Table of content

|  |  |  |
| --- | --- | --- |
| Section | Content | Page Number |
| 1 | Introduction | 2 |
| 2 | Quality of work | 3 |
| 3 | Program Design | 3 |
| 4 | User interface design | 6 |
| 5 | Solution | 8 |
| 6 | Suggestions for improvement | 9 |
| 7 | Conclusion | 10 |

Don’t forget to attach the database file with the thing already created email2 lol

1. **Introduction**

The email program is designed to allow users to compose and send emails, as well as view their inbox. The program allows users to input their login email address and password, as well as the recipient's email address, email title, and email content. Once the user clicks the "send" button, the email is stored in a SQLite database within an ‘email’ table and ‘sent’ to the recipient. Details of this feature will be explained below.

The purpose of this laboratory is to learn from the process of writing a complete program that resembles some basic principles of an email system. The language used is Python 3.0 and the method of storing and writing the data to a database is either through excel (csv) or structured query language (SQL). In this case, the SQLite3 was used, and python code was written to assist in the writing of the data to be present in the database. The program’s base functions include, allowing the user to:

* Register a user by allowing them to create and save their email and password
* Login in the user
  + Logged in user can compose emails to send to other users
  + Logged in user can view emails sent by other users
  + Logged in user can ‘logout’
* Compose email feature
* View inbox feature

Each of these functions will be discussed in depth throughout the rest of this report. Along with future ideas which will hopefully be implemented later. Screenshots of the code will be included in this document. A demonstration video and the source code files will be included as separate attachments to the submission.

1. **Quality of work**

The program meets all specified functionalities such as a competent GUI, a pre-made database, a register and login system, a compose window and an inbox system. These features are also checked and tested to work without errors. The buttons and menus are big, clear, and easy to use, additionally, the windows are reasonably sized and even the display boxes and colour are easy to use. However, there are a few minor improvements that can be made, which will be discussed in a later section.

There are some important things to be made aware of before testing the program and evaluating its integrity. Firstly, the program is constructed to read and write to and from an SQL database. An empty database with premade tables, complete with defined fields will be included in the submission. A potential improvement here will be discussed in the suggestions section. The code file and the database file must be in the same folder for the code to run flawlessly. A third-party software such as DB Browser for SQLite can be used to view the table and the changes it would go through.

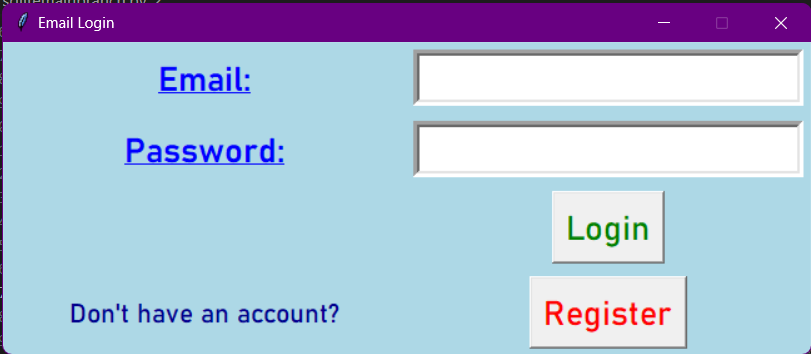
1. **Program design**

The program is well-designed and uses appropriate structures. The program uses a SQLite database to store the email information, which is an appropriate design choice. A brief description of the program’s intricacies will be described below:

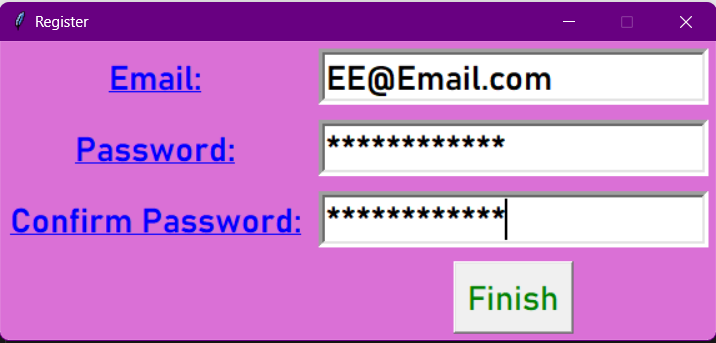
1. The program begins with the first block of code being the import lines for the packages which is then followed by the connection function which is necessary to connect to the database. The connection function also includes a try-except block to catch an potential and unwanted errors.
2. When the program is run, the first set of code to be executed would be the Login window code and the connection code.
   1. The login window is a comfortably sized window with two input boxes for entering the user’s email and password if they are registered. A Login button is present to run the login function. Below, a register function in case that the user does not have a registered email in the database.
      1. The register button, once clicked, opens another window with a different colour scheme and has fields that are required to create another user profile. Once the user fills in the fields and presses the finish button, the data is read from the entries and inserted into the ‘users’ table in the database for future logins.
         1. It is worth noting that these windows are locked to a resolution size and cannot be resized for cosmetic purposes to avoid unnecessary expansion
         2. It is also worth noting that the email entry fields have data validation to prevent the entry of any data that doesn’t have an ‘@’ symbol and a ‘.com’ at the end. The field is also unique to prevent duplication entries and has case-sensitive entries. The user is prompted to enter a valid address again by a message box upon an unsuccessful effort.
         3. There is also a validation check to make sure that the password and confirm password fields are identical to ensure mistakes are voided in user creation.
         4. Upon a successful registration, the INSERT command is used to insert the data into the ‘users’ table in the database. A message is then displayed, and the registry window is closed, and the login window is focussed for re-entry.
      2. When the login button is clicked, the program makes a check, using the SELECT command, with the database to see if the inserted email and password match an entry in the database. If it does, successful login is issued, if not, an error message box is shown prompting the issue to the user.
      3. Once entering the correct login details, and after a message is displayed. The ‘email’ window is displayed. This window’s ability to resize is also locked and three buttons are displayed.
         1. A simple logout button with red font basically completely terminates the program and refreshes the program to be restarted by the user again for another attempt at testing or whatever the user desires.
         2. A compose button which contains the sending email function and GUI.
         3. An inbox button that opens the inbox window for the user to view received emails.
      4. If the user wishes to send an email to another user, the compose button can be used. Once clicked, a large compose email window is displayed to the user. Three fields are present allowing the user to enter the recipient’s email address (has data validation), it also allows the user to enter a subject and then the content and bulk of the email. A ‘send’ button is present at the bottom which reads the entries of the user’s boxes and inserts them into the ‘emails’ tables under respective columns. A scrollbar is present in the content section for particularly large emails.
         1. The compose feature automatically fills in the sender’s email address column with the logged in user’s address to make the user’s experience easier and less confusing.
      5. If the user wishes to view the emails sent to them by other people, the inbox button can be used. If the user is a new email address and doesn’t have any emails sent to them, an empty inbox is displayed with a message.
         1. However, if emails are present, they are displayed in the forms of buttons which have the sender’s email as their button text.
         2. These buttons once clicked, will open another window that contains the contents of the email which include: the sender’s email address, the desired recipient email address, the subject and the content and text of the email.
         3. This is possible due to the SELECT command of SQL, which is used to filter out all the emails except the ones where the recipient address matches the logged in user’s address. It then attaches the respective row of data to a button and when clicked, displays the data in a pretty window for easy reading. Even when viewing the emails, the sender’s address is automatically read as the logged in user and inserted during composition without user input.
      6. Once the user has finished their business with the program, they can close the respective windows to continue other tasks or press the logout button (which uses sys.exit, that stops debugging) to close everything all at once and conclude the session.
3. **User interface design**

The user interface is clear and easy to use. Users can easily input their information and send emails without difficulty. Message boxes with helpful and instructive text are deployed when errors (usually user induced) are encountered, allowing the user to more easily use the program in a seamless manner without too much trouble or getting stuck. Thought was put into the choice of colour to make it as user friendly as possible regarding readability and visibility. Big buttons and subtle fonts were used to enhance the comfort of the user’s experience and ease. Below are screenshots of the code and GUI of the program.

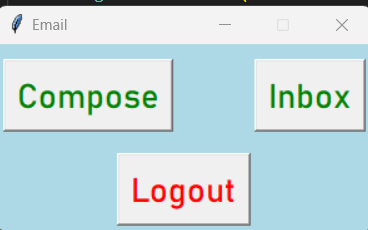
**Login window:**



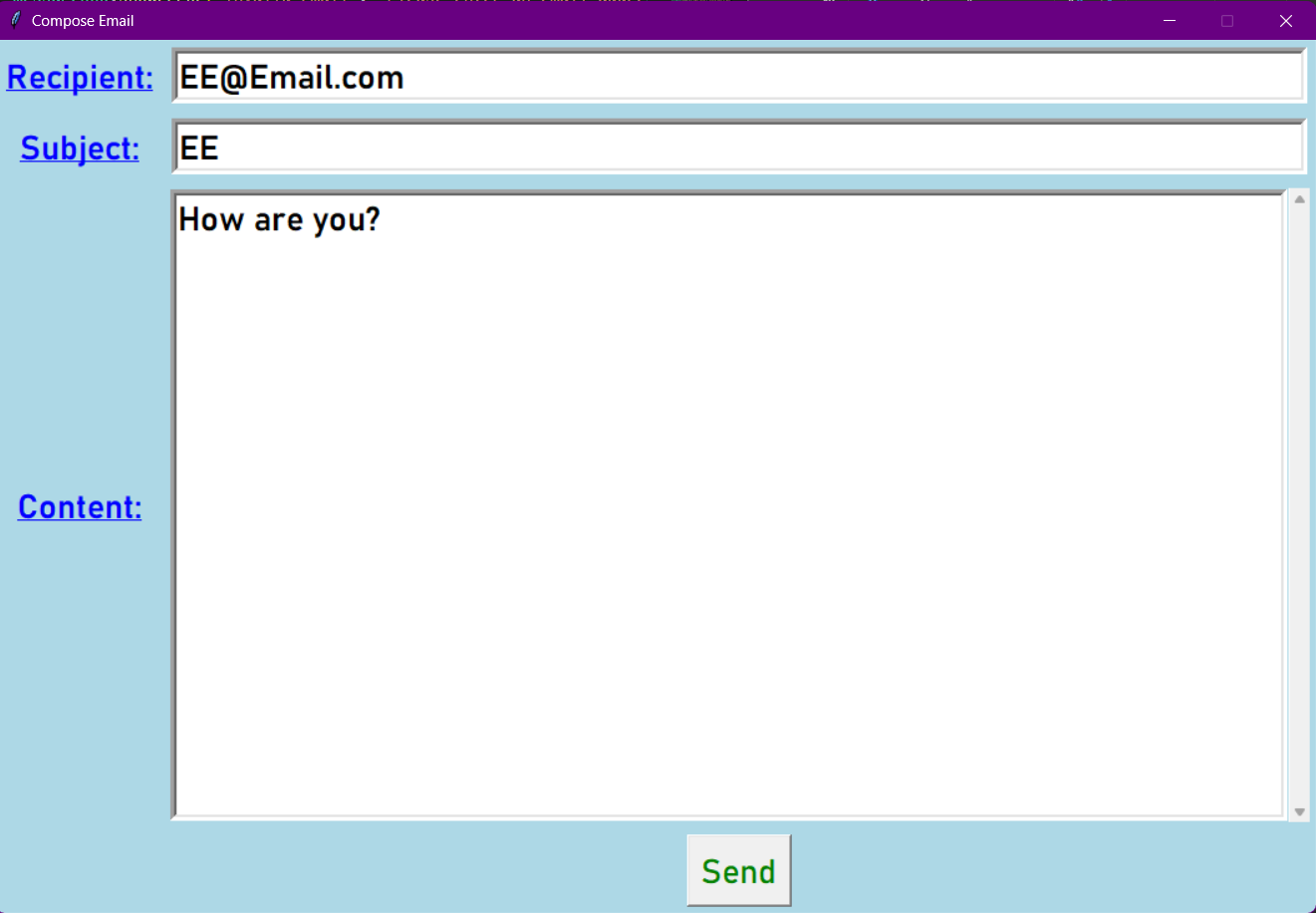
Register window:



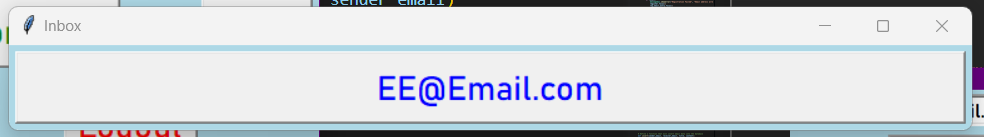
Email home window:



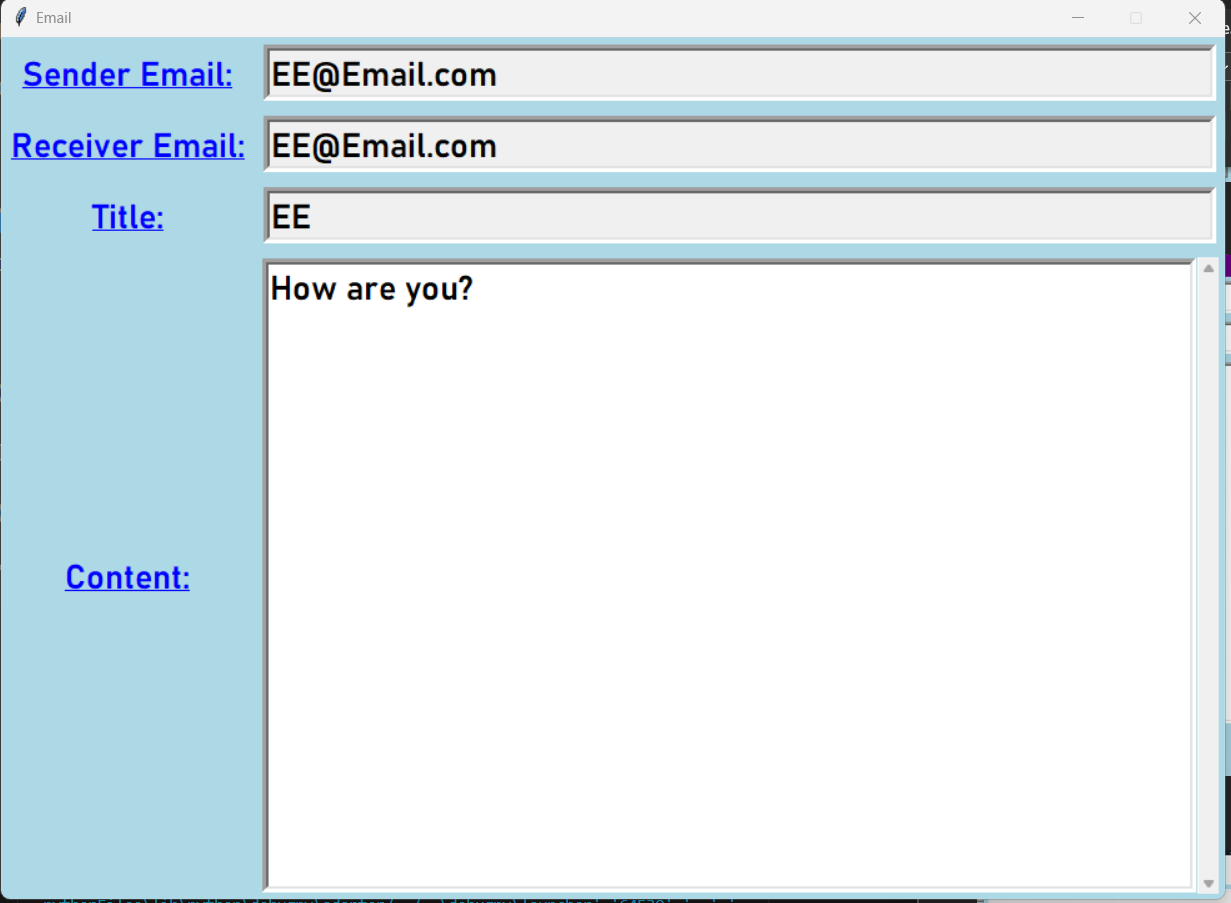
Compose window:



Inbox window:



Email window (example):



1. **Solutions**

The program is complete and can run without errors. The program successfully stores emails in a database and sends them to the recipient's email address. It passes on test data shown below:

|  |  |  |  |
| --- | --- | --- | --- |
| Test no. | Test problem | Outcome | Result |
| 1 | -Insert a wrong format in any of the entries that requires an email entry:  Test format: 6Days | -Message box pops out prompting the user to input a valid email address format and gives an example of a valid one. | Pass |
| 2 | -Insert different passwords during registration  Test:  EE  & E | -Message box pops out prompting the user that the passwords do not match. | Pass |
| 3 | -Attempt to login with a non-existent email  Test: EE@Email.com | -Message box pops out informing the user that the email or password is wrong | Pass |
| 4 | -Creating a registered user, logging in with registered user, composing an email that is sent to the same address(self) and viewing the sent email in the inbox  Test:  [Test@gmail.com](mailto:Test@gmail.com)  Password: Test  Subject: Test  Content: Test | -Completely successful with all steps being fluid and showing no errors | Pass |

1. **Suggestion for improvement**

Many improvements and potential additional features were entertained throughout the construction of this program and if time had allowed it, the possibility of implementing this feature would have been a more realistic goal. They are nonetheless mentioned below for the sake of future improvement.

Future ideas:

* However, the program does not use any advanced data structures, such as linked lists or trees, which could potentially improve the program's efficiency.
* Inclusion of a confirmation message when the email is sent successfully. This would provide feedback to the user and confirm that the email was sent.
* Implementing advanced features such as auto complete or suggestion boxes could improve the user experience.
* Fix the focus feature of tkinter to pull up the respective window for re-entry.
* Have an alternative to closing the other windows besides using the close tab button

1. **Conclusion**

Overall, the email program is a well-designed and functional solution. The program effectively stores and sends emails without errors. The program could be improved with a few additional features, but it meets all specified functionalities. The task provided a valuable learning experience and could potentially be improved by adopting advanced features from other email programs.

In conclusion, this is a simple program with a simple and friendly GUI. The program can read and write to an existing database in SQLite. It is able to log users in and allow them to compose and view emails. The code also has some basic error handling and data validation for specific fields and circumstances to improve realism and interaction quality for the user. The simple email program could be improved by adjusting existing functions with advanced features or by introducing new functions for advanced features or missing basic ones such as a confirmation feature before sending emails.