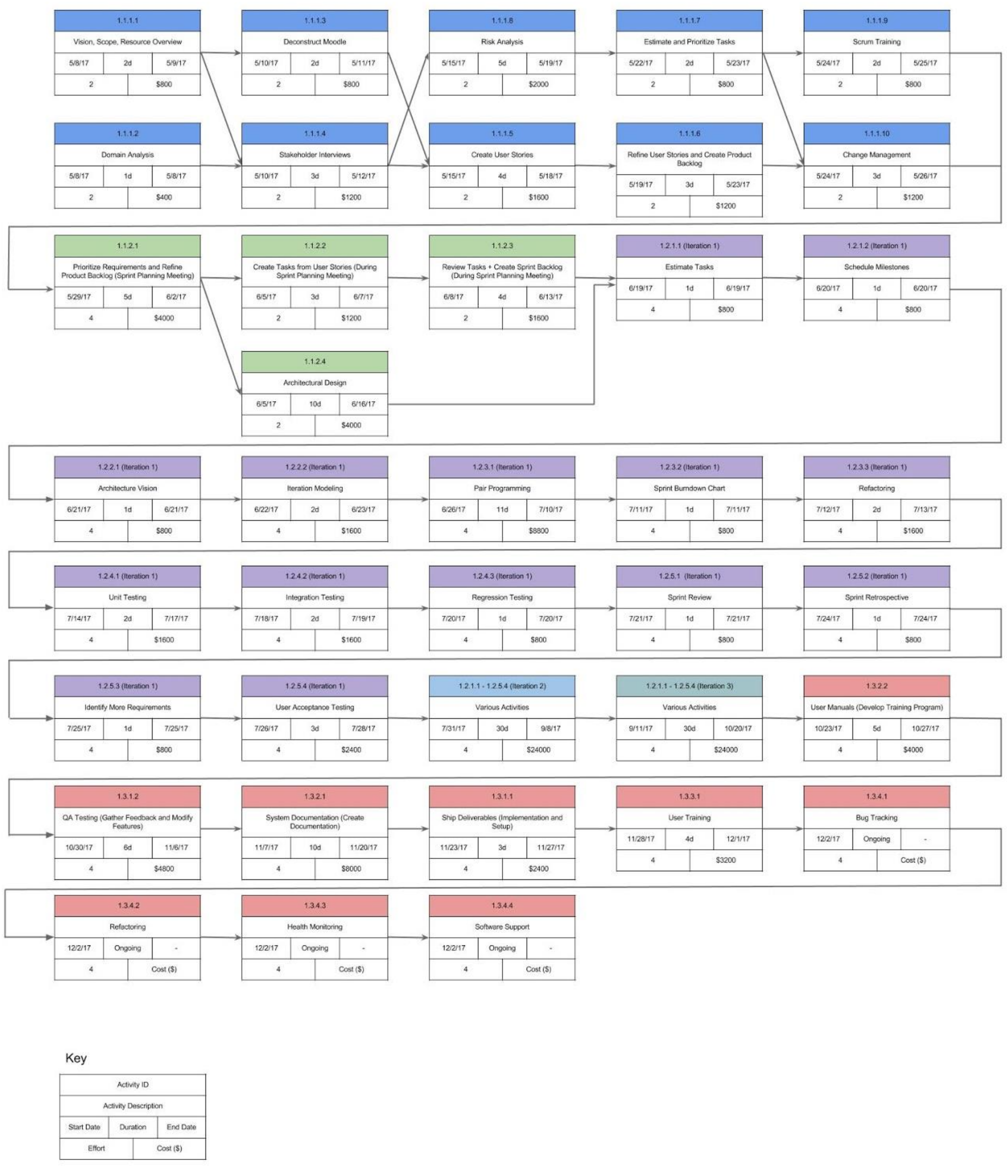

Figure 7.4: Cost estimate PDM

After adding the cost values to the PDM, we calculated the sum of the activities for each third level activity. Since the cost for each iteration is the same, we performed the calculations for Iteration 1, and used the same cost for Iterations 2 and 3. Bug tracking, refactoring, health monitoring, and software support in the maintenance activity are excluded from the cost estimate as those are after the completion of the project. The sums for the third level activities are below.

1.1.1	Planning	= 1.1.1.1 + 1.1.1.2 + 1.1.1.3 + 1.1.1.4 + 1.1.1.5 + 1.1.1.6 + 1.1.1.7 + 1.1.1.8 + 1.1.1.9 + 1.1.1.10 = \$800 + \$400 + \$800 +\$1200 + \$1600 + \$1200 + \$800 + \$2000 + \$800 + \$1200 **= \$10800**
1.1.2	Staging	= 1.1.2.1 + 1.1.2.2 + 1.1.2.3 + 1.1.2.4 = \$4000 + \$1200 + \$1600 + \$4000 **= \$10800**
Iteration 1		**= \$24000**

Iteration 1 Calculations

1.2.1	Planning	= 1.2.1.1 + 1.2.1.2 = \$800 + \$800 **= \$1600**
1.2.2	Design	= 1.2.1.1 + 1.2.1.2 = \$800 + \$1600 **= \$2400**
1.2.3	Implementation	= 1.2.3.1 + 1.2.3.2 + 1.2.3.3 = \$8800 + \$800 + \$1600 **= \$11200**
1.2.4	Testing	= 1.2.4.1 + 1.2.4.2 + 1.2.4.3 = \$1600 + \$1600 + \$800 **= \$4000**
1.2.5	Review	= 1.2.5.1 + 1.2.5.2 + 1.2.5.3 + 1.2.5.4 = \$800 + \$800 + \$800 + \$2400 **= \$4800**
Iteration 1 Total	= 1.2.1 + 1.2.2 + 1.2.3 + 1.2.4 + 1.2.5 **= \$1600 + \$2400 + \$11200 + \$4000 + \$4800** **= \$24000**	

Iteration 2		**= \$24000**
Iteration 3		**= \$24000**
1.3.1	Deployment	= 1.3.1.1 + 1.3.1.2 = \$2400 + \$4800 **= \$7200**

1.3.2	Documentation	= 1.3.2.1 + 1.3.2.2 = $8000 + $4000 **= $12000**
1.3.3	Training	= 1.3.3.1 **= $3200**

To calculate the cost for the entire project, we added each third level cost estimate together. The calculation is below.

Project Cost Estimate	= 1.1.1 + 1.1.2 + 3 * (1.2.1 + 1.2.2 + 1.2.3 + 1.2.4 + 1.2.5) + 1.3.1 + 1.3.2 + 1.3.3 + 1.3.4 = $10800 + $10800 + (3 * $24000) + $7200 + $12000 + $3200 **= $116000**

From the calculations, the estimated cost for the project is **$116000**.

8.0 References

[1] MoodleDocs (2016) [Online]. Available: https://docs.moodle.org/32/en/Main_page

[2] E. Jenett. (1972). Choosing Your Networking Techniques (1972) [Online]. Available: http://www.pmi.org/learning/library/networking-technique-cpm-pert-pdm-5737

[3] J. Nair. (2013, May 28). Calculating Level of Effort: How Do We Approach It? (2013) [Online]. Available: https://www.projectmanagement.com/articles/278837/Calculating-Level-of-Effort--How-Do-We-Approach-It

[4] M. Piscopo. (2017). Work Breakdown Structure (2017) [Online]. Available: http://www.projectmanagementdocs.com/project-planning-templates/work-breakdown-structure-wbs.html

[5] Learn About Scrum (2016) [Online]. Available: https://www.scrumalliance.org/why-scrum

[6] M. James. Scrum Methodology [Online]. Available: http://scrummethodology.com/

[7] Project Scheduling [Online]. Available:http://www.projectinsight.net/project-management-basics/project-management-schedule

Appendix A: File Index

Due to the high resolution of some of our images and files, we have attached full-sized images in the project archive/folder. They are as follows:

File Name	Location in Document	Description
pdmCost.jpg	Figure 7.4 (page 23)	Cost estimate PDM, full resolution (1613 x 1905)
pdmEffort.jpg	Figure 7.3 (page 22)	Effort estimate PDM, full resolution (1613 x 1905)
pdmTime.jpg	Figure 7.2 (page 19)	Time estimate PDM, full resolution (1613 x 1905)
projectSchedule.mpp	Task List (page 15-16)	Project file for Microsoft Project 2016, task list and Gantt chart
projectSchedule.pdf	Figure 6.1 (page 17)	Gantt chart generated by Microsoft Project 2016

If any files are not included in the archive/folder or they cannot be opened, please contact the project team leader.