Team 2 – OpenSource e-Mail Client

Project Proposal & Plan

1. Introduction

With the emergence of a geographically spanning internet in the late 1960's, so to came a need for person specific contact. Soon enough, electronic mail arrived, supported primarily by ISPs. Open source mail clients were not far behind, but the solution was followed by a variety of issues. Unreliable back-ends were covered by fancy looking views falling on the lack of direct accountability through open source to provide a careless model with a fancy front end looks. Other clients provide an eyesore to the client, misleading from the strong backend supporting the service. With neither situation optimal, the typical user will use licensed software for email.

1.1 Project Overview and Statement of Proposal

We noticed these two extremes and decided that creating a web-mail client with a simple, effective, and robust back end could be paired with a strong, responsive, and user-friendly front-end presentation. Providing an open source service which is not only user friendly but executed properly can mark the difference between true software engineers and the casual coder.

Statement of Proposal: *We propose to create a simple Open-Source online e-mail service to enable clients to display, render, and send emails with attachments from a browser using a simple and intuitive responsive design.*

1.2 Project Scope and Objectives

We would like to be able to release it to the public at the end of the project. As such, are audience is for individuals or small companies that need a a free, stable, and feature rich mail client. Our scope then will target two different audiences. The first group is made up of individuals, and the second for companies or organizations such as schools. The individuals will need a feature rich platform to compete with major companies such as Microsoft and Google. The second group on the other hand will be more focused on the security and stability of a free, open-source package.

Our main objective is to create a stable, secure, and user-friendly mail client. The next objective is to add features to the basic package such as rendering attachments, sorting, filtering, folder organization, PGP signing, and responsive design. Each of these features will improve the user's ability to organize and use the software securely and effectively.

2. Risk Management Strategy

2.1 Risk Table

*Please note that the Probability is rated by LOW (<25%), MID (50%), HIGH (>75%)

Risks	Cat.	Prob.	Impact	RMMM
Fully Functional Mail Client	PS	HIGH	3	Implement in Phases
Open-Source Release at Semester End	PS	HIGH	2	Keep a good timeline
Other Projects Running Concurrently (school)	PR	LOW	2	keep team members in the loop and flexible

Negative Impact to Developers	BU	LOW	1	create a good product
Negatively Impact Our Grade	BU	HIGH	3	document, plan, and implement consistently. No procrastinating
Isn't Well Recieved By OpenSource Customers	CU	MID	2	research the cliental
Negatively Impact Our Other Commitments	BU	MID	3	keep teammates flexible and able to help each other as needed
Server Could Become Unobtainable	DE	LOW	4	have a back up server ready
Project Could Be To Big For Experience Level	ST	MID	3	split into phases to make more manageable
Chose a Bad Dev. Cycle	PR	MID	2	choose a flexible dev cycle so changes can be made to it later on.
Browser Updates Cause Our Code Base to Fail	DE	LOW	3	make sure it works in current browsers, then patch as needed
Code Base Becomes Lost or Corrupted	TE	MID	4	using git for version control
Team Isn't Experienced Enough	ST	MID	2	more experienced team members help. Dedicate time for the learning curve.
Team Doesn't Work Together	DE	LOW	4	weekly meeting to establish communication and work out any issues
Loss of a Team Member	ST	LOW	2	keep a wiki and git so to mitigate the loss of info
Breach of Secure Information	TE	MID	3	regularly test the security
Software Bugs	PR	HIGH	2	test code before committing and use branches as needed
Team Members being Away for Meeting	ST	MID	3	take meeting minutes

Category values:

PS – Product Size Risk

BU – Business Impact Risk

CU – Customer Relations Risk

PR – Process Risk

TE – Technology Risk

DE – Development Environment Risk

ST – Risk Associated with Staff Size and Experience

Impact values:

4 – catastrophic

3 – critical

2 – marginal

1 – negligible

2.2 Discussion of Risks to Be Managed

Perhaps the most common risk that will affect most if not all real world projects that we face minimally, is customer satisfaction. Because we are not working directly with any specific customer, our only satisfaction comes from ourselves as well as Dr. Shin. In order to release as open source, however, we do need functioning software. This means that we have to work towards a product that open source customers will like, without being able to talk to them directly. Along with this, we face other unique large risks. As students our commitments run plenty. Not only could this project be far too large or require more than we have experience for, but we also risk neglecting other classes and work, adding to a large personal risk. Most remaining risks are fairly typical in a web project. We risk set backs by being unable to obtain a server or in choosing a bad development cycle, which would not be fatal, but would be a severe set back.

We also risk the instability of our own code base, rooted in either faulty coding or constant updates in the web browsing software. This is unlikely to be a huge issue, however it may be a large loss of time, requiring constant updates of our own code base as we go. Along with all this, we risk lack of proper team work. If we are unable to work together as a team to bring each piece of this project together we will be unable to succeed. This means not only each of us doing our job properly, but being able to understand what each other is doing and how our task plays a role, which is rooted in our proper communication. Worse than this would be the loss of a member, which may leave holes in our skill and experience set. While we don't all know the exact same things, we have enough overlap in our skill sets that we should be able to handle whatever we may encounter, but a loss of a member will weaken the chain.

2.3 Risk Mitigation, Monitoring, and Management Plan

This section describes what is to be done to avoid each risk (Section 2.3.1), how the team will monitor its activities to detect when a risk becomes a problem (Section 2.3.2), and what will be done for each risk if it becomes a problem (Section 2.3.3).

2.3.1 Risk Mitigation

In order to avoid these risks, our first line of defense is communication. If we maintain open and as complete communication as possible, many risks will be avoided. Ensuring that we are all happy with the product is as close to customer satisfaction as we can get. To be sure we reach our deadlines, we must ensure that if we individually need help with our task, that we ask for it. This will not only help ensure a functioning product at the end along with a proper open source release, but also help us avoid the inherent risk of negative impact upon our other classes and work. In order to avoid choosing a less appropriate development cycle, we must look into each of our choices and make the best informed decision that we can. Many risks however, cannot be avoided as they depend upon outside parties. This includes updates in browsers and lack of proper experience across our whole team. Experience issues can be eased through thorough research and learning prior to implementation, however, some issues cannot be avoided simply because they are unforeseeable. Our effort to avoid inter-group risks, such as losing a member, should consist of communication with a positive attitude. We must encourage each other and offer help when needed in order to maintain and support a friendly and successful environment.

2.3.2 Risk Monitoring

To ensure that our other classes, work, and obligations are unaffected by this project, we must monitor each other's work. While this does not mean we need to keep each other's schedule, we should ensure that each member is still focusing on outside work when appropriate while adhering to our project schedule when needed. Not focusing on other work can be as detrimental to a team member as neglecting this project could be. In order to assist our monitoring of risks, we will take several precautions. First we will use Microsoft Project for time lines and other resource progression. Our weekly meetings will ensure that we are all on the same page, promoting efficient and thorough communication. Lastly, as we begin implementing our designs, we will utilize git in order to share code knowledge and ease remote collaboration. This combination of weekly meetings and shared access ensure that experience issues as well as development will be maintained by the group as a whole.

2.3.3 Risk Management (Contingency Plans)

Should any risk grow beyond avoidance and we catch it while monitoring, it is important that we know how we should handle whatever problem may be. First, our use of git and a wiki will provide a back up for our code base and our in progress documentation which will ease the loss of a team member, the loss of the code base, or any faults our communication with one another. The possibility of a back up server could be useful in case the current one in use goes down or is otherwise compromised. As a means to ensure we have functional product, we will work in modular phases with an interactive development cycle. Not only does this guarantee we won't reach the end of the semester empty handed, but it will help prevent choosing a development method that will ultimately fail. However, our biggest risks, which may sneak up on us, come in the human factors. Communication, flexible team roles, and team-wide research and training will help recover from anything between the loss of a team member, to universal lack of experience, to other school obligations taking priority. These general plans will guide through any issues that may arise and help ease if not resolve the problems we may face.

3. Schedule

Below is a list of tasks associated with the project, a Gantt chart depicting task durations, dependencies and completion dates, and a summary of resource assignments for each task. Please note that since our requirements and dev. cycle have not been finalized the information in the section will change. It has been filled out to the best of our knowledge, but it is in general terms as our research on requirements is still in the beginning stages. A better and complete analysis of our task list and time lines will be provided in the design specification.

3.1 Task List

1. Research email standards and requirements for basic functionality.
2. Research audience for an Open-Source release.
3. Research Human Computer Interaction (HCI) for email clients.
4. Decide best development cycle based off requirements, team experience, and time frame.
5. Document the Requirements Specification.
6. Design project using UML diagrams.
7. Create User-Interface (UI) wire-frames based on UML and requirements.
8. Create a good code base focusing on the MVC modal using our UML diagrams.
9. Document the Design Specification.
10. Establish server for our developer environment.
11. Implement first phase of project (basic functionality).

12. Implement first phase UI.
13. Evaluate first phase.
14. Establish next phase of project.
15. Test code base for functionality and security.
16. Document final release.
17. Present final product.
18. Release to Open-Source clients.

3.2 Timeline Chart

Due to the size of the chart it is attached in Appendix A on pg 8.

3.3 Resource Table

Task	People	Hardware & Software	Special
1. Research Standards	Angelo, Ian	None	N/A
2. Research Audience	Jonathan, William	None	N/A
3. Research HCI	Melanie	None	N/A
4. Decide Dev-Cycle	TEAM	None	N/A
5. Document Requirements	Jonathan, Melanie, William	Google Docs	N/A
6. UML Design	TEAM	MS Visio, Google Docs, UML	N/A
7. UI WireFrames	Melanie, Jonathan	Paper, MS Visio,	N/A
8. Code Design	Angelo, Jonathan, Melanie	UML,	N/A
9. Document Des. Spec.	Jonathan, Melanie, William	Google Docs	N/A
10. Establish Server	Ian	Hardware Server, OS, TomCat	N/A
11. Implement Phase 1	Angelo, Ian, William	Server, IDE, Git	N/A
12. Implement Phase 1 UI	Melanie, Jonathan	Web Browser, UML, IDE, Git	N/A
13. Evaluate Phase 1	TEAM	IDE, Google Docs	N/A
14. Establish Next Phase	TEAM	UML, Google Docs.	N/A
15. Testing	William, Ian	IDE, Browsers, Testing Suite	N/A
16. Document Final Release	Jonathan, Melanie, William	Google Docs	N/A
17. Present	TEAM	Google Docs	N/A
18. OpenSource Release	TEAM	Git, Our Software Package	N/A

4. Project Resources

4.1 People

1. Melanie Palmer - Team Lead, UI Design, Developer and Document-er.
2. Ian Neal - Sys. Admin, Developer, and Tester
3. Jonathan Brandt - Project Design, Developer, and Document-er.
4. William Fong - Design, Developer, Document-er, and Tester
5. Angelo Luna - Code Design, Developer, and Tester

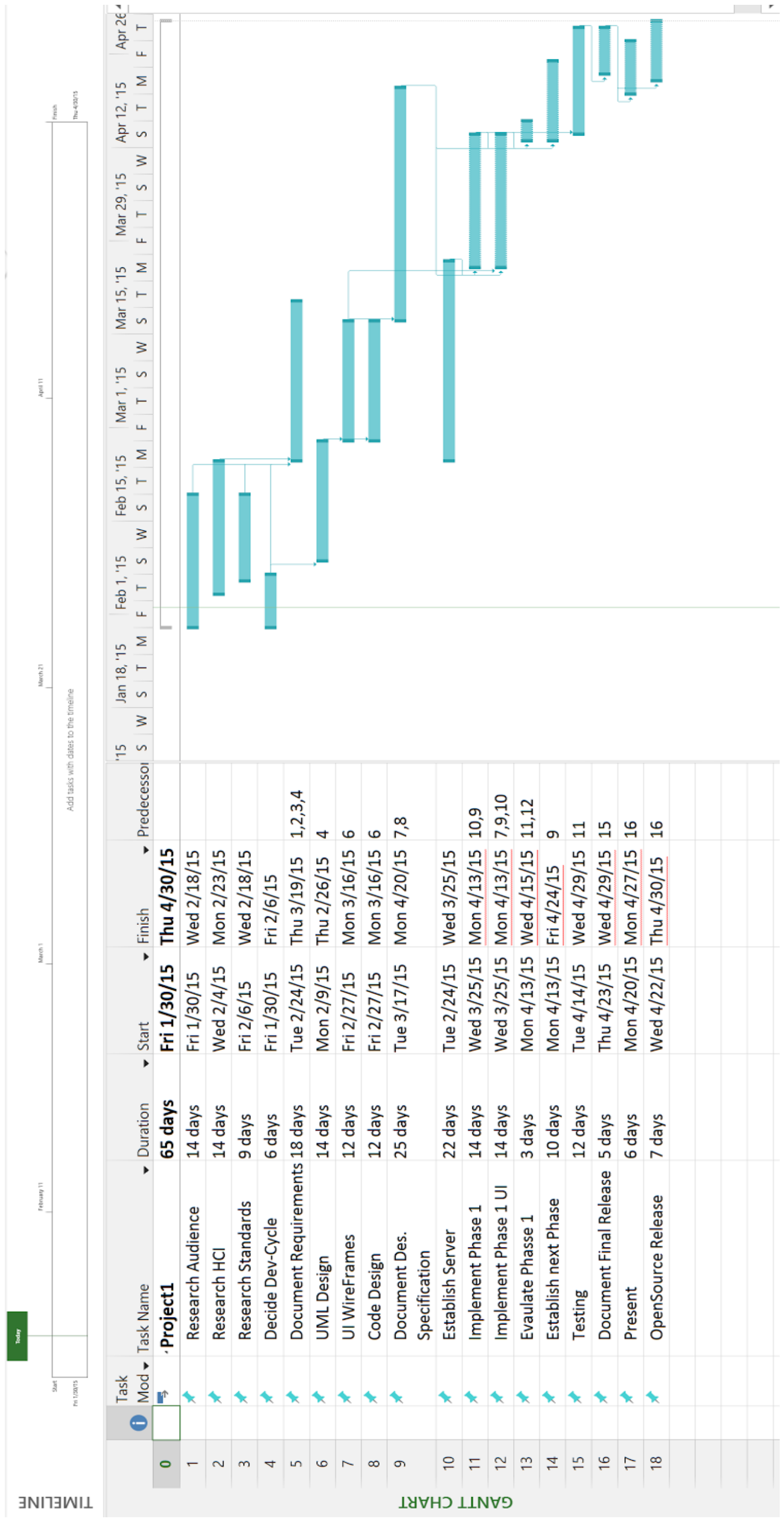
4.2 Hardware and Software Resources

There are many things that we required in order to build the email client. First we require a server as a way to run the mail client and test it. Software we require has a much wider selection. First, we require software that will assist us in design modeling. We will use a program such as Microsoft Project which will allow us time-lines and Gantt charts. We will then use a UML program to help establish our layout. After implementing our project, we may require some home-brew programs which will test our email service. Once testing and debugging is complete, the project is ready to be released open-source.

4.3 Special Resources

We will not be using any specialty resources in this project.

5. Appendix A - Gantt Chart



	Task Mode	Task Name	Duration	Start	Finish	Predecessor
0		**Project1**	**65 days**	**Fri 1/30/15**	**Thu 4/30/15**	
1		Research Audience	14 days	Fri 1/30/15	Wed 2/18/15	
2		Research HCI	14 days	Wed 2/4/15	Mon 2/23/15	
3		Research Standards	9 days	Fri 2/6/15	Wed 2/18/15	
4		Decide Dev-Cycle	6 days	Fri 1/30/15	Fri 2/6/15	
5		Document Requirements	18 days	Tue 2/24/15	Thu 3/19/15	1,2,3,4
6		UML Design	14 days	Mon 2/9/15	Thu 2/26/15	4
7		UI WireFrames	12 days	Fri 2/27/15	Mon 3/16/15	6
8		Code Design	12 days	Fri 2/27/15	Mon 3/16/15	6
9		Document Des. Specification	25 days	Tue 3/17/15	Mon 4/20/15	7,8
10		Establish Server	22 days	Tue 2/24/15	Wed 3/25/15	
11		Implement Phase 1	14 days	Wed 3/25/15	Mon 4/13/15	10,9
12		Implement Phase 1 UI	14 days	Wed 3/25/15	Mon 4/13/15	7,9,10
13		Evaulate Phasse 1	3 days	Mon 4/13/15	Wed 4/15/15	11,12
14		Establish next Phase	10 days	Mon 4/13/15	Fri 4/24/15	9
15		Testing	12 days	Tue 4/14/15	Wed 4/29/15	11
16		Document Final Release	5 days	Thu 4/23/15	Wed 4/29/15	15
17		Present	6 days	Mon 4/20/15	Mon 4/27/15	16
18		OpenSource Release	7 days	Wed 4/22/15	Thu 4/30/15	16