Nikomen Mail Client

Team 2
Requirements Specification

1. Project Overview

We noticed two extremes represented by currently available webmail clients. One side of the spectrum includes programs with very well built back end systems, which were very powerful and well functioning, however they suffered from a lack of responsive and user-friendly front end views. On the other hand, we see well organized and intuitive front-end models, which suffered from weak or poorly designed back ends. We decided that creating a web-mail client with a simple but 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 will not only prove the concept, but can add a refreshing new product to the market.

1.1 Scope and Objectives

We would like to be able to release our product to the public at the end of the project. As such, our audience is for individuals or small companies with a need for a free, stable, and feature rich email client. Our product's scope will encapsulate two different target audiences. The first is made up of common consumers, individuals who will look for a feature-rich platform to compete with major companies such as Microsoft and Google. The second will be companies or organizations such as schools, which will be more focused on the security and stability of a free, open-source package. In either case, the reliability of our product will provide a large foundation for these other specifications to be built upon.

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

1.2 Supplementary Requirements

The non-functional requirements of any web-based program are often extensive. Foremost, we will require constant availability of the server. Any downtime can affect the usability of our product and will determine the success of our program outside of the confines of this project. To retain users, we will require an intuitive interface and layout. This means having a simple and uncluttered page design, with easy to find navigational features such as obvious login/logout buttons, well placed toolbars, and an overall visual design that is appealing and clear will increase the likelihood of our product to succeed. This will be partly achieved through customizable settings as well as user-designed filters and sorted mailboxes.

2. Customer Requirements

In this section we will go over the descriptions of each of the actors used in our project. We will then go through the use case diagrams both for the project as a whole and the more specific use cases. Lastly we will go into our use cases descriptions.

Actor Descriptions

- User
  - This actor represents a user of the email client. This individual will be writing and reading emails. In addition they are also provided tools to organize and maintain their account. These settings are associated with their account on the server.
- Server
  - This actor represents the server. The server handles all active user sessions. During a session, the server will handle requests initiated by users. Any requests/posts for data (emails) will require the server to interact with the database.
- Database
  - This actor represents the database. The database contains the information on all user accounts and manages email data. Any requests/posts for data will be sent to the server and then handled by the database.

Use Case Diagrams

The image below is our higher-level use case diagram. This features the individual actions the user may go through during their session. The rest of the images in the section are diagrams for specific actions. Each use case diagram is explained in the corresponding use cases.

Figure 1: General Use Case Diagram

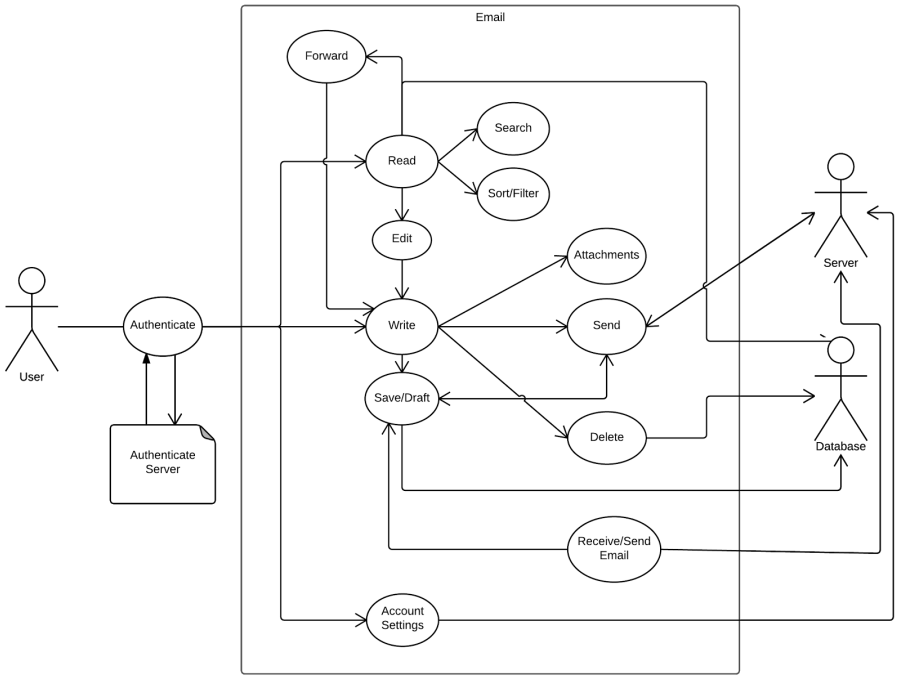


Figure 2: Read Use Case Diagram

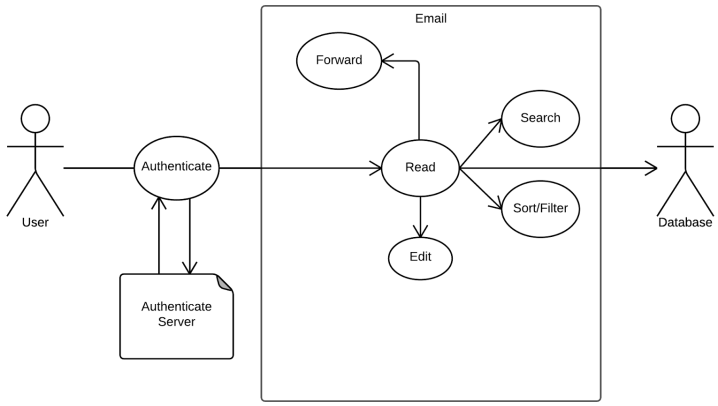


Figure 3: Write/Draft Use Case Diagram

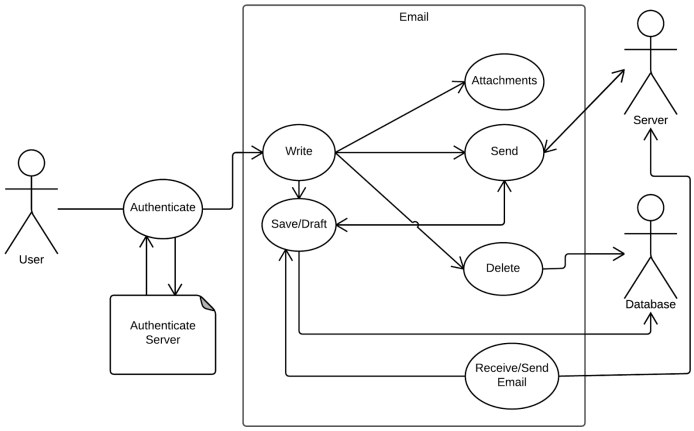
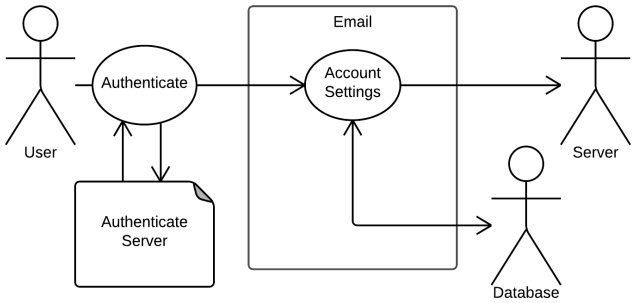


Figure 4: Settings Use Case Diagram



Use Case

The table below features the high level overview of the program.

Table 1: General Use Case

Use case name	**Email web client (simple)**
Participating actors	Initiated by the User, and Server Communicates with the DataBase, Server
Flow of events	1) The use successfully authenticates and will have access to many options 2) The user is able to Read, Write, and access Account Settings and logout 3) The use can then access many methods from within the first methods
Entry condition	The user authenticates successfully into the system
Exit condition	The user chooses to log out
Exceptions	The user failed to authenticate. The server(s) cannot be accessed.

Below features our prioritized use cases. These are the main functionalities the user will attempt. Instead of splitting them out in the usual fashion, we chose to showcase them side-by-side. These are having an equal priority, and we believed this modification lended to the readability of the uses cases.

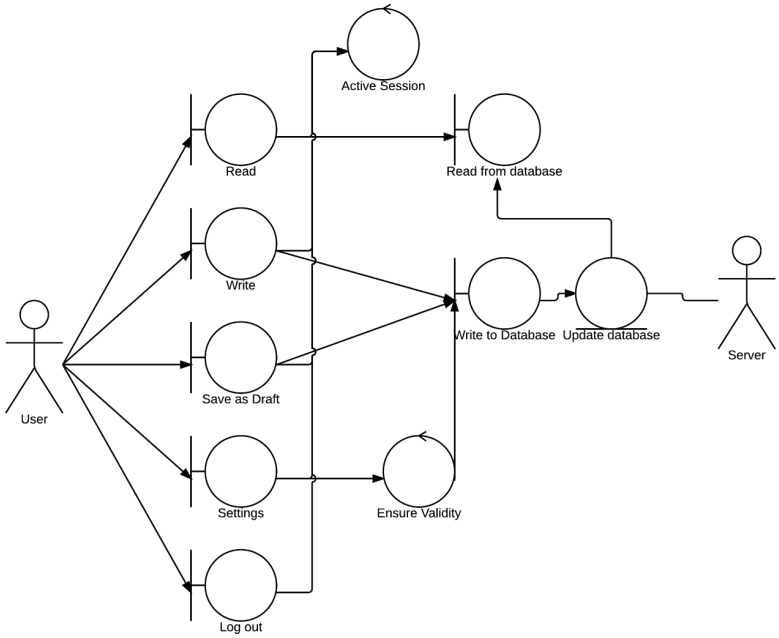
Table 2: Use Cases

USE CASE CHART					
Use case name	Writing/Sending and following methods	Receiving emails	Reading and following methods	Search/Filter	Attachments
Participating actors	Initiated by the user. Communicates with the Database. Communicates with the Server.	Initiated by the Server or the user. Communicates with the Database. Communicates with the Server.	Initiated by the user. Communicates with the server and database	Initiated by User Communicates with display	Initiated by user
Flow of events	1) The user authenticates and is accepted. 2) The User chooses to write, which can be initiated from forwarding a message, choosing to edit a saved (writable) message, replying to a message, or choosing to write a new email 3) The use is able to write a message using text 4) The User can optionally choose to add attachments 5) When the User is finished writing, the user can save, discard, or send the message 6) If saved, the message is saved to a user desired box in the database 7) if discarded, the message is simply discarded 8) if send, the message is sent out to the server for the server to handle sending, the message is also stored in the SENT box 9) When one of these options is chosen, the user is exited out of the write screen	1) The Server constantly is checking for new messages 2) optionally, the user chooses to check for new emails 3) if one is received, the server checks who the message belongs to, saves the message appropriately and flags the message as unread. 4) The server is continually checking for emails	1) The use is authenticated and is within a certain inbox 2) The user can choose which email to read 3) The database retrieves the email for the user 4) the user can choose to delete, save, edit (if the email is editable because it is a draft). or reply/forward. 5) If save, the message will be saved in a certain inbox. The use can also specifically choose to save the attachment locally. 6) if reply/edit/forward, the write function will be called 7) if the user chooses discard, the message will be saved in a deleted box, if the message is already in the deleted box, the message will be deleted from the database	1) Retrieve matching mail 2) Apply conditionals (sort/filter) 3) Display	1) User submits a request to attach an item while writing 2) display will pop up with attachment options 3) user can continue with other writing options
Entry condition	The user or another method initiates the Write method	the server finds that a new email has arrived	the user initiates the read method	The user chooses to search or filter mail in a box	chooses to attach an item while writing
Exit condition	The use chooses to send, save, or discard while writing a message.	the server saves the email to the proper uses inbox	the user exits or chooses one of the options that the read method can call.	no mail, stops filtering/sorting	attachment is removed or embedded
Exceptions	Failed to send, sent to a non existent email, server is down		Bad attachment on email, bad message, database cannot be found.	no display, no mail, bad filter	attachment cancelled, attachment is a bad file, too large of a file

2.1 Structural Analysis

In this section we will discuss the structure of our project. First we will go through our analysis model of what sessions the user will be interacting with. Next we will look at our Class Diagram. This contains all the objects for our program. It also features all the associations, inheritance, and multiplicity seen for each object. The class diagram demonstrates exactly how our code will be laid out in an Object Oriented manner. Lastly, we will look at our deployment diagram. This shows the different controllers being used within our project and how they interact.

Figure 5: Analysis Model



Writing, reading, and changing settings are some of the main interfaces that the user can use. These must all be inside of an active user session (which is began when the user logs in). Validation of input and any requests/posts for data are hidden from the user and are only accessible by the server/database.

Below is the class diagram for the project. Here, one of the main components is the account class. The account class contains all of the user's personalized information along with organizational parameters for containing email. Each account is guaranteed to have an Inbox, Sent, Trash, & Drafts email boxes. We also include the ability for the user to create their own boxes which email can be stored in. Other classes describe the components of email, such as the address, sender, and most importantly: the message. The message itself is made up of message parts. This is due to the stands for email. Another large component to each account is the Address Book. Each account will have at least one person (the user) in their address book. They can then add, edit, or delete contacts or mailing lists.

Figure 6: Class Diagram

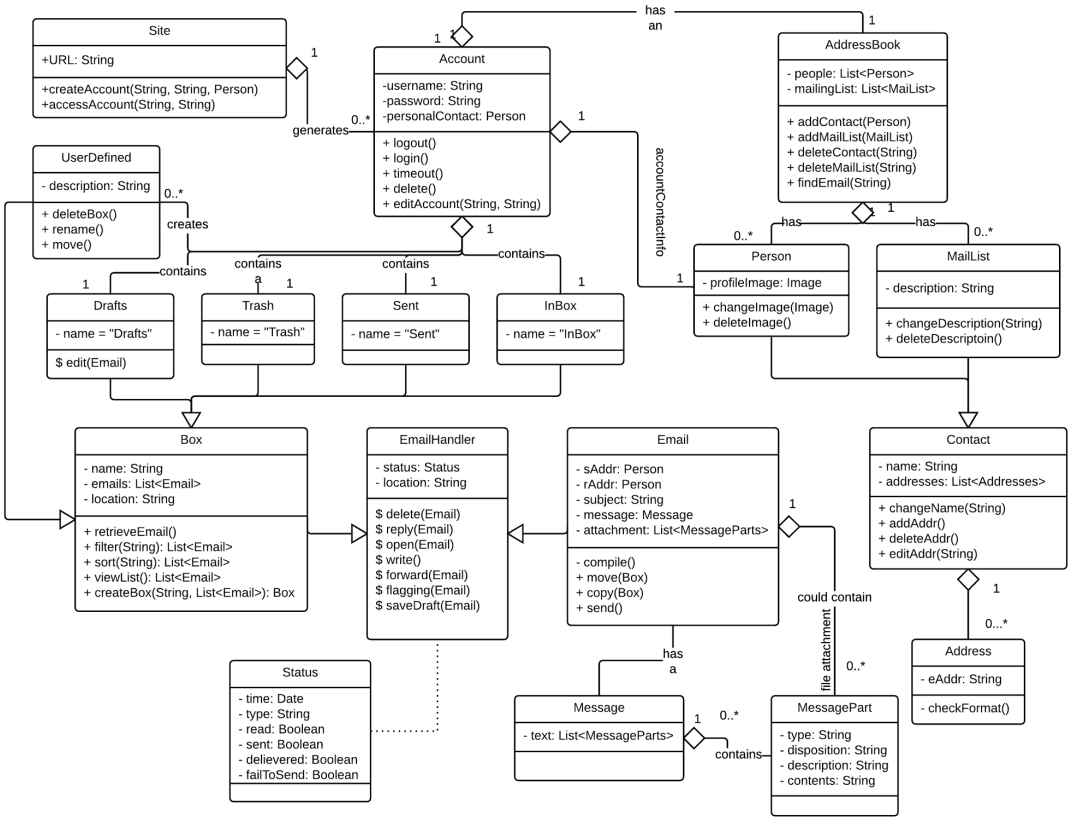
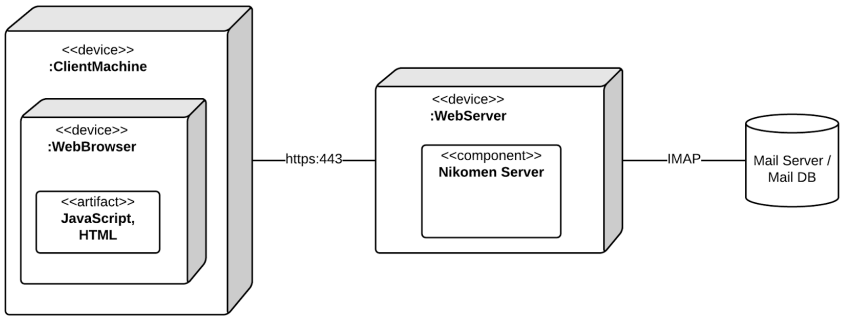


Figure 7: Deployment Diagram



Above is a deployment diagram for our system. The middle component is a technical web-server, which will carry the Nikomen server program. The client machines will open run any browser of choice to access the site. Using JavaScript and servlets, the browser will communicate with the web-server, sending requests and user input, as well as retrieving displayed content. The server will in turn, communicate with the database and mail server, retrieving the

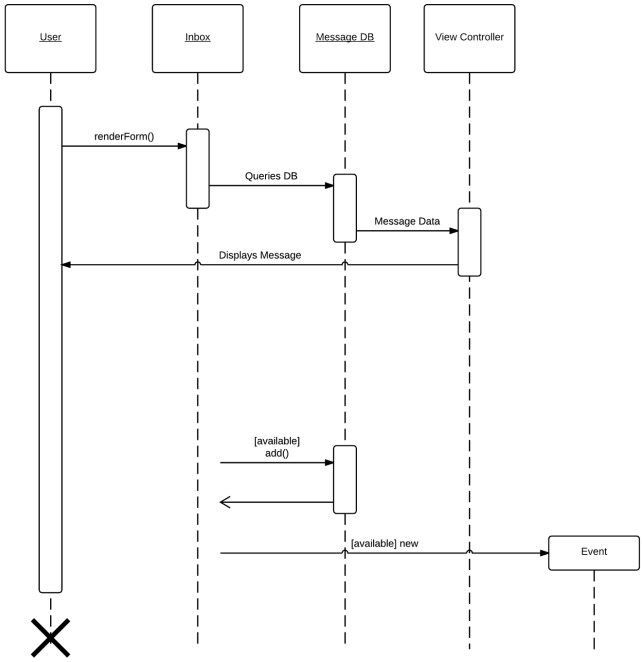
data to be displayed to the client. Using the Internet Message Access Protocol (IMAP), the web server will be able to access the mail server and retrieve the messages for each client.

2.2 Behavioral Analysis

Our four main user cases that define the behavior of the system are creating an Account, Logging in, Reading messages, and writing new messages. The sequence charts below depict this, followed by the activity diagram that brings them all together.

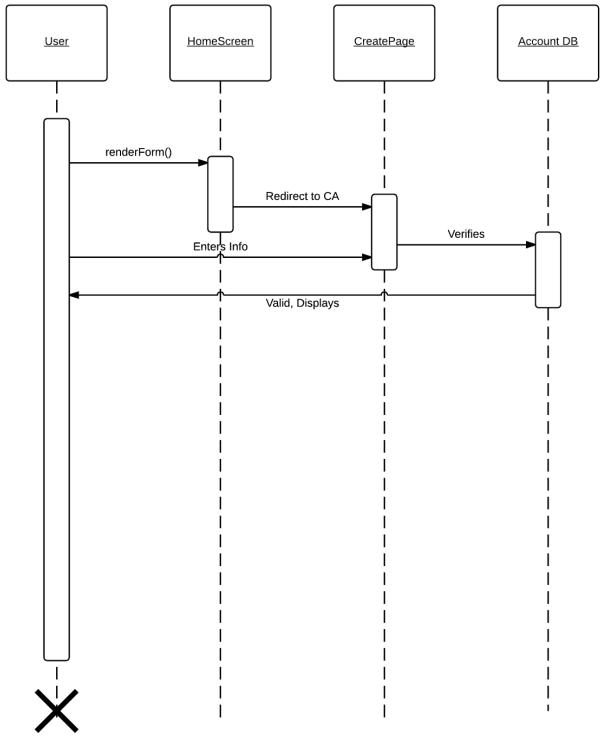
For reading messages, the user will be displayed the inbox which will list their messages. When a message is selected from the inbox, the client system will query the database, requesting the message. Assuming their is no error in the database and the message is found, the database will then forward the data to the view controller which will then format the data to be displayed to the user. The database can then update the client's list of unread messages, send corresponding attachments or act in correspondence to other events.

Figure 8: Read Sequence Diagram



When creating an account the user will be rendered a form. They will enter the requested info and press submit. The controller will check the entered data for validity and then create a new account object on the server with the information stored in the account database. The user will then be redirected to their inbox and can begin using their new account.

Figure 9: Create Account Sequence Diagram



When logging in after an account has been created, the user will simply enter their username and password into the specified form fields. The controller will check the username against those in the account database and if it finds a match, it will then check the password hash against the stored one associated with the username. If this matches the user has been authenticated and will be directed to the homepage inbox.

When writing a new message, the new email page is displayed by to the user, with fields for recipient and subject. Once the message has been completed, it is encoded and sent to the server which will then forward it out of the network and to its final destination. The user will be redirected to the inbox.

Figure 10: Login Sequence Diagram

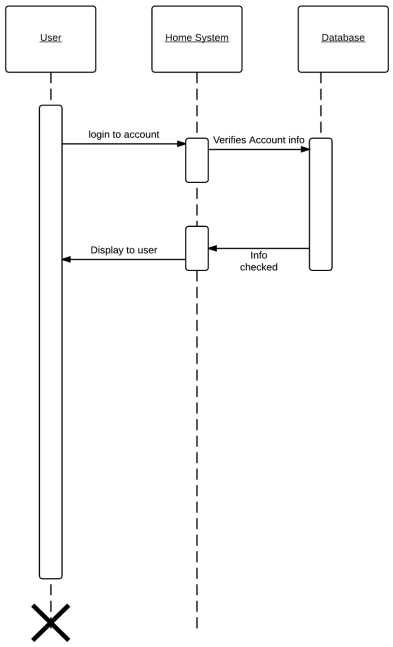


Figure 11: Write New Message Sequence Diagram

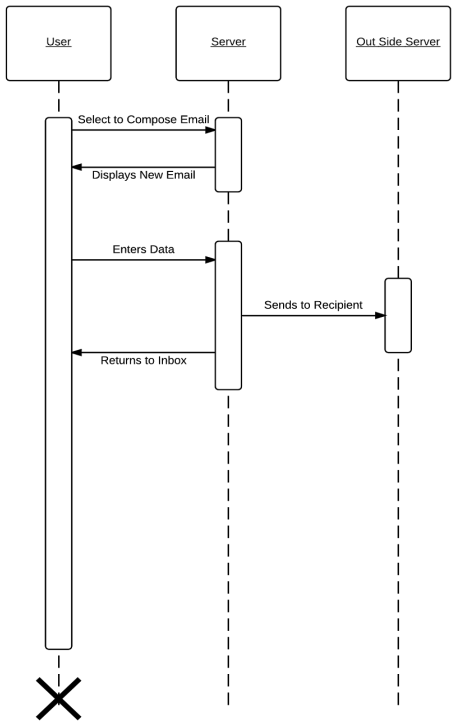
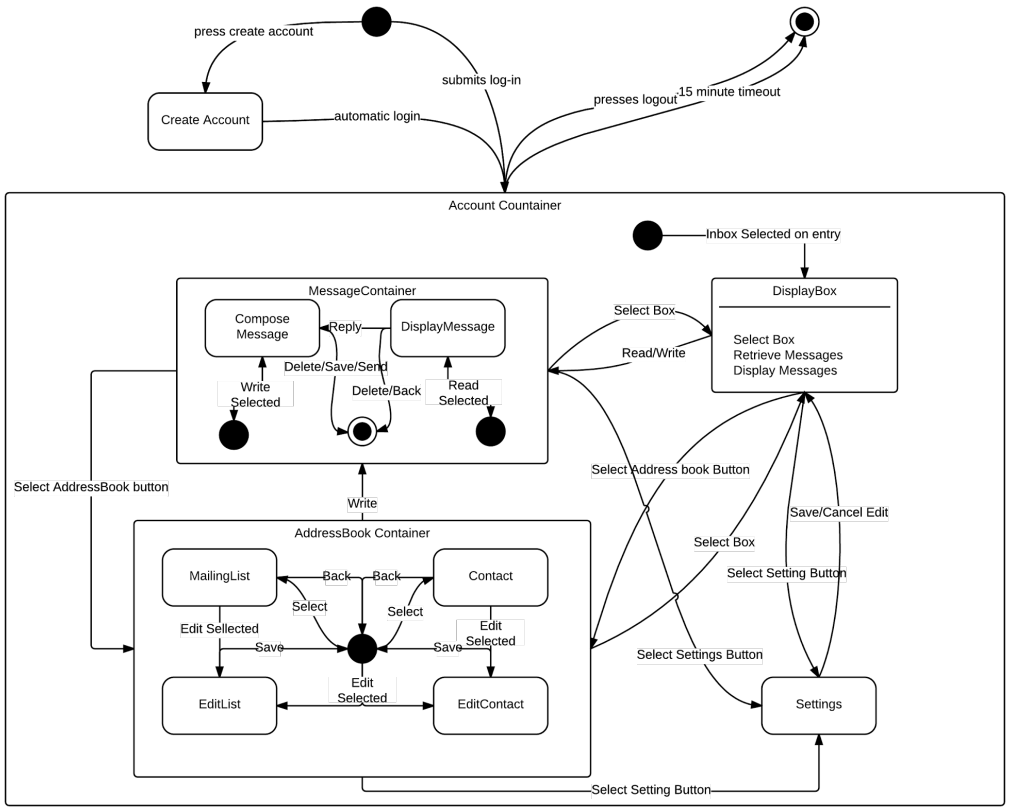
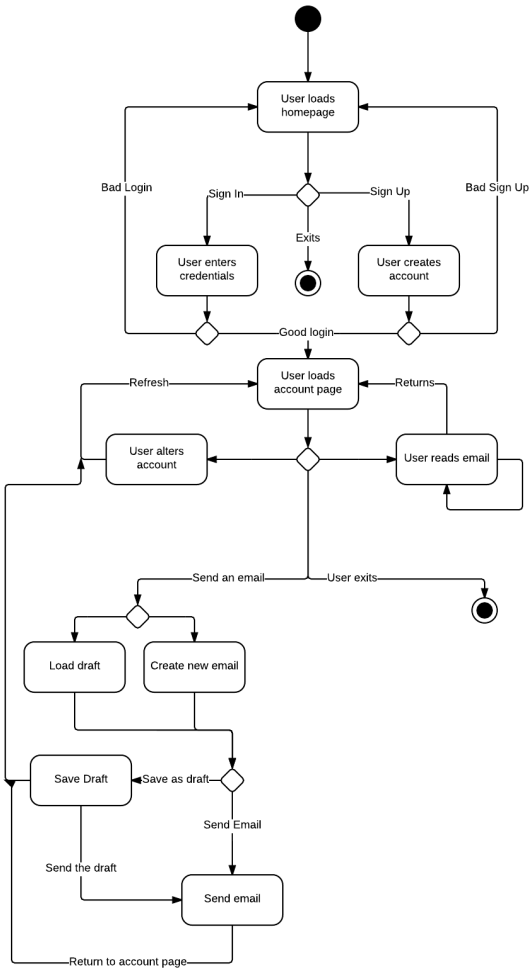


Figure 12: State Diagram



The state chart above depicts the possible change in states that can occur with our system. The initial state is the user accessing the homepage. They can then either enter their credentials and log in or link to the create account page to make a new account. When an account is made or the user logs in, the machine enters the account's container. The inbox is selected upon entry, through the user can change the box being displayed to view their sent messages, drafts, spam, or other custom box. If a message is selected, the state changes to display the message. From here it can move back to the box or can move to the write message state to either reply or forward the message. The message may also be deleted, directing the state back to the inbox display. From the inbox, the address book can be accessed. From this state, the mailing list or contact details can be accessed. The state may change to an editing contact or list from here, returning to the address book once editing has finished. The user will return to the inbox when finished viewing the address book. From any state within the account, the settings can be edited, again returning to the inbox once the edited settings have been saved. (State Chart)

Figure 13: Activity Diagram

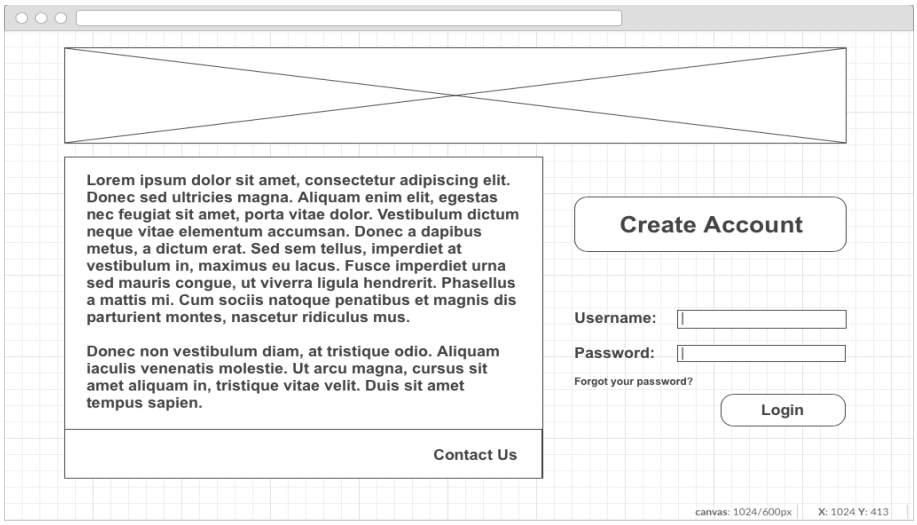


The activity diagram above depicts the possible actions and following actions that can occur starting when the user opens the program to the home screen. The user then can either sign in, sign up, or exit. The user may end up returning to the home screen if they sign in or sign up with invalid credentials. When the user has signed up or logged in, their account page is loaded. From here the user may read emails, which then returns to the inbox. This activity will loop as the user reads all the emails in the box. The user may also choose to send an email, edit the account settings, or exit the program. If the user selects the edit settings feature, they will be directed to the form, enter information, and then save. Saving will return the user to the home page. If the send an email, one of two things will happen. They may load an existing draft, or they may choose to start a new message. Once they write the message, they will either choose to save a draft which will save a copy of the email without sending it, or send the email. Either case will then loop them back to the inbox once complete. Once the user is done, he will exit the program, ending the activity.

2.3 User Interface Analysis

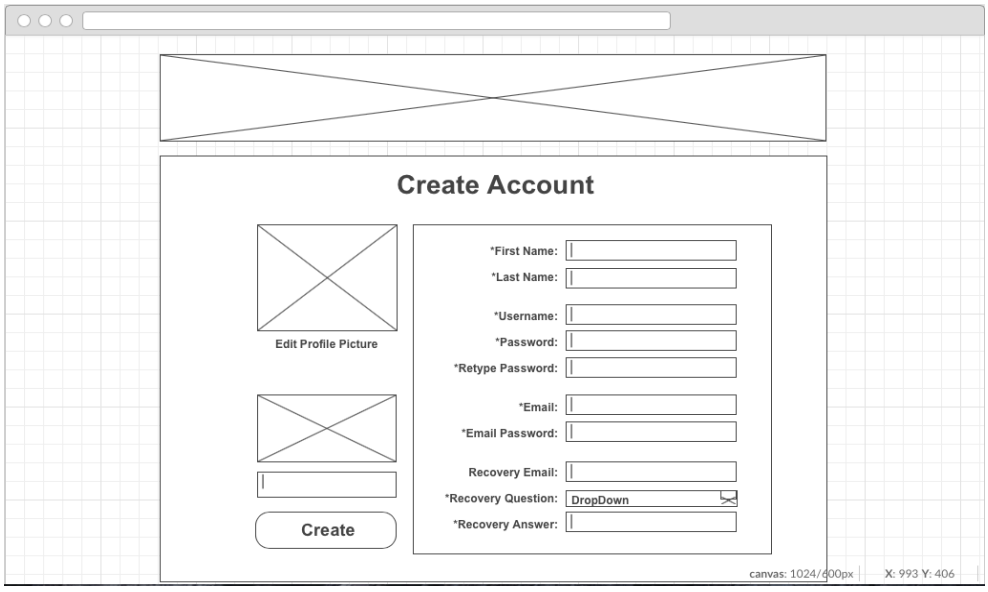
One of our main goals was to implement a user interface that was simple but responsive and overall, user friendly. Because so many web clients available focus heavily on a fancy interface but lack proper back-end management, we felt that a more concise design would be optimal.

Figure 14: Home Screen Wireframe



Our home screen will be displayed as above. Our logo will be at the top, with a small description of our product and company on the left. On the right of the window the user will be presented with two options. The first, a button that will lead them to our page that allows new users to make an account that will link to and then display their emails. The second option is a two field form, prompting for a username and password and will log a user into their account.

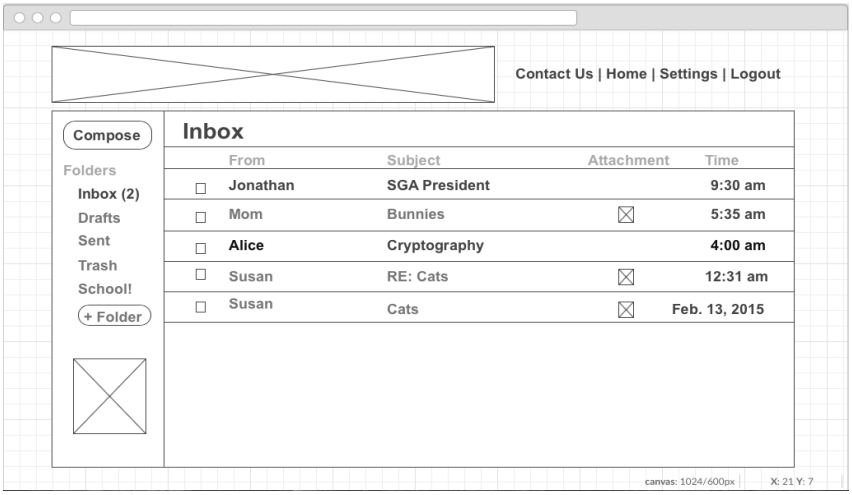
Figure 15: Create Account Wireframe



The create account page is shown above. It is a form consisting of fields for the user's name, chosen username and password, their email, and two options for recovery in case the user forgets their credentials. There will also be an option to upload a profile picture.

When the user logs in, they will be displayed their inbox, as shown below. The logo will be in the top left, with links to Contact Us, Home, Settings, and the logout option adjacent. Below that in the left column will be the list of the user's different mailboxes with the option to create boxes with custom filters. Below this will be the link to the user's address book. At the top of the column will be the button to compose a new email. The user will also be able to view their emails, which will be displayed as the selected boxes contents, sorted as the user wishes.

Figure 16: Inbox Wireframe



When the Compose button is selected the form for a new email message will be displayed. To, CC's, and the Subject will be at the top with the select attachments below it. A large text box for writing text in the message will be below that.

Figure 17: Write Message Wireframe

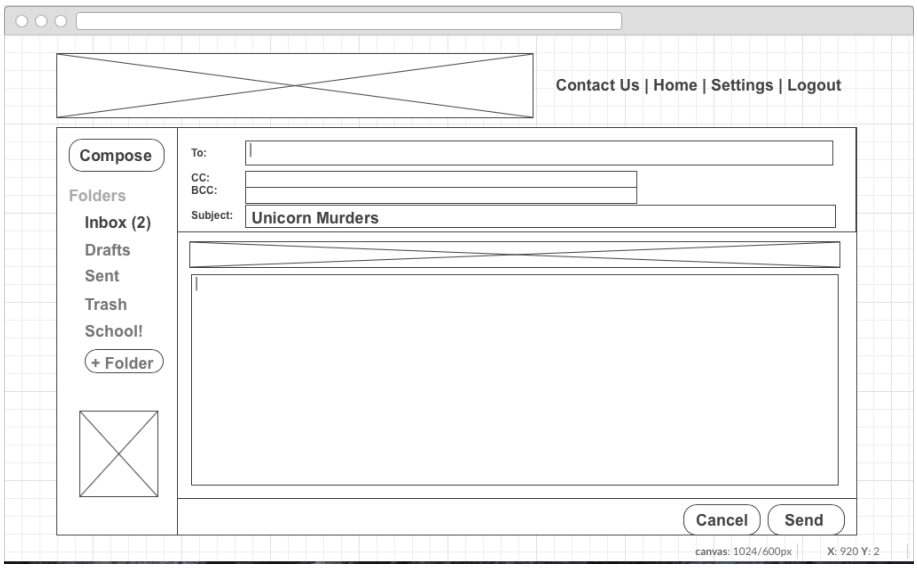
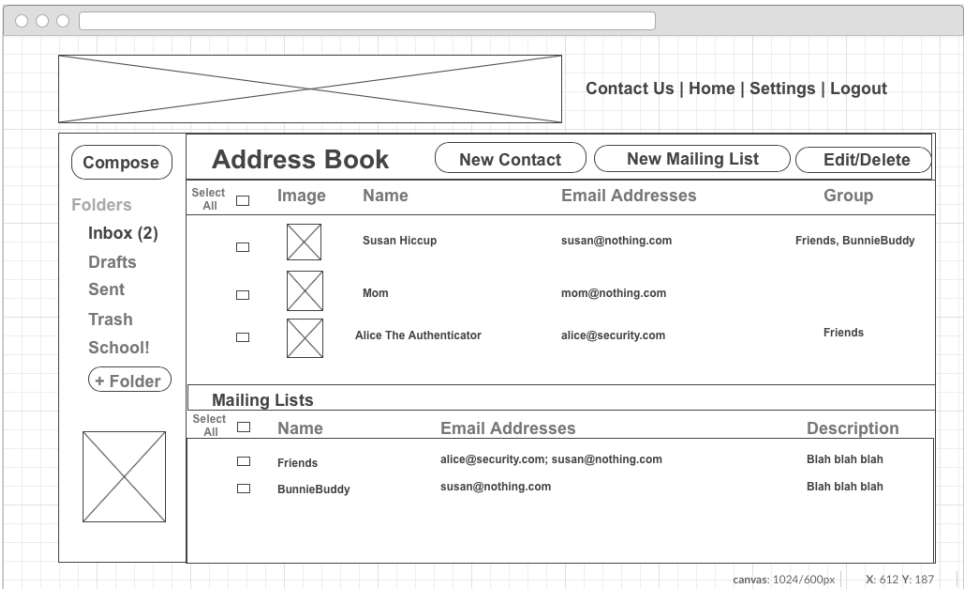


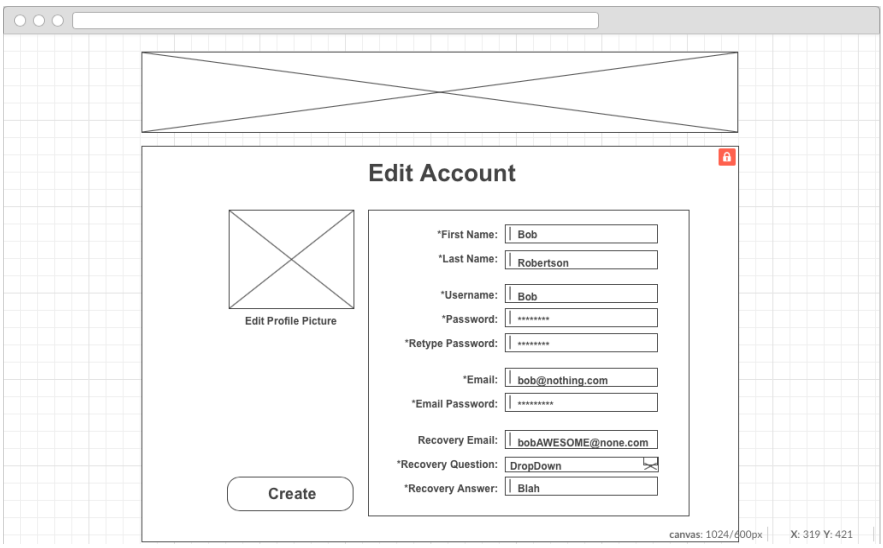
Figure 18: Address Book Wireframe



When the user views his or her address book the option to create sub lists of contact groups which will be displayed at the bottom under Mailing Lists. Above this will be the individual contacts that the user has added. Each contact will have a profile image, a Name, any email addresses associated, and any groups that the contact belongs to. The user will be able to add new contacts and create new mailing lists as well as select contacts to edit or delete.

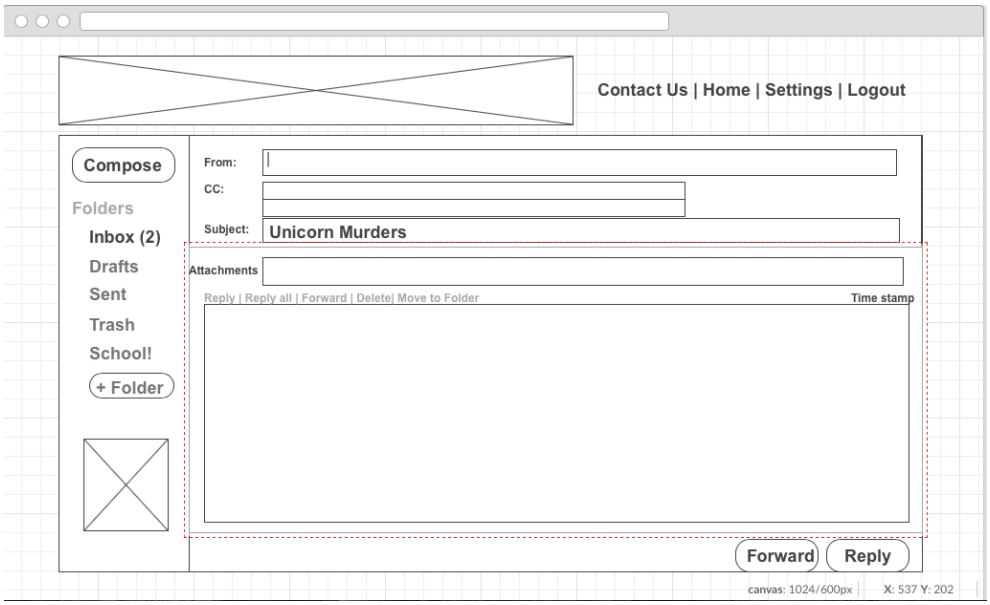
Should the user at any time select the Settings option in the top right, they will be directed to the page below. This page will allow the user to edit any details of their account from name to password and such. Once the details are as wished, the user will select Save and their changes will be saved to the server.

Figure 19: Edit Account Wireframe



When the user selects an email, it will be displayed as below. The logo and navigational bar will still be at the top and the left column will still contain the folder and address book options. The top of the center box will contain the email message's header information, From, Subject and CC's. Below that the messages contents will be displayed, with any attachments in the box above and the text in the large box below. The user will then be presented with options to either forward the email or reply to it. The user can return to any box by selecting it on the left. They also will be able to reply, reply all, forward, delete, or move the message by the option bar between the attachments and text.

Figure 20: Read Wireframe



3. Validation and Criteria

Considering that this project is providing a service to a broad audience, each use case must be rigorously tested for a variety of inputs. For sending email, the email must be guaranteed to arrive if provided with a correct address(s). This must also be expanded to include that the email will arrive regardless of the message and any attachments. The email must also be guaranteed to render. To test this, garbage and inputs designed to break the system will be applied.

When logging in the system will need to be testing for not only login credentials but prepare in case if the user disconnects while attempting to log in. The system must properly recover and continue providing service. To test this, user instances will prematurely kill connections to the server while attempting to log in. This form of testing will also be applied to any other user action on the system.

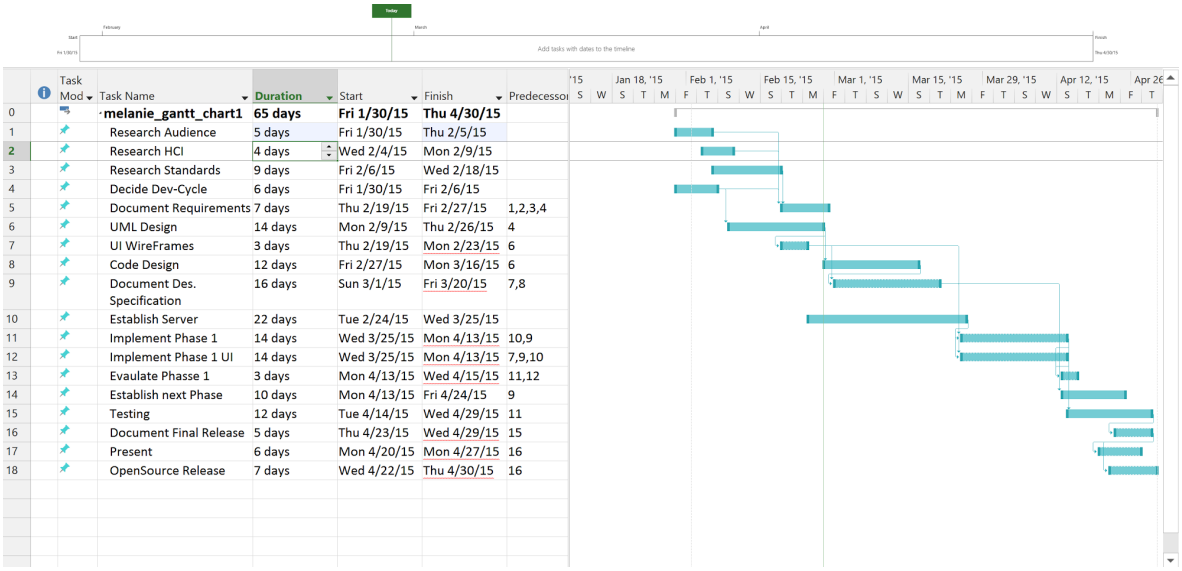
For creating accounts the system must be able to recognize bad inputs and handle them. This can range anywhere from the user attempting to place special characters in input fields to overly long inputs. Testing will need to make sure that it is impossible for a user to place in bad system both to protect them and the reliability of the server.

4. Appendices

4.1 Project Status

Our team got a bit behind due to a mistake on our proposal Gantt chart. As such, we were steadily working, but not quite as fast as we needed to be. Other than the extension on this document, we are now ahead of schedule. We have finished our requirements, our server has a basic setup, and we are about to begin our first phase of implementation. Below is our updated Gantt chart.

Figure 21: Project Schedule Gantt Chart



	Task Mod	Task Name	Duration	Start	Finish	Predecessor
0		melanie_gantt_chart1	65 days	Fri 1/30/15	Thu 4/30/15	
1		Research Audience	5 days	Fri 1/30/15	Thu 2/5/15	
2		Research HCI	4 days	Wed 2/4/15	Mon 2/9/15	
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	7 days	Thu 2/19/15	Fri 2/27/15	1,2,3,4
6		UML Design	14 days	Mon 2/9/15	Thu 2/26/15	4
7		UI WireFrames	3 days	Thu 2/19/15	Mon 2/23/15	6
8		Code Design	12 days	Fri 2/27/15	Mon 3/16/15	6
9		Document Des. Specification	16 days	Sun 3/1/15	Fri 3/20/15	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