Software Requirements Specification

for

Email-Client

Version 1.0 approved

Prepared by Intro to SE Group 2:

Sheldon Waid, sw1914; Eli Hartnett, ech364; Braedon Kimball, bmk228; Elliot Ware, mw2161

Group 2

August 31, 2020

1. Introduction

1.1 Purpose

The purpose of this document is to create an email web application by which users can send and receive messages with attachments, sort, search, and forward emails, and be able to login to the web application through a portal.

1.2 Document Conventions

This document is written in 11-pt Arial font using Google Docs as a means to write sections of the document collaboratively. Functional requirements are placed in tables to facilitate understanding of their design. The main sections are in enlarged font, bolded, and numbered to provide clarity and to make them distinguishable from each other. Subsections are numbered in decimal and also bolded for the same reason.

1.3 Intended Audience and Reading

This document is intended for any student working on the web application, as well as the other students who may wish to interact with our lab. It is also for the instructor of Intro to Software Engineering and the lab instructors who administered the project requirements. This project will be useful to any CSE student who wishes to learn about web development as well as host/client communication between servers, remote data communication, and web design. The order in which one may read this document should be from section 01 through section 05.

1.4 Product Scope

The purpose of the email web application project is to facilitate understanding of software development using AGILE techniques and development cycle. Furthermore, it will allow users to login to access emails, compose, edit and send emails to other users, search and forward emails to other users, and send attachments along with the email. We will use Python for data communication and <language> to design the user interface. Github will be used in order to maintain prototype versions, share code, and maintain documentation.

1.5 References

https://msstate.instructure.com/courses/42669/files/2216369/download
https://msstate.instructure.com/courses/42669/files/2227555/download
https://krazytech.com/projects/sample-software-requirements-specificationsrs-report-airline-database

2. Overall Description

2.1 Product Perspective

The purpose of this product is to offer an efficient and safe emailing service for the user. The main interface will display the user's most recent emails as well as other feature options the user can call upon. The product is new and self-contained, meaning that it will contain everything necessary to function. The product will be a website of web applications that is a web-based emailing service. This is an open service, meaning that it can be used in both a working environment and a school environment.

2.2 Product Functions

· Login into their account via email address and password to access the emails

· The accessibilities of this product include being able to receive, read, compose, and send emails

· The user will be able to search and forward emails to other users while also being able to receive forwarded emails from another user.

· In addition to emails, the user will be able to send and receive any attachments which include but is not limited to images, files, and links.

2.3 User Classes and Characteristics

User Class classified by highest to lowest priority of satisfaction (highest: 1, lowest: 5)

1. Admin: this is the highest priority user class with the most advanced technical or educational experience. This class has the highest security and access privileges. This class can add or delete other accounts of any other user class, but cannot view other user's emails.

2. Manager: This user class has a proficient level of technical or educational experience and can manage only his/her individual account. Because this user class is a manager but not an admin, only certain employee accounts can be managed from this class. Such as a manager-employee scenario or teacher-student scenario.
3. Employee: This user class has a proficient level of technical or educational experience and can manage only his/her individual account. The security level of this user class is basic, including username/password.
4. Instructor: This user class has a proficient level of technical or educational experience and can manage only students accounts in a school setting. The security level of this user class is basic, including username/password.
5. Student: This is the most basic user class used only for school settings and managed by teacher or employee user class. The security level of this user class is basic, including username/password.

2.4 Operating Environment

· The email client will work on Windows operating systems and will be available only on any machine with internet connection.

· The email client will be written in Python and HTML and will implement a database that is compatible with both languages.

· The email service will be peer-to-peer

2.5 Design and Implementation Constraints

The email service will have to be used with the internet. Users will only be able to view their own accounts and cannot access any other user's without the proper credentials. For prototypical purposes, there will be a limited number of users, and as the email service updates, the number of allowed users at a time will increase. For the initial versions of the email service, a website only version will be available. The email service will run on a SQLlite database or Pandas which are compatible with Python. The customer's organization can contact the developer for any technical support or maintenance.

3. System Features

3.1 Create a Unique Account

3.1.1 **Description and Priority**

Users will be able to make a unique account that cannot be replicated.

Priority: High

3.1.2 **Stimulus/Response Sequences**

Stimulus: User clicks on New Account.

Response: Application opens a new account window.

Stimulus: User types in a username.

Response: Application compares username to the database and flags if it is a duplicate.

Stimulus: User types in a password.

Response: Application compares password to the database and checks for password specifications.

Stimulus: User types in an email address.

Response: Application compares email address to the database and flags if it is a duplicate.

Stimulus: User clicks create account.

Response: Application saves data into database as attributes.

3.1.3 **Functional Requirements**

Username Textbox	Application shall have a textbox for user to type in a username for comparison.
Password Textbox	Application shall have a textbox for user to type in a password for comparison.
Email Textbox	Application shall have a textbox for user to type in an email address for comparison.
Create Account Button	Application shall have a button for submitting data to the database for comparison.
Users Database	Application will have a database containing data for comparison.

3.2 Login to a Unique Account

3.2.1 **Description and Priority**

Users will be able to login to a unique account.

Priority: High

3.2.2 **Stimulus/Response Sequences**

Stimulus: User types in a username.

Response: Application compares username to the database and flags if it does not exist.

Stimulus: User types in a password.

Response: Application compares password to the database and checks to see if it matches the password with that username.

Stimulus: User types in an email address.

Response: Application compares email address to the database and checks to see if it matches the email to that account.

Stimulus: User clicks login.

Response: Application brings up the home login page with email.

3.2.3 **Functional Requirements**

Username Textbox	Application shall have a textbox for user to type in a username.
Password Textbox	Application shall have a textbox for user to type in a password.
Email Textbox	Application shall have a textbox for user to type in an email address.
Login Button	Application shall have a button for submitting data to the database for comparison.
Users Database	Application will have a database containing data for comparison.

3.3 List of Mail

3.3.1 Description and Priority

Users will be able to see a list of the latest received mail.

Priority: High

3.3.2 Stimulus/Response Sequences

Stimulus: User logs in.

Response: Application shows latest received mail.

Stimulus: User clicks on the mail button.

Response: Application opens the mail link.

3.3.3 Functional Requirements

Mail Panel	Application shall have a panel with mail.
Mail Button	Application shall have a button connected to the mail link.

3.4 Mail Meta-Data

3.4.1 Description and Priority

Users will be able to see the meta-data of the received mail. This will be the time, sender, and subject.

Priority: High

3.4.2 Stimulus/Response Sequences

Stimulus: Click on an email.

Response: Application opens email and has meta-data written on it.

3.4.3 **Functional Requirements**

Mail Meta-Data	Application shall save the meta-data upon receiving the mail.

3.5 Sending Mail

3.5.1 **Description and Priority**

Users will be able to send emails.

Priority: High

3.5.2 **Stimulus/Response Sequences**

Stimulus: User clicks compose.

Response: Application brings up a compose panel.

Stimulus: User clicks on send.

Response: Application compiles composition and sends it to the appropriate address.

3.5.3 **Functional Requirements**

Compose Button	Application shall have a button to bring up a composition panel.
Send Button	Application shall have a button that compiles the composition and sends the mail to the correct address.

3.6 Attaching documents to email

3.6.1 **Description and Priority**

Users will be able to attach a document to the email.

Priority: High

3.6.2 **Stimulus/Response Sequences**

Stimulus: User clicks on attach button in mail composition panel.

Response: Application shows a file directory path text box that the user can type the location of a file into.

Stimulus: User clicks on the second attach button to attach the file in the path.

Response: Application attaches it to the email

3.6.3 **Functional Requirements**

Attach Button	Application shall have a button on the composition panel to open a file directory path text box.
Attach Button 2	Application shall have a button on the attach panel that attaches the listed directory.

3.7 Mail Search

3.7.1 **Description and Priority**

Users will be able to search through their mail.

Priority: High

3.7.2 **Stimulus/Response Sequences**

Stimulus: User clicks search bar.

Response: Application opens search panel.

Stimulus: User types in search bar.

Response: Application shows mail containing the letters the user typed in.

3.7.3 **Functional Requirements**

Search Textbox	Application shall have a textbox that lets the user type in what they want to search.

3.8 Forward

3.8.1 **Description and Priority**

Users will be able to forward an email to another user.

Priority: High

3.8.2 **Stimulus/Response Sequences**

Stimulus: User clicks forward on opened email.

Response: Application opens a mail panel with empty receiver and subject box.

3.8.3 **Functional Requirements**

Forward Button	Application shall have a button that will open a mail panel with that email and an empty receiver and subject box.

3.9 Logout

3.9.1 **Description and Priority**

Users will be able to logout of the account currently logged in.

Priority: High

3.9.2 **Stimulus/Response Sequences**

Stimulus: User clicks logout.

Response: Application closes all panels and returns to the login panel.

3.9.3 **Functional Requirements**

Logout Button	Application shall have a button that closes all panels and returns to the login panel.

4. Other Nonfunctional Requirements

4.1 Performance Requirements

To perform properly, this application is required to run using a website or web app. Upon logging in, users should be able to compose, edit, and send emails (including attachments) to other users. Logged in users should be able to search and forward emails to other users. The system should work in real time meaning after one user sends an email, the recipient should be able to see it on their end nearly immediately if they are logged in.

4.2 Safety Requirements

Harm to the product would include someone/a hacker getting access to another user's login credentials. To keep this from happening, we will have minimum restrictions on password complexity. A possible loss to the product includes someone/a hacker getting access to the

storage of emails, login credentials, etc. and leaking or erasing the data for loss to the product or users. The product's design must uphold policies to properly handle public, protected, and private data/functions to keep the product safe.

4.3 Security Requirements

The product needs to have security so that users can only see the data they are allowed access to. This includes data such as login credentials (usernames and passwords) and email contents (communication logs or attachments/files). For a user to login successfully, they must correctly input their email (case sensitive) and password (case sensitive and minimum complexity requirement of letters, characters, and numbers).

4.4 Software Quality Attributes

The quality of the product needs to be robust and reliable so that users feel confident using it. The design and code needs to be readable, clear, and simple so users can focus on what the product was made for and programmers can focus on delivering a more robust and reliable product. The design should also leave room for future improvement for things such as dual authentication when logging in, adding a section to sync contacts, adding an integrated calendar, and etc.

5. Other Requirements

Appendix A: Glossary

Button: A defined area on the application a user can click. It is linked to a piece of action code that will be executed when clicked.

Textbox: A defined area on the application a user can type into. It is connected to a button and will take the information typed in once that button is clicked.

Database: A structured piece of code to hold data of the users.

Flags (verb): Notifies the user of the application.

Panel: Part of the user interface that can be seen.

Web App / Web Application: A software program that is used over the internet rather than being downloaded to a device.

Python: A computer programming language to control the program

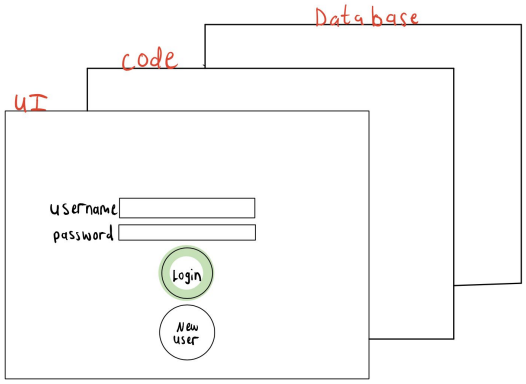
Self-Contained: a program or library that is fully independent and contains everything necessary to function properly.

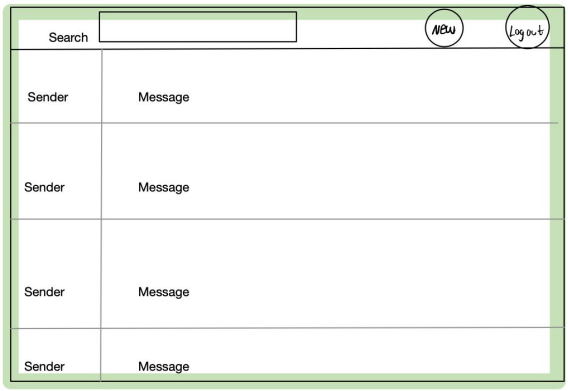
HTML: A computer programming language to assist the program; specifically for a graphical user interface

Agile: Software development method

Github: An online platform to assist software developers in communication and organization

Appendix B: Analysis Models





Appendix C: To Be Determined List

No TBD references were listed in this document.