**Simple Email System**

**Soo Yung Ting**

**33354456**

**UNIVERSITY OF SOUTHAMPTON**

**(MALAYSIA CAMPUS)**

**30th March 2023**

Declaration of Academic Integrity

**Please sign to indicate that you have read and accepted the following statements. Your assignment will not be accepted without this declaration.**

|  |
| --- |
| I confirm that:   1. I have read and understood the University’s [Academic Integrity Guidance for Students](http://www.southampton.ac.uk/assets/imported/transforms/content-block/UsefulDownloads_Download/AAA325B0BCDE4123BF24D81F85B8882C/Academic%20Integrity%20Guidance%20for%20Students.pdf) and that in the attached submission I have worked within the expectations of the Regulations Governing Academic Integrity. 2. I am aware that failure to act in accordance with the [Regulations Governing Academic Integrity](http://www.calendar.soton.ac.uk/sectionIV/academic-integrity-regs.html) may lead to the imposition of penalties which, for the most serious cases, may include termination of programme. 3. I consent to the University copying and distributing any or all of my work in any form and using third parties (who may be based outside the EU/EEA) to verify whether my work contains plagiarised material, and for quality assurance purposes. |

|  |
| --- |
| The extent to which I have worked with others is as follows: |

This submission is inadmissible without a written signature below (applies to hard copy only).

Name: Soo Yung Ting

Diagram

Description automatically generated with low confidence

Signature:

Date: 30th March 2023

**Table of Content**

|  |  |  |
| --- | --- | --- |
| **Section** | **Content** | **Page Number** |
| **1** | **Introduction** | **4** |
| **2** | **Quality Of Work** |  |
| **3** | **Program Design** |  |
| **4** | **User Interface Design** |  |
| **5** | **Solutions** |  |
| **6** | **Suggestions For Improvements** |  |
| **7** | **Conclusion** |  |
| **8** | **Appendix** |  |

**Introduction**

A brief introduction on how the system works.

**Problem Statement**

As a computer scientist, that have been tasked with designing a Simple Email System which is similar to popular email clients such as Outlook and Gmail. The goal is to create an easy-to-use system with a user-friendly graphical user interface (GUI) that enables users to send and receive emails with ease. The current challenge is that there is no existing system that meets the desired functionality, and users require a reliable and straightforward email client that meets their communication needs. This lack of an effective mechanism for sending and receiving emails has led to users experiencing difficulties in managing their correspondence, leading to missed opportunities and decreased productivity. To address this problem, the client desires an email system that provides up-to-the-second email updates, user-friendly interface, and accurate data management to enable efficient email communication. The system should be adaptable and capable of accommodating the client's expanding needs while being able to generate reports and analyse email data in graphs.

Email has become an indispensable tool in today’s dynamic world, enabling us to effortlessly reach people worldwide for various reasons -- work-related or personal ones. An email platform was programmed to cater communication, inbox management, and organizational needs, ensuring a secure and trusted system. A user-friendly interface, can easily compose, send, and receive emails from anywhere, anytime, as long as it is accessible.

When a user opens the program, the user first needs to log in with their email address and password or create a new account by clicking on the sign-up button and completing the registration process. Once the user is logged in, the GUI will display the inbox, which is a list of received emails. From there, the user can select an email to read or delete. To compose a new email, the user can click on a “Compose” button or icon, which will bring up a new email composition window. In the email composition window, the user can enter the recipient’s email address, subject line, and the body of the email. If the user wants to attach a file, they can click on the “Attach file” button on the bottom left of the window and select the file they want to attach. Finally, when the user is ready to send the email, they can click on the “Send” button on the bottom right of the window, which will send the email to the recipient. The GUI may also provide additional features, such as the ability to format the text and add links.

**Quality Of Work**

The functionalities of the program work as per specifications by checking screens and controls like menus, buttons, icons, etc.

* **Reliable:** the email system should be reliable and have a high uptime rate which means that it should be able to handle a large volume of emails and deliver them without fail.
* **Speed:** the email system should be fast and be able to deliver them fast and able to deliver the emails quickly. This is an important aspect because many users expect their emails to be delivered promptly.
* **User interface:** user-interface should be easy to use and navigate. This will help to improve user satisfaction and reduce the likelihood of user errors.
* **Security:** The email system should be secure and protect user data from unauthorized access. This includes implementing encryption, multi-factor authentication, and other security measures to prevent hacking and other security breaches.
* **Scalability:** The email system should be able to handle a growing number of users and emails without compromising performance. This means that it should be designed to scale up as the user base grows.
* **Compatibility:** The email system should be compatible with different email clients and platforms. This includes being able to send and receive emails from different email clients and platforms.
* **Customization:** The email system should be customizable to meet the unique needs of individual users and organizations. This includes the ability to customize email templates, signatures, filters, and other settings.
* **Clarity:** The menu should be clear and easy to understand, with well-labeled and organized options. Users should be able to quickly find what they are looking for without confusion or frustration.
* **Functionality:** The menu should provide access to all the necessary features and functions of the email system, such as composing, sending, and receiving emails, as well as managing contacts, folders, and settings.
* **Accessibility:** The menu should be accessible to all users, regardless of their level of experience or ability. It should be easy to navigate with keyboard shortcuts or other accessibility features and should be optimized for use on different devices and screen sizes.
* **Sign-In and Sign-Up buttons:** These buttons should successfully allow the user to either log in to an existing account or create a new account with a valid email address and password. They should also provide appropriate feedback to the user if there are any errors or issues with their login or sign-up process.
* **Compose button:** This button should successfully open a new email composition window with all the necessary fields for composing a new email, including To, Subject, Body, and Attachment options.
* **Send button:** This button should successfully send the composed email to the recipient's email address and provide appropriate feedback to the user if the email was sent successfully or not.

**Program Design**

The program design uses appropriate structures. The overall program design is appropriate.

* Screenshots and some explanation

|  |  |
| --- | --- |
| Graphical user interface, application  Description automatically generated | Graphical user interface  Description automatically generated |
| Graphical user interface, text, application, Word  Description automatically generated | Graphical user interface  Description automatically generated |

*For better view of these images, please refer to the appendix.*

This program design uses the Tkinter library to create a login system. The design consists of several structures, including the initialization of the root window, various UI elements, functionality, placement of UI elements, and error handling. The program starts by initializing the root window and defining its properties such as size, title, and background colour. UI elements such as labels, entries, and buttons are then created and customized using various parameters. The program defines two functions, "login" and "open\_registration," which are executed when the user clicks the "Sign in" or "Sign up" buttons, respectively. The UI elements are placed on the root window using the ‘place’ method, and the placement of these elements is done in a way that provides a good user experience. Lastly, error handling is implemented by highlighting the entry fields in red and displaying an error message box if the entered email and password do not match any of the users in the "users.txt" file. Overall, the program design follows a well-structured approach that ensures a good user experience by providing an easy-to-use and intuitive interface.

The GUI includes various input fields for the user to provide their personal information, such as their first and last names, date of birth, email, phone number, and gender. Once the user submits their information, the script checks whether the provided email is valid and not already taken. If the email is valid and available, the user's information is saved to a file called "users.txt". If there are any errors in the user's input (such as an invalid email, password confirmation failure, or already taken email), an appropriate error message is displayed in a pop-up dialog box. If the registration is successful, an info message is displayed, and the registration window is closed.

The graphical user interface (GUI) for an email inbox application makes use of the tkinter library to create windows, frames, buttons, and a listbox to display the email messages. The inbox functionality is achieved by reading email data from a text file, parsing the data to extract information such as the sender, recipient, and subject, and then displaying the subject lines in the listbox. The GUI provides options to search for emails by keywords, view and delete emails, and sign out of the application. The code also includes some functions that are currently empty or not fully implemented, such as composing and sending emails. The script is structured with function definitions at the top and the main GUI creation code at the bottom. Overall, the script is well organized and easy to read, with comments explaining the purpose of each function and variable.

The code is structured as a class called "Email" that contains the necessary attributes for an email such as sender, recipient, subject, message, and attachment. The class also includes a method for formatting the email as a string and a method for attaching a file. The main part of the program is contained in a function called "send\_email()" which is called when the "Send Email" button is pressed. This function retrieves the user input from the various tkinter widgets such as entry fields and text boxes, creates an instance of the "Email" class with the input values, formats the email as a string, and writes the email to a file. The program also includes a function called "attach\_file()" which is called when the "Attach File" button is pressed. This function opens a file dialog box that allows the user to select a file to attach to the email. Additionally, the program includes a function called "limit\_characters()" that is called every time a character is entered into the message text box. This function limits the number of characters in the message to 1000 by deleting any characters beyond the 1000th character. Overall, the program is well-structured with clear separation of concerns between the different functions and the "Email" class. The use of tkinter widgets and functions allows for easy user input and interaction, while the file writing functionality allows for the emails to be saved for later use.

**User Interface Design**

User interaction is as specified and is natural to the user. You do not need help while you are using the system.

The user interface of this code is simple and intuitive. The login screen provides two input fields for email and password, along with clearly labeled labels. The login button is placed below the input fields and is labeled appropriately. The registration button is also provided in case a user does not have an account yet. The overall design of the interface is visually appealing with a blue colour scheme and a clear font style. The error messages that are displayed in case of incorrect input are also clear and concise. The user interaction is straightforward, and the input fields are easy to use, which makes it natural for users to log in without any assistance. Overall, this interface provides a smooth and seamless experience to the user.

This code is a GUI application written in Python using the tkinter library that allows a user to register an account by providing their personal information such as name, email, password, gender, date of birth, and phone number. The entered information is validated to ensure that the email is valid and not already taken, the password and confirm password match, and all required fields are filled. If the entered information is valid, it is saved in a text file named "users.txt" in the same directory as the script. The code consists of a register\_account function that is called when the user clicks on the Register Account button. The function extracts the information entered by the user from the GUI fields and validates it. If the entered information is valid, the function saves it to the text file and displays a success message. The check\_email function is used to check if an email is already taken by reading the existing email addresses from the text file. The code also sets up the GUI window with labels and entry fields for each of the required fields. It also sets up combo boxes for the date of birth and a drop-down menu for the gender. The register\_account function is bound to the Register Account button using the command attribute.

This is a Python code that uses the tkinter module to create a basic email client. The user interface of this code is simple and easy to use. The user can interact with the system using a few buttons and a list box.

The search feature is straightforward and user-friendly. The user enters a search query in the search box, and the system displays the emails that match the search query. The search button executes the search and displays the results in the list box. The user does not need any help to use this feature.

The system also provides the user with several buttons to perform different actions. The Compose button allows the user to write a new email, and the Sent button displays the emails that the user has sent. The Delete button allows the user to delete an email from their inbox. The Sign Out button logs the user out of the system. The list box displays the subject of each email. When the user selects an email from the list, the system displays the email data in a new window. The email data includes the sender, recipient, subject, and message body. Overall, the user interface of this code is intuitive and easy to use. The user does not need any help while using the system.

The user interface of this code provides a simple and intuitive way for users to compose and send emails. The interface is divided into four sections: sender information, recipient information, subject, and message.

The user is required to fill in all the necessary fields before sending the email, and the code checks for empty fields and displays an error message if any fields are left empty. Additionally, if the message exceeds 1000 characters, a warning message is displayed, and the excess characters are automatically deleted.

The interface also provides a button to attach a file to the email. Clicking the "Attach File" button opens a file dialog window that allows the user to select a file to attach. If a file is selected, a success message is displayed, and the file is attached to the email. Overall, the user interaction is natural and straightforward. The user does not need any technical skills to use the system. The code makes use of tkinter's widgets, such as Labels, Entries, Text, and Buttons, to create the interface. The interface is also visually appealing, with a light blue background and clear labelling of each field.

**Solution**

The program is a complete solution that runs without errors. It meets all the specifications and works for all test data.

**Suggestions For Improvement**

Your suggestion to your partner if there is anything that she/he could do to improve the program.

1. Use a password hashing function to store passwords in the "users.txt" file to enhance security.
2. Consider implementing error handling for file I/O operations (e.g., file not found or permission errors).
3. Use a validation function to check the user's input email and password before checking them against the data in the "users.txt" file. The validation function should check if the email is in the correct format and if the password meets the minimum requirements for strength (e.g., contains both uppercase and lowercase characters, numbers, and symbols).
4. Add a "Forgot Password" option that allows users to reset their password by sending a password reset link to their email.
5. Password strength: You can add password strength checking to ensure that users create strong passwords. A strong password should have a minimum length, contain both uppercase and lowercase letters, and include digits and symbols.
6. Error handling: Currently, the program displays a message box for each error that occurs. It would be better to handle errors more gracefully by displaying a single error message with all the issues that need to be fixed.
7. GUI design: The current GUI design can be improved by using a more modern and attractive theme. You can use ttkthemes to add different themes to the application.
8. Internationalization: You can make the application more accessible to users by adding support for different languages. You can use the gettext module to add internationalization support to the application.
9. Input validation: You can add more input validation to ensure that users enter valid information. For example, you can check that the phone number is in a valid format or that the date of birth is not in the future.
10. Improve search functionality: The current search function only searches for text within the email fields. It could be improved to search for text within the body of the email as well.
11. Improve GUI: The GUI could be improved by adding more features, such as the ability to sort emails by date or sender. Additionally, the layout could be improved to make it more user-friendly.
12. Real time update: could let the inbox to update in real time like once an email is sent the inbox is immediately refreshed and the new email shows up.
13. Add input validation: The program should validate that the email addresses entered in the sender and recipient fields are valid email addresses, and that the subject and message fields are not empty.
14. Improve the file attachment feature: Currently, the program allows the user to attach only one file. It would be better if the program allowed the user to attach multiple files. Additionally, the program should check that the attached file is not too large, and should display an error message if the file is too large to be attached.

**Conclusion**

After you have run and tested the program, what is your overall experience of using the program or as an evaluator of the program? Did you learn something from the task, or would you improve your own program by adopting a few of the functions from your partner?

I think the program looks well-structured and readable, with a simple and easy-to-understand user interface. The code seems to be written concisely and efficiently and uses proper error handling to inform the user about any invalid inputs. However, there are a few areas that could be improved.

For example, the password is stored in plain text, which is not secure. It is recommended to use a more secure way to store passwords on the database. Additionally, the user interface could be more visually appealing by using more colours and graphics.

I learned that creating a program like this requires a solid understanding of programming concepts, data structures, and algorithms. The program also highlights the importance of careful input validation and error handling to ensure that the program can handle unexpected inputs and errors gracefully.

Regarding improvements, it's difficult to say without knowing the specific implementation details of the program. However, adopting useful functions or algorithms that could enhance the program's functionality and performance could be beneficial. Collaborating with a partner can also be a great way to learn new programming techniques and approaches.

In terms of the email client program, I think it is a simple and well-designed tool that allows users to quickly compose and send emails without having to use a more complex email client. The use of the Email class to organize and format email details is excellent, and the class contains a method for formatting the email string, which makes the code for creating and sending emails cleaner and easier to understand. I also appreciate the use of the filedialog and messagebox modules to provide a user-friendly interface.

However, there are areas for improvement in the email client program. For example, additional features such as support for multiple email accounts and the ability to view sent messages could be added. Input validation for email addresses, especially for the sender and recipient fields, would also be beneficial. Adding a feature that allows users to save their email as a draft, so they can continue editing and send it later, could also be helpful. Finally, a more robust email sending mechanism that supports sending emails through SMTP servers could be considered.

Overall, I think this email client program is an excellent starting point for a simple email client and would be a useful tool for users who do not need the full range of features provided by more advanced email clients and the program seems to be functioning correctly and accomplishing its intended purpose..

**Appendix**

Graphical user interface, application

Description automatically generatedGraphical user interface

Description automatically generatedGraphical user interface, text, application, Word

Description automatically generatedGraphical user interface

Description automatically generated