MailJar Development Report

Prepared by Zach Shaver and Kim Ficara March 30th 2017

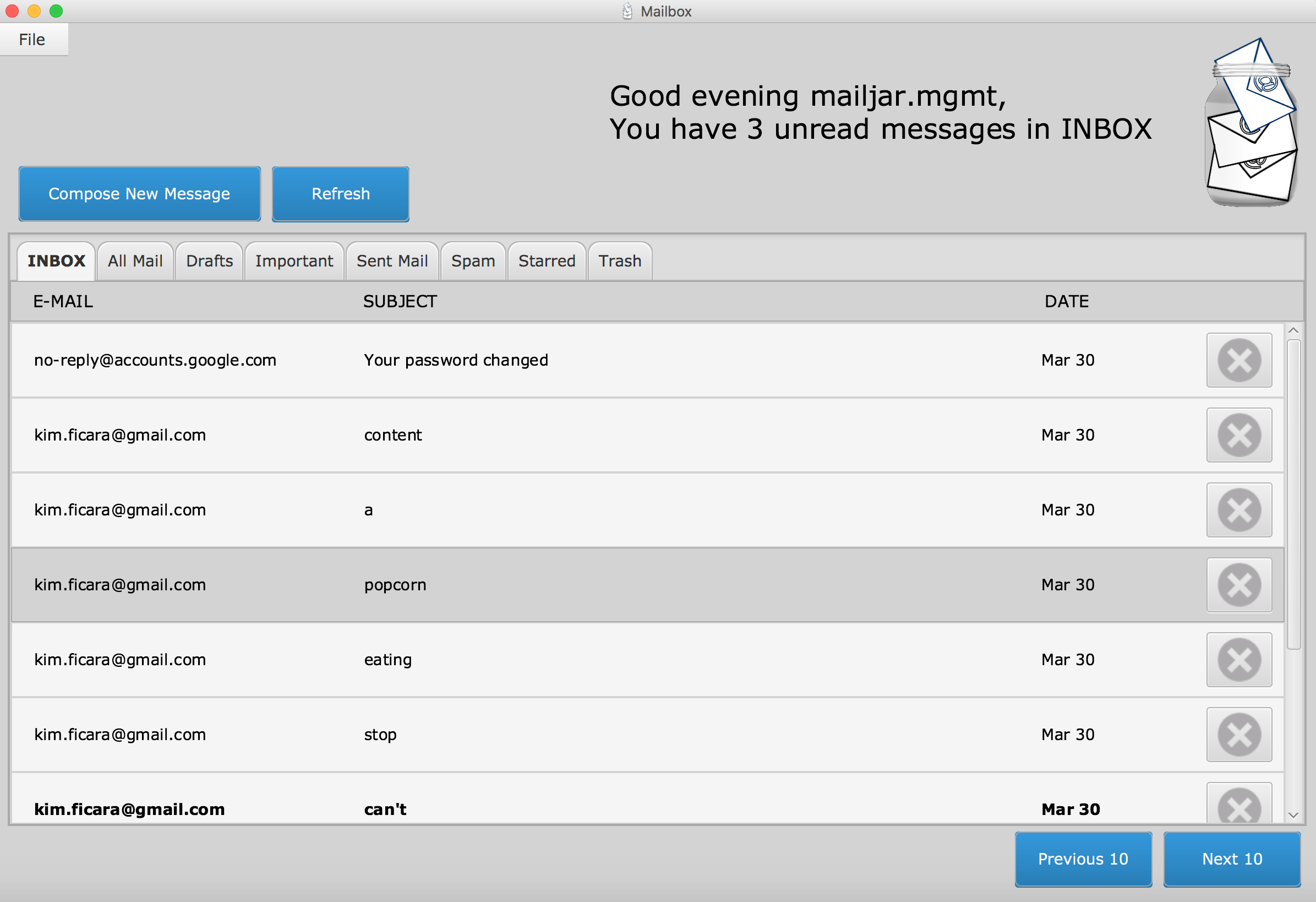
# User Interface

### Login Window



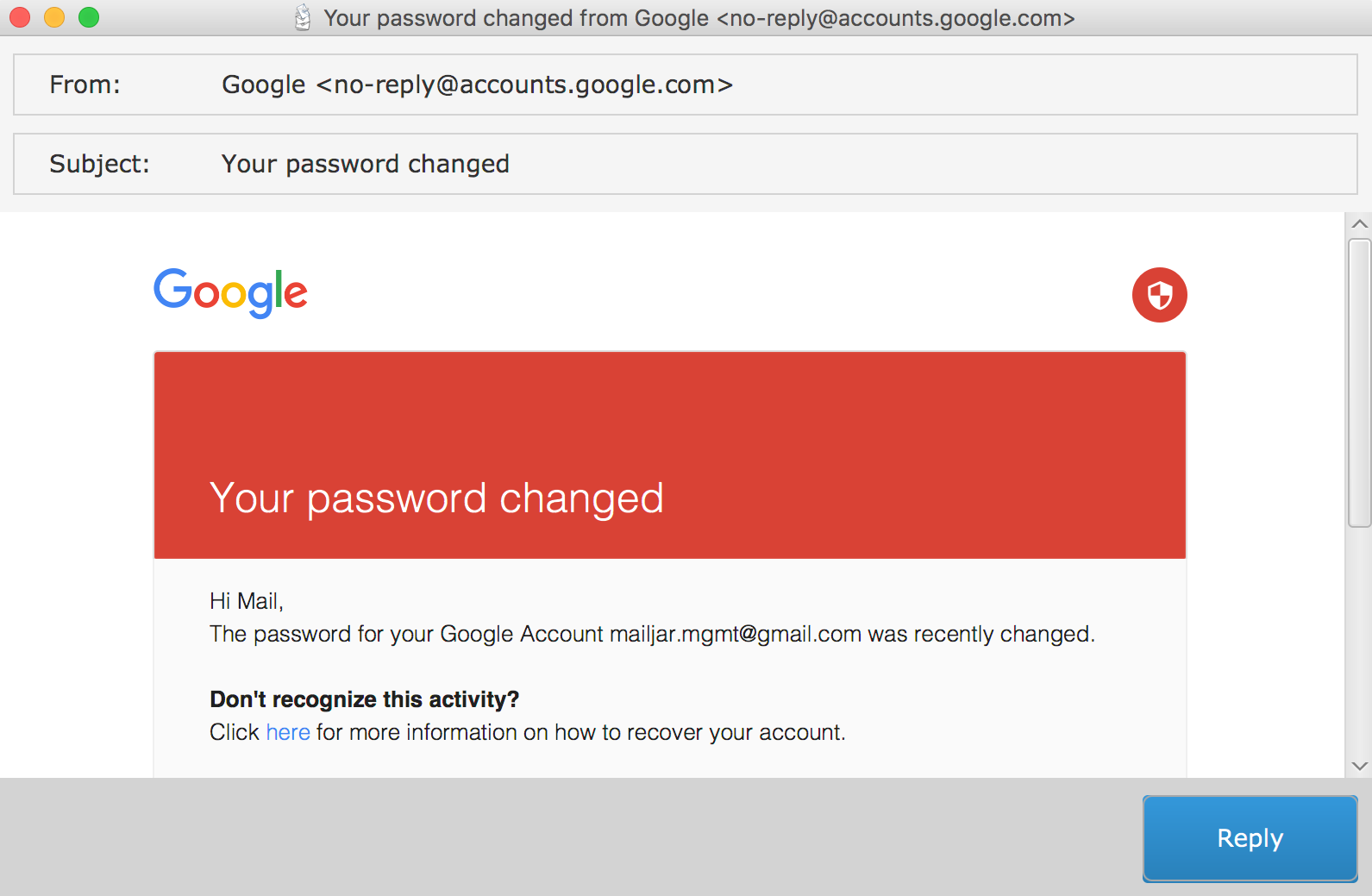
* Combo box with 5 different hosts
* Hotmail.com
* Gmail.com
* Uwindsor.ca
* Outlook.com
* Live.com
* Prompt message to update user on login status
* Password field is masked

## Mailbox Window



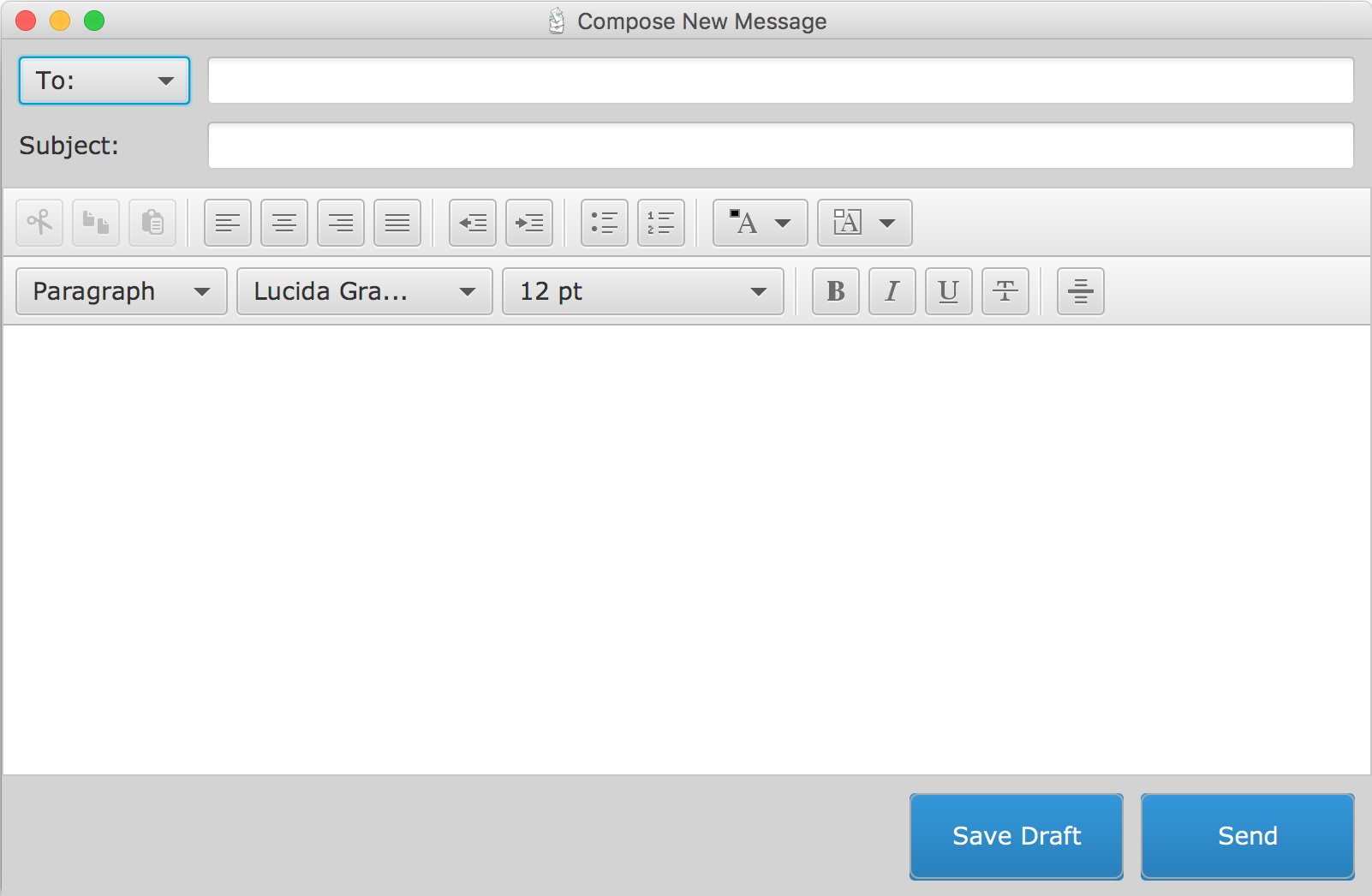
* Customized greeting prompt with updated number of unread messages
* Compose new message button
* Refresh message button
* Tabs with different folder names and content
* 10 messages displayed per page
* User can browse through 10 messages per page
* Delete message button flags selected message for deletion

## Message Window



* Displays messages in HTML format, including images
* User can hyperlink to internet sites
* Reply button

## Compose Message Window

* Option to send a message to multiple recipients (cc, bcc)
* Subject line
* HTML editor to allow user to fully format and customize their message
* Save draft button saves the message to draft folder (it existing)
* Send button sends the message to recipients and closes the window

# Implementation

#### Logging in

* When the log in button is pressed, the e-mail address and password are collected.
* A request is made to the host for two different classes of connection: simple mail transfer protocol (SMTP) and internet message access protocol (IMAP)
* SMTP is used for sending messages and IMAP is used for receiving and updating the messages
* If the request is satisfied, the mailbox window will open with these connections.
* If the request fails, an exception will be printed at the bottom of the log in window.

#### Mailbox

* Once the connection has been made, we filter through all folders of the mailbox and create a tab for each of the folders and add an event listener to the tabs.
* A message is displayed according to the time of day (e.g. Good Morning) with the number of unread messages. If a user refreshes, changes folder or reads a new message, the greeting will be updated.
* When a tab is pressed, the tab pane will load and display the messages for that given folder.
* The default tab pane will be the first folder in the list (usually the inbox).
* When the delete button is pressed, the delete flag will be set for that message and when the folder is refreshed the message will be removed from the mailbox.
* The refresh button reconnects to the server and retrieves the messages from the current folder and displays the messages in the current tab pane.
* The “next 10” and “previous 10” buttons loop through the array of messages stored in the mailbox model and display 10 messages at a time in the tab pane.
* The view for opening a message is “MessageWindow” and the view for composing a message is called “ComposeWindow”.
* If the user closes the MailboxWindow, the entire program will close. The Mailbox Window can be closed by clicking the “x” in the top left corner or selecting exit menu item.

#### Opening a Message

* When a user requests to open a message, we first check to see if the current folder is drafts. If it is not drafts, we open the message in a regular MessageWindow.
* The sender, subject and content of the message are loaded in MessageWindow upon construction.
* When the reply button is clicked, it will pass the message to the ComposeWindow (explained in the next section).
* If a new message is opened for the first time, the seen flag is set to true and the message line bold style is removed.

#### Composing a New Message

* The compose window is invoked for three different cases: sending a message, replying to a message and opening a draft.

1. Sending a message from the MailboxWindow creates a blank e-mail to be filled out by the user.
2. Replying receives the message and loads the sender as the recipient, prepends a Re: to the subject line and inserts a horizontal rule above the replied to content.
3. Opening a draft loads the message content exactly as it was saved.

#### Other

* If an exception is thrown anywhere in the program, it will be displayed in a pop up window.
* All other messages to the user will be displayed on the scene.