**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** | **4 - 5** |
| **3** | **Program Design** | **5 - 6** |
| **4** | **User Interface Design** | **6** |
| **5** | **Solutions** | **7 - 8** |
| **6** | **Suggestions For Improvements** | **9** |
| **7** | **Conclusion** | **9** |
| **8** | **Appendix** | **10 - 11** |

**Introduction**

**Problem Statement**

As a computer scientist, it is commissioned to design a simple email system similar to the popular email clients (such as Outlook and Gmail). The purpose is to create a user-friendly system through the user-friendly graphics user interface (GUI), and users can easily send and receive emails. The current challenge is that there is no system corresponding to the required functions. Users need a reliable, simple email client to meet their communication requirements. The lack of effective mechanisms for sending and receiving emails, which leads to difficulties in the management letter, which leads to missed opportunities and reduce productivity. To solve this problem, customers want to provide an email system that provides the current email update, user-friendly interface and accurate data management to achieve effective email communication.

In this fast-paced modern era, our reliance on email has grown beyond measure. Whether we need to make business connections or stay in touch with loved ones, this technological innovation makes global communication easier than ever before. With its emphasis on security and functionality, email software has adapted to meet the demands of users across the world. By allowing us to manage our inboxes and streamline our organizational processes, email has become an everyday essential. Its user-friendly interface means that we can send and receive messages at any time and from any location.

**Definition**

Email systems are commonly known as, electronic messages that can be sent, received, and managed through digitally - a platform or software application. A server stores and takes care of email messages, while a client application empowers users to access and manage their individual email accounts.

**How it works**

If the user opens the program, the user must first sign in with its email address and password or click the registration button and complete the registration process to create a new account. Once the registered user, the GUI will indicate the income box that represents the email list received by the representative. From there, users can choose to read or delete the email. To form a new email users can click a button or display the “Combination” symbol of the new email composition window. In the email composition window, users can enter the recipient’s email address, subject and main body. If the user wants to add a file, it can click the “Add File” button at the bottom left of the window, and then select the file to be added. If the user is ready to send email, they can click the “Send” button in the lower right corner of the window to send to the recipient.

**Real World Applications**

A straightforward email system is a flexible tool that may be used by people, companies, and organisations for a range of objectives. Email is primarily used for communication, enabling quick and inexpensive transmission of messages, documents, and other items to friends, clients, and co-workers.

Email can be utilised by firms for marketing. Sending newsletters, special offers, and other marketing information to subscribers via email is a well-liked method of connecting with clients and potential clients. Email can also be used as a tool for team collaboration, allowing members to share files, assign tasks, and maintain communication. Email is also a helpful tool for customer service, with many companies using it to respond to inquiries, offer assistance, and address problems. Email can be used for recruitment purposes to contact possible job prospects, post job openings, receive resumes, and arrange interviews.

Email can also be utilised in the classroom, where teachers can send messages to students, provide course materials, and give homework assignments. Finally, a straightforward email system can be utilised as a tool for personal organisation, enabling users to set reminders, create to-do lists, and keep track of crucial dates and events.

**Quality Of Work**

* **User interface:** Simple to use and navigate. Users will be more satisfied as a result, and user mistakes will be less likely.
* **Security:** The email system needs to be safe to prevent unauthorised access to user information. To stop hacking and other security breaches, this entails putting encryption, multiple-factor authentication, and other security measures in place.
* **Compatibility:** Different email clients, platforms, and operating systems should be able to use the email system. Being able to send and receive emails using several email clients and platforms is part of this.
* **Customization:** The email system needs to be adaptable to the particular requirements of different users and organisations. The ability to alter email templates, signatures, filters, and other settings is part of this.
* **Clarity:** The options on the menu should be well-labeled and arranged, and they should be simple to grasp. Users should have no trouble or difficulty finding the information they need.
* **Functionality:** The menu should give users access to all of the email system’s essential features and capabilities, including the ability to create, send, and receive emails as well as control contacts, folders, and settings.
* **Accessibility:** Keyboard shortcuts and other accessibility features should make it simple to explore, and it should be designed to work well on a variety of screens and devices.
* **Sign-In and Sign-Up buttons:** The user should be able to sign into an existing account or establish a new account with a valid email address and password by clicking the Sign-In and Sign-Up buttons. If there are any mistakes or problems with the user’s sign-in or registration procedure, they should also give the user the proper feedback.
* **Compose button:** This button should correctly create a new email composition window with To, Subject, Body, and Attachment choices for creating a new email.
* **Send button:** Send the constructed email to the recipient’s email address successfully and give the user the necessary feedback if the email was sent successfully or not.

**Program Design**

|  |  |
| --- | --- |
| 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 theme uses the tkinter library to create a login system. A theme consists of several structures, including root window initialization, various UI elements, functionality, placement of user-interface (UI) elements, and error handling. The program first initializes the root window and defines its properties, such as size, title, and background colour. Then create and customize UI elements such as labels, entries, and buttons with various parameters. The program defines two functions, “login” and “open\_registration”, which are executed when the user clicks the “Sign in” and “Sign up” buttons respectively. UI elements are placed in the root window using the “place” method, and the placement of these elements is done in a way that provides a good user experience. Finally, error handling is implemented by highlighting the input field in red and displaying an error message box if the entered email address and password do not match any users in the “users.txt” file. Therefore, the program design follows a well-structured approach that ensures a good user experience by providing an easy-to-use and intuitive interface.

The user can enter their personal data in a variety of fields on the registration page, including their first and last names, date of birth, email, phone number, and gender. After the user enters their data, the script verifies that the email address they have provided is valid and not already in use. The user’s details are stored in a file named “users.txt” if the email is legitimate and accessible. A pop-up dialogue box will appear with the relevant error message if the user’s input contains any mistakes (such as an invalid email, a password confirmation failure, or an email that has previously been used). An information message appears, and the registration window closes if the registration is successful.

The inbox application uses the tkinter library to create windows, frames, buttons, and list boxes to display email messages. The Inbox functionality is implemented by reading email data from a text file, analysing the data to extract information such as sender, recipient, and subject, and then displaying the subject line in a list box. The GUI provides options to search emails by keyword, view and delete emails, and log out of the application. The code also includes some functionality that is currently empty or not fully implemented, such as composing and sending email. The script consists of the function definitions above and the main GUI creation code below.

The code is structured as a class called “Email” that contains the necessary properties for an email, such as sender, recipient, subject, message, and attachments. The class also contains a method to format the email as a string and a method to attach 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 takes user input from various tkinter widgets such as input fields and text fields, creates an instance of class “email” with the input value, 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 where the user can 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 in the message text field. This function limits the number of characters in a message to 1000 by removing all characters after the 1000th character.

**User Interface Design**

The interface of this code is simple and intuitive with a smooth and seamless experience. The login screen offers two input fields for email and password, and a clearly labeled header. The login button is located below the input field and is labeled accordingly. Also provides a sign-up button if the user doesn’t have an account yet. The overall UI design is visually appealing with a blue colour scheme and clean typography. The error messages displayed when the input is incorrect are also clear and concise. User interaction is straightforward and input fields are easy to use, allowing users to sign up naturally without external help.

User can 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.

The inbox of the email system user interface 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 and when users enter 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 and user does not need any help to use this feature. It 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.

Users may easily and intuitively compose and send emails using the email composing user interface. Sender information, recipient information, subject, and message make up the interface’s four components. Before sending the email, the user must complete all needed data; otherwise, the code checks for empty fields and shows an error message. A warning notice is also presented, and the extra characters are automatically erased if the message is longer than 1000 characters. The user interface also has a button for email attachments. The “Attach File” button activates a file dialogue box where the user can choose the file they want to attach. If a file is selected, a success message is displayed, and the file is attached to the email.

**Solution**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Description** | **Steps** | **Expected** | **Pass/Fail** |
| 0001 | Registering a new account with valid inputs. | 1. Enter a valid email that has not been used before. 2. Enter a valid password. 3. Confirm the password. 4. Select a valid date of birth, gender, and phone number. 5. Enter a valid first name and last name. 6. Click the “Register Account” button. | “Account registered!” message is displayed. | Pass |
| 0002 | Registering account with an invalid email. | 1. Enter an invalid email that does not contain “@” or “.” characters. 2. Enter a valid password and confirm the password. 3. Select a valid date of birth, gender, and phone number. 4. Enter a valid first name and last name. 5. Click the “Register Account” button. | Error message “Enter a valid email!” is displayed. | Pass |
| 0003 | Registering a new account with an email that has already been used. | 1. Enter an email that has already been used in the “users.txt” file. 2. Enter a valid password and confirm the password. 3. Select a valid date of birth, gender, and phone number. 4. Enter a valid first name and last name. 5. Click the “Register Account” button. | Error message “Email already taken!” is displayed. | Pass |
| 0004 | Registering a new account with mismatched passwords. | 1. Enter a valid email. 2. Enter a valid password. 3. Enter a different password in the “Confirm Password” field. 4. Select a valid date of birth, gender, and phone number. 5. Enter a valid first name and last name. 6. Click the “Register Account” button. | Error message “Passwords do not match!” is displayed. | Pass |
| 0005 | Successful login with an existing user. | 1. Input the email and password of the user that has been successfully signed up before 2. Click the “Sign in” button. | Appears with the message “Sign in successful!” and opens the inbox.py file. | Pass |
| 0006 | “Sign Up button”. | 1. Click the “Sign up” button. | Program should close and the Registration module should open. | Pass |
| 0007 | Unsuccessful login with an incorrect email | 1. Input a non-existing email in the email\_entry field. 2. Input any password in the password\_entry field. 3. Click the “Sign in” button. | Both the email\_entry and password\_entry fields should have a red highlight around them, and a message box should appear with the message “Invalid email or password!” | Pass |
| 0008 | Unsuccessful login with an incorrect password | 1. Input the email of the user in the email\_entry field. 2. Input an incorrect password in the password\_entry field. 3. Click the “Sign in” button. | Both the email\_entry and password\_entry fields should have a red highlight around them, and a message box should appear with the message “Invalid email or password!” | Pass |
| 0009 | Search function returns the expected results when searching for an email. | 1. Click on the search entry field. 2. Type in a search query. 3. Click on the search button. 4. Check that the emails matching the search query are displayed in the listbox. | Emails matching the search query are displayed in the listbox. | Pass |
| 0010 | Search function returns no results when searching for a non-existent email. | 1. Click on the search entry field. 2. Type in a search query for a non-existent email. 3. Click on the search button. 4. Check that no emails are displayed in the listbox. | No emails are displayed in the listbox. | Pass |
| 0011 | Delete button removes the selected email from the list. | 1. Select an email in the listbox. 2. Click on the delete button. 3. Check that the selected email is no longer displayed in the listbox. | Selected email is no longer displayed in the listbox. | Pass |
| 0012 | Compose email function opens a new email window. | 1. Click on the compose button. 2. Check that a new email window is opened. | A new email window is opened, for user to write their email. | Pass |
| 0013 | Sent function displays the expected sent emails in the listbox. | 1. Click on the sent button. 2. Check that the expected sent emails are displayed in the listbox. | The expected sent emails are displayed in the listbox. | Pass |
| 0014 | Email is successfully sent when all fields are filled correctly. | * 1. Enter the sender email address in the “From” field.   2. Enter the recipient email address in the “To” field.   3. Enter the subject of the email in the “Subject” field.   4. Enter the message in the “Message” field.   5. Click the “Send Email” button. | A success message box is displayed saying “Email sent successfully.” | Pass |
| 0015 | Warning message is displayed when message length exceeds 1000 characters. | 1. Enter the sender email address in the “From” field. 2. Enter the recipient email address in the “To” field. 3. Enter the subject of the email in the “Subject” field. 4. Enter a message in the “Message” field with more than 1000 characters. 5. Click the “Send Email” button. | Warning message box is displayed saying “Message exceeds 1000 characters.” | Pass |
| 0016 | Warning message is displayed when a field is left blank. | 1. Leave one of the fields (sender, recipient, subject or message) blank. 2. Click the “Send Email” button. | Warning message box is displayed saying “Please fill in all fields.” | Pass |
| 0017 | Attachment is successfully added to the email. | 1. Click the “Attach File” button. 2. Select a file to attach. 3. Click the “Send Email” button. | A success message box is displayed saying “Email sent successfully.” | Pass |
| 0018 | Warning message is displayed when no file is selected to attach. | 1. Click the “Attach File” button. 2. Close the file dialog without selecting any file. 3. Click the “Send Email” button. | Warning message box is displayed saying “No file selected.” | Pass |

**Suggestions For Improvement**

A password hashing method might be added to the “users.txt” file to save passwords, and a validation function should be used to verify the user’s email address and password before comparing them to the information in the “users.txt” file. Another validation might be included to make sure that the email addresses given in the sender and recipient fields are genuine and that the subject and message boxes are not empty. Error handling for file I/O operations should also be taken into consideration, such as handling for file not found or permission issues. To further guarantee that users submit accurate data, a different kind of input validation should be included. For example, it could be checked that the phone number is formatted correctly or that the user’s birthdate is not in the future. It is also possible to incorporate password strength checking to make sure that users provide secure passwords. A validation function should make sure that the email is formatted correctly, and that the password satisfies the minimal standards for strength, which include including both uppercase and lowercase letters, digits, and symbols. A “Forgot Password” option that enables users to change their passwords by having a password reset link sent to their email should be included in the programme in order to make it more user-friendly.

The existing design should be changed with a more appropriate, eye-catching, and contemporary theme in order to improve the GUI. To support various, internationalisation can also be introduced. It should be possible to search for text both inside email fields and within the email’s body thanks to better search capability. By including capabilities like the option to sort emails by date or sender and by making the layout more user-friendly, the GUI might also be made better.

After an email is sent, real-time updates may be performed to refresh the inbox and display the new email. The ability to attach multiple files should be added to the file attachment function, and it should also be modified to ensure that the connected file is not too huge, giving an error notice if it is.

**Conclusion**

The programme appears to be well-structured, legible, and has a straightforward user interface. The code appears to have been written clearly and effectively, and it employs appropriate error handling to alert the user to any incorrect inputs. There are a few places, though, that may use improvement. For instance, it is not safe to save the password in plain text. Using a more secure method to store passwords on the database is advised. More colours and visuals might also make the user interface more aesthetically pleasing.

I discovered that writing a programme of this nature needs a thorough knowledge of programming ideas, data structures, and algorithms. The programme also emphasises the significance of meticulous input validation and error handling to make sure the programme is capable of gracefully accepting unexpected inputs and faults. It’s impossible to predict gains without knowing the program’s exact implementation specifics. It could be advantageous to incorporate practical algorithms or functions that could improve the program’s efficiency and functionality. I consider the email client programme to be a straightforward and well-designed tool that enables users to send and write emails fast without using a more complicated email client. The Email system, is a great tool for organising and formatting email data. As a result, the code for composing and sending emails is cleaner and simpler to comprehend. I especially value how the ‘messagebox’ and ‘filedialog’ modules were used to create a user-friendly interface.

The email client programme can be improved in certain areas, though. For instance, other features might be introduced, such the ability to see sent messages and support for multiple email accounts. Additionally helpful would be email address input validation, particularly for the sender and recipient sections. It could also be beneficial to include a function that enables users to store their email as a draught so they can continue changing it and send it later. A more reliable email sending system that enables sending emails over SMTP servers may also be taken into consideration.

**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