Software Requirements Specification for MailJar

Prepared by Kim Ficara and Zach Shaver. Submitted on March 12th, 2017

1. Introduction

This specification covers the base functionality desired of the MailJar e-mail client, such as logging in, reading e-mail, and sending e-mail. The main purpose of the product is to allow users to access their e-mail service from their desktop, as opposed to logging in through their browser. The intended readers are the software development team, product managers and clients.

2. Overall Description

MailJar is a self-contained product, and is not a part of any existing system nor is it replacing an existing product. It must allow the user to: 1) log into their e-mail, 2) view their received messages, 3) send a message, and 4) refresh to pull in new e-mails. (See Appendix A for a state diagram displaying how this is to be accomplished). There will not be a separation of user classes and privileges; all users will be offered the same functionality. The program will be written in Java and thus will operate in the Java Virtual Machine (JVM) environment, allowing for ease of portability. It will fetch emails using the IMAP protocol, as opposed to POP3, so that e-mails will be stored on the server instead of the client computer. The design pattern to be employed will be Model-View-Controller. The GUI will be built using the JavaFX framework. A simple help menu will be included as user documentation. The only third-party dependency will be the JavaMail API, which will provide the programming interface for accessing and sending email.

3. External Interface Requirements

The user interface will provide controls via keyboard and mouse. Functions such as logging in or sending an email, where the main process will be filling a form, should be made so that the user can accomplish the action completely using the keyboard (e.g. Tabbing between fields, pressing Enter to submit). The GUI components will consist of: 1) a login window, 2) a main inbox window, 3) a compose message window, and 4) a message display window. The program will not require any specialized hardware, but will need access to the internet. It will allow the user to select from a list of hosts (e.g. Gmail, Live, Yahoo, etc). The actual getting and sending of e-mail will be done by looking up the IMAP server for the particular host.

4. System Features

4. 1. Log-In

High Priority. User will enter their username and password, select their host, and press submit/enter to log-in. If there is an error, it will be displayed to the user so they can correct it. Otherwise, it will show the inbox window.

4. 2. Show inbox

High Priority. User will see a certain number of e-mails, listed one after another in a scrollable window. Each message will consists of the sender, the title, a brief summary of the message, and the date and/or time the message was received. Double clicking on or pressing enter while a message is selected will open up the message window to show the user the full message.

4. 3. Show full message

High Priority. The full HTML content of the message will be displayed using JavaFX's WebView component, along with other information like the sender and title. The user will be able to forward or reply to the e-mail by clicking a button, which will open up the send e-mail window with the current message as a draft. For replies, the recipient will be set to the sender.

4. 4. Send e-mail

High Priority. The user will be able to enter the recipients and the title of the e-mail. They will be able to enter their message in JavaFX's HTMLEditor component, which will allow them to style their message as they choose. The e-mail will be sent when a button is clicked. If there is an error, a message will be displayed.

4. 5. Refresh

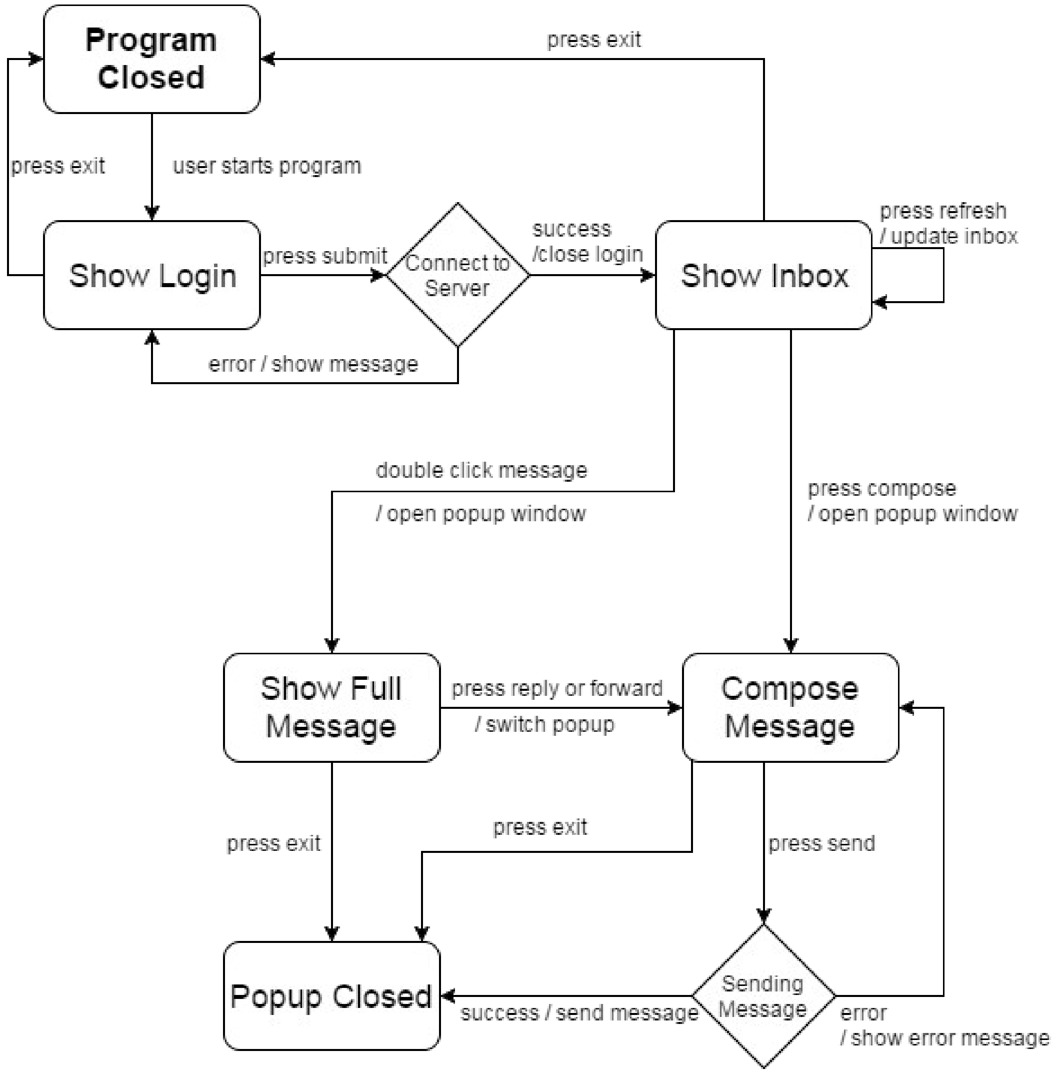
Medium Priority. User should be able to select the refresh button in the inbox window, and new messages will be displayed. This will require fetching emails and updating window if there are new messages.

5. Other Nonfunctional Requirements

There are no expected performance requirements, as the program is not computation-heavy. There are a couple security requirements however. Since the program will be local, the passwords can be stored, if necessary, on the client's computer. E-mail in general is vulnerable to Javascript injection attacks, so when the messages are displayed they should be stripped of any Javascript code.

Appendix A

State Diagram of Application



Program Closed
press exit
press exit
user starts program
Show Login
press submit
Connect to Server
success /close login
Show Inbox
press refresh / update inbox
error / show message
double click message / open popup window
press compose / open popup window
Show Full Message
press reply or forward / switch popup
Compose Message
press exit
press exit
press send
Popup Closed
success / send message
Sending Message
error / show error message