

February 5, 2019

Matthew Karko

Mr El-miski

Mailit

Software Major Project

Table of Contents

[Stage 1: Defining the Problem 4](#_Toc10491730)

[Description of the problem 4](#_Toc10491731)

[Solution 4](#_Toc10491732)

[Idea Generation 4](#_Toc10491733)

[Idea 1 4](#_Toc10491734)

[Idea 2 4](#_Toc10491735)

[Idea 3 4](#_Toc10491736)

[Idea selection 4](#_Toc10491737)

[Boundaries for the Software 5](#_Toc10491738)

[Gantt Chart 6](#_Toc10491739)

[Existing solutions 7](#_Toc10491740)

[Social and ethical considerations 7](#_Toc10491741)

[Current Solutions 7](#_Toc10491742)

[Cost effectiveness 7](#_Toc10491743)

[Licensing considerations 7](#_Toc10491744)

[Software Development Approach 7](#_Toc10491745)

[Design specifications 8](#_Toc10491746)

[List of specifications 8](#_Toc10491747)

[Developers Perspective 8](#_Toc10491748)

[Users perspective 8](#_Toc10491749)

[System Documentation 10](#_Toc10491750)

[Context Diagram 10](#_Toc10491751)

[IPO Diagram 11](#_Toc10491752)

[Data Dictionary 11](#_Toc10491753)

[Storyboarding 12](#_Toc10491754)

[Dataflow Diagram 16](#_Toc10491755)

[Structure Diagrams 0](#_Toc10491756)

[Communication issues 0](#_Toc10491757)

[Communicating with the client 0](#_Toc10491758)

[Quality assurance 0](#_Toc10491759)

[Guaranteeing a quality product 0](#_Toc10491760)

[Judging the quality of the product 1](#_Toc10491761)

[Stage 2: Planning & Designing Software Solutions 2](#_Toc10491762)

[Standard Algorithm 2](#_Toc10491763)

[Custom-designed logic used in software solutions 2](#_Toc10491764)

[Psuedocode 2](#_Toc10491765)

[Flowcharts 5](#_Toc10491766)

[Standard libraries of code used 8](#_Toc10491767)

[Test Data 9](#_Toc10491768)

[Documentation of the overall software solution 10](#_Toc10491769)

[Context Diagram 10](#_Toc10491770)

[IPO Diagram 11](#_Toc10491771)

[Data Dictionary 11](#_Toc10491772)

[Storyboarding 12](#_Toc10491773)

[Dataflow Diagram 16](#_Toc10491774)

[Structure Diagrams 0](#_Toc10491775)

[Interface Design in Software solutions 0](#_Toc10491776)

[Issues that affect interface design 0](#_Toc10491777)

[Interface storyboard consideration 1](#_Toc10491778)

[Factors to be considered when selecting the programming language to be used 5](#_Toc10491779)

[Factors to be considered 5](#_Toc10491780)

[Factors to be considered when selecting the technology to be used 6](#_Toc10491781)

[Performance requirements 6](#_Toc10491782)

[Benchmarking 6](#_Toc10491783)

[Stage 3: Implementation of Software Solutions 7](#_Toc10491784)

[Techniques in developing well written code 7](#_Toc10491785)

[Good programming practice 7](#_Toc10491786)

[Errors faced 7](#_Toc10491787)

[Methods to detect errors 9](#_Toc10491788)

[Software debugging tools 10](#_Toc10491789)

[Documentation of the overall software solution 11](#_Toc10491790)

[User documentation 11](#_Toc10491791)

[Technical documentation 11](#_Toc10491792)

[Hardware requirements 12](#_Toc10491793)

[Testing the software solution 12](#_Toc10491794)

[Evaluate success of the project 12](#_Toc10491795)

[Test data use 12](#_Toc10491796)

[Post implementation review 14](#_Toc10491797)

[Discussion with client 14](#_Toc10491798)

[Modifying code to meet changed requirements 14](#_Toc10491799)

[Changes that may need to take place 14](#_Toc10491800)

[User Documentation Modification 15](#_Toc10491801)

# Stage 1: Defining the Problem

## Description of the problem

The problem that I’m going to solve with my code solution is one to do with the late arrival of students. When a student is late the Office ladies get notified and they must send out a message to all the parents manually. This is a very lengthy process for the office ladies and depending on other jobs, may be very frustrating for them to organise for that to happen. What the office ladies want is for this to be automated to reduce stress and unnecessary time consumption.

## Solution

My proposed solution is a piece of software that will automate the part sending all the notifications to all the parents.

## Idea Generation

### Idea 1

An idea that really intrigued me was being able to remotely access your home pc from your phone anywhere in the world. Currently there are many solutions, but they have many requirements such as port forwarding and having a static IP and so on. This means that the process is very hard and is a major turn off for everyday technology users. This was a very achievable design but I came to the conclusion that it wasn’t a viable project.

### Idea 2

This idea is about facial recognition and finger print scanning to enter houses, and access garages and so on. This allows for an extra layer of protection towards your belongings and can be very useful in many situations. Keys should be a thing of the past and we should be moving forward as we are with our phones. Although, it does raise some alarming issues with privacy and security. This can be due to people being to hack into your house.

### Idea 3

This idea is the most realistic and the most secure for privacy reasons. With the current late system, you come in and the office ladies take your name and you then go to class. Now they will have to find the relating emails and place them in a new column. The file will get uploaded to my code and from there my code will handle emailing all the parents. The GUI will have the option to give the file, double check the list and then you will be able to send the emails.

## Idea selection

I have decided to go with Idea 3. This idea is the most secure for the student’s information and it is the most achievable idea. This being that the other 2 require an excessive amount of knowledge and practice with the programming language. Choosing this idea means that I will also be able to focus more on the key areas such as the GUI for the program and understanding the client’s needs.

## Boundaries for the Software

There are all types of boundaries when it comes to my program. There are many security issues when it came to choose between the ideas as ideas 1 and 2 required access to the school’s database which can reveal a lot of information if were in the wrong hands. I also did research into messaging the parents as well as emails but there were too many complications including owning external hardware and being able to code with a different extension. This would also increase the cost of the product. The software can only read excel formatted documents. This is also a setback but when I discussed this with the office ladies they said that the information that they have is already in an excel spreadsheet.

## Gantt Chart

## Existing solutions

### Social and ethical considerations

There are no existing solutions that suit the needs of the office ladies and comply with the Department of Education’s rules. Although there are some solutions that can do the job, but there a lot of ethical issues and considerations that the school has to think about. For example, allowing a 3rd party outsourced solution have access to personal information of all the students at the school raises a lot of issues. There a lot of social considerations when it come to the students and the parents, they may see these current solutions as being dangerous and intrusive and they may wish to not participate in the program and this can be an important issue for the school.

### Current Solutions

* Socketlabs
* Zoho
* Sendgrid
* Icontact

With these current solutions there are not many customisation options with them as you do not have full access to the code, meaning you did not create it yourself. This means that you cannot add extra features later and update the solution yourself. Whereas with my code, we have full access to it and we can customise everything.

### Cost effectiveness

When outsourcing the solution, they charge you per email you send, and this can be anywhere from 3 cents all the way to 20 cents. Alternatively, with my solution there will be no cost, which will make it a very cost-effective solution. Although later on through development there may be costs introduced if we decide to put it onto a server and have new features such as automated text system as well.

### Licensing considerations

When outsourcing the solution there a lot of licensing terms that the school will have to follow due to it being someone else’s solution. These terms may include upfront payments to be able to use the program in the first place, and then payments every year or so. There also may be terms that the school must abide by and risk penalties if not followed.

## Software Development Approach

I have researched and taken into consideration all the aspects of designing and creating the code and I have evaluated all this information, from this I have decided to go with the agile approach to coding. Doing an agile approach allows me to do a basic build of the code and continue improving the code, it gives me more time to code and add new features as I go along. It also allows me to be very flexible with the code, for example if I was doing structured and say I wanted to change something about the design, I wouldn’t be able to change it easily and I would have to go back and redesign the whole code. Whereas with agile, it will allow the code to be very flexible which is one of the main reasons why I decided to go with agile. It also allows me to focus on the needs of the clients and their values. Improving quality will also will be very easy and accessible, whereas in the structured approach your design is what is said to be the best version of the code, so any new ideas that come up will become a hassle to add in. With agile, we are also able to test the code earlier and we can test certain parts which allows us to find errors with the code early on. Another key point that agile approach focuses on is business values (bang for your buck) which then reduces the overall cost of your product.

## Design specifications

### List of specifications

* Be able to read excel files
* Have a GUI
* Have a send button
* Be able to send emails to several people
* Be able to read and search files

This program will not require much hardware power. It requires connection to the internet and will require a CPU of any power, a monitor and minimum of 2GB of ram.

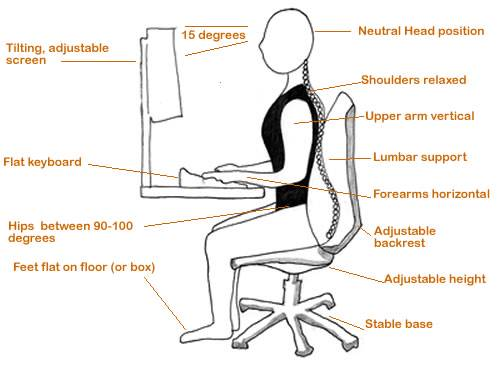
### Developers Perspective

The data types that will be used in my code are of several formats. It ranges anywhere from strings to integers to even excel spread sheets. This allows for the code to be a lot more compact and for the information to be stored in other files where it is accessible. This also means that the code will depend on these files but shouldn’t be a problem.

### Users perspective

#### Ergonomics

For users, if they follow the recommended ergonomics as shown below for the use of a computer or laptop then they should be fine as my solution does not require for any external hardware besides a mouse and keyboard.



#### Hardware

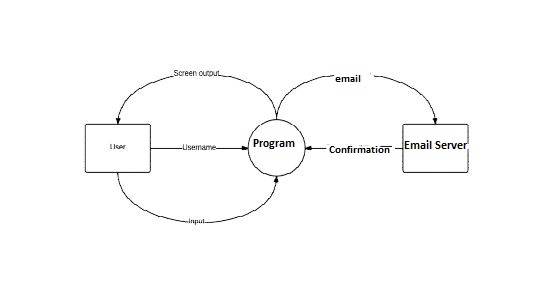
My solution does not require very powerful hardware. A basic configured pc bought from a shop will run the program just fine as it is a very minimalistic program. But it will require input devices such as the keyboard and the mouse.

#### Existing knowledge and skills

The skills required to be able to run and use the software solution are ones that come and are required in regular PC use. Mainly, if confident in the operation of Windows OS then the solution then the use of the solution will be fine.

## System Documentation

### Context Diagram



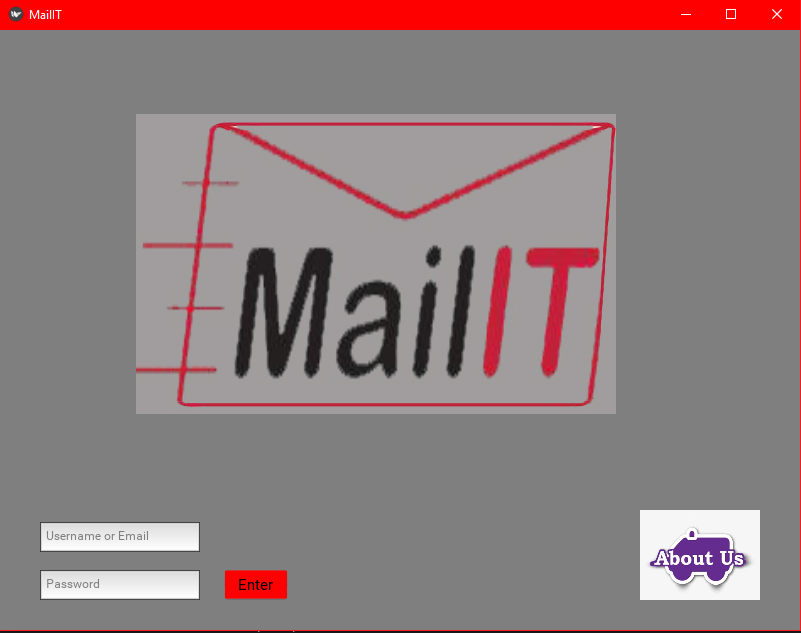
### IPO Diagram

|  |  |  |
| --- | --- | --- |
| Input | Process | Output |
| Enter | * Move to next TextInput * Activate button functions | * Add a error message if the input is invalid * Display the TextInput |
| Buttons are pressed | * Assigned functions for the button * Have multiple functions | * Print out what function is being executed |
| Email Sender | * Sends email to the GMAIL Server * Forwarded to recipients | * Dialogue box saying emails has been sent |
| About Page | * Opens up a new page * Calls image | * Image is displayed on the page |
|  |  |  |

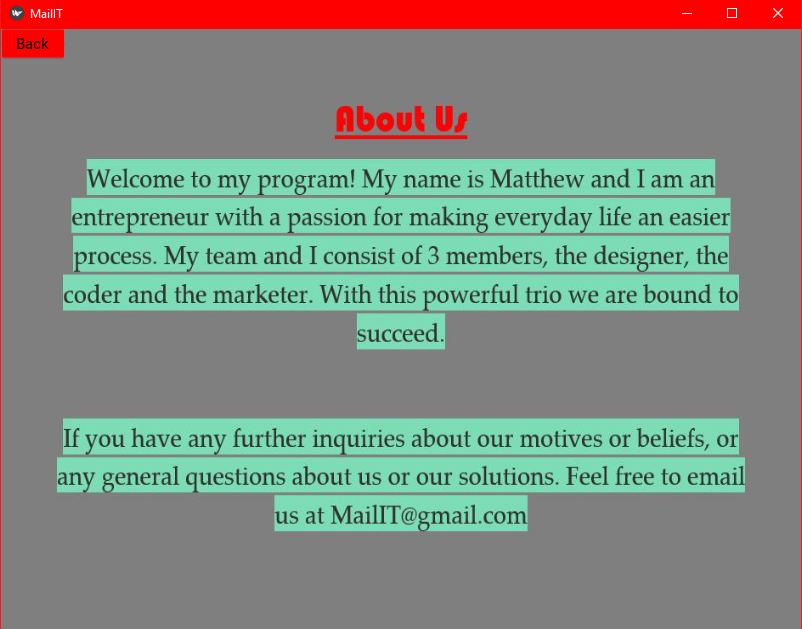
### Data Dictionary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identifiers | Data type | Scope | Description | Example |
| Allnames | Array | Public | Array of student names | Ahmed |
| Em | Array | Private | Array of student names | Michael |
| Len(allnames) | Integer | private | Length of an array | 23 |
| self.emlist1.text | String | Private | A string that is printed | All correct |
| self.emlist.text | String | Private | A string that is repeated | Hello how are you |
| self.box=FloatLayout() | Float | Private | The size of the screen | 400x400 |
| namez | Array | Private | List of inputted names | Jeff |
| e | File | Private | Opening the file to read it, reading it | “Database.xlsx” |
| Self.namelist.text | String | Private | Printing a string | “Hello world” |
| Df | Array | Private | Array of database names and emails | Smith, [s@gmail.com](mailto:s@gmail.com) |
| names\_text\_input | String | Public | Inputted names | Mark |
| Self.printnames | String | Public | Printing names on screen | Matthew  Mark |

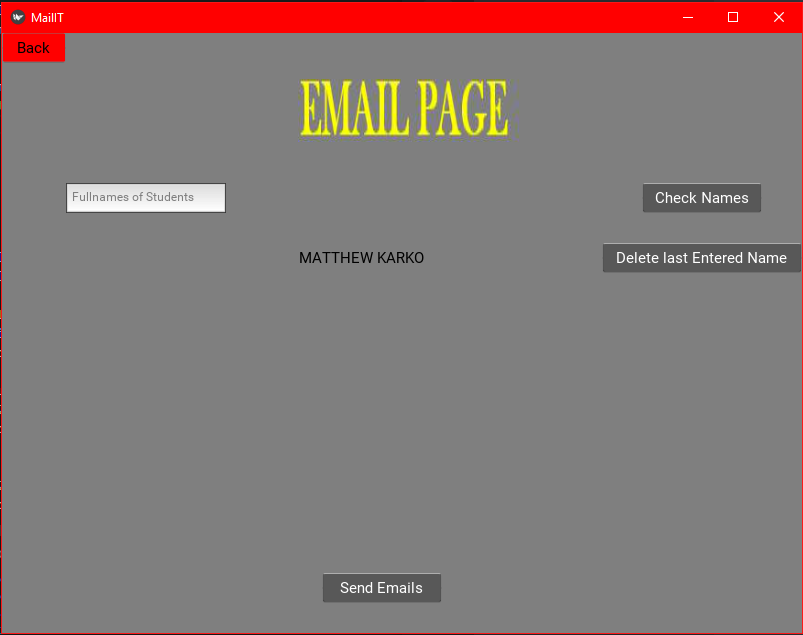
### Storyboarding



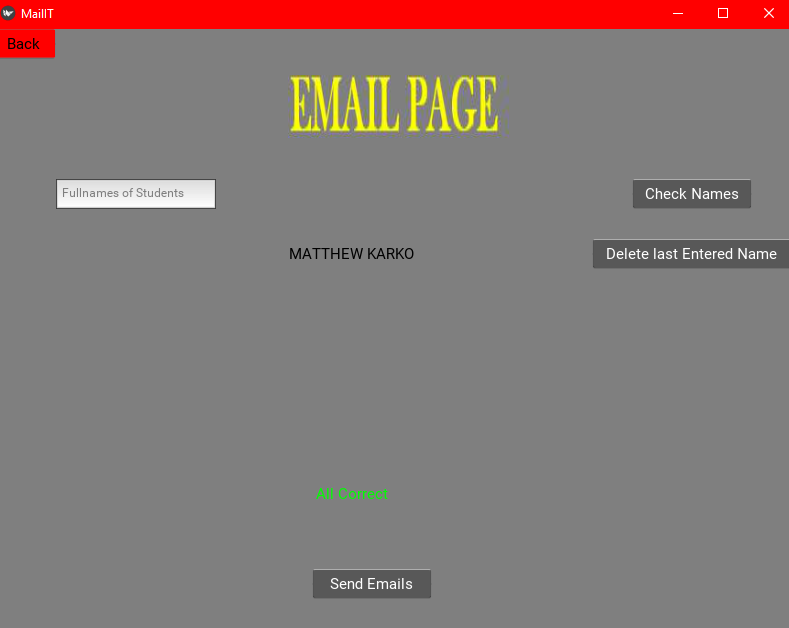
This is the login/introduction screen of the application. There is 2 Text inputs, 2 buttons and the logo of the program. The ‘About Us’ icon leads to a page explaining the people behind the project, their intentions and some characteristics about then. After you enter the correct details and press the ‘Enter’ button it will lead you to the email page.



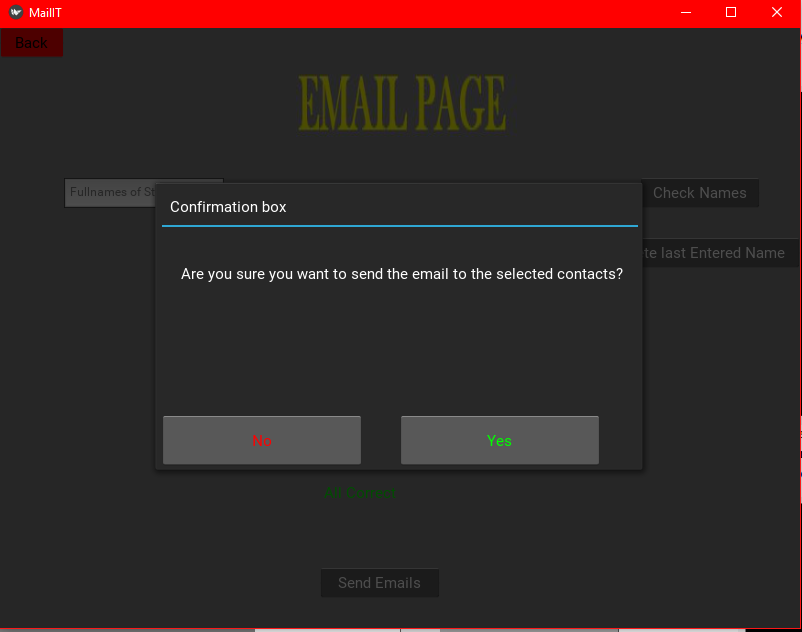
This is the about us page. There is a brief description of what the team is all about and how they came up with the idea for the program. Users are also able to press the back button that leads to the login screen.



This is the email page. It has one Text Input for the names of the students. Then it has 3 buttons all doing different funcitons. The Check Names button gives the user the option to double check his entries into the program. It also has a TextDisplay, being the name list that is entered.

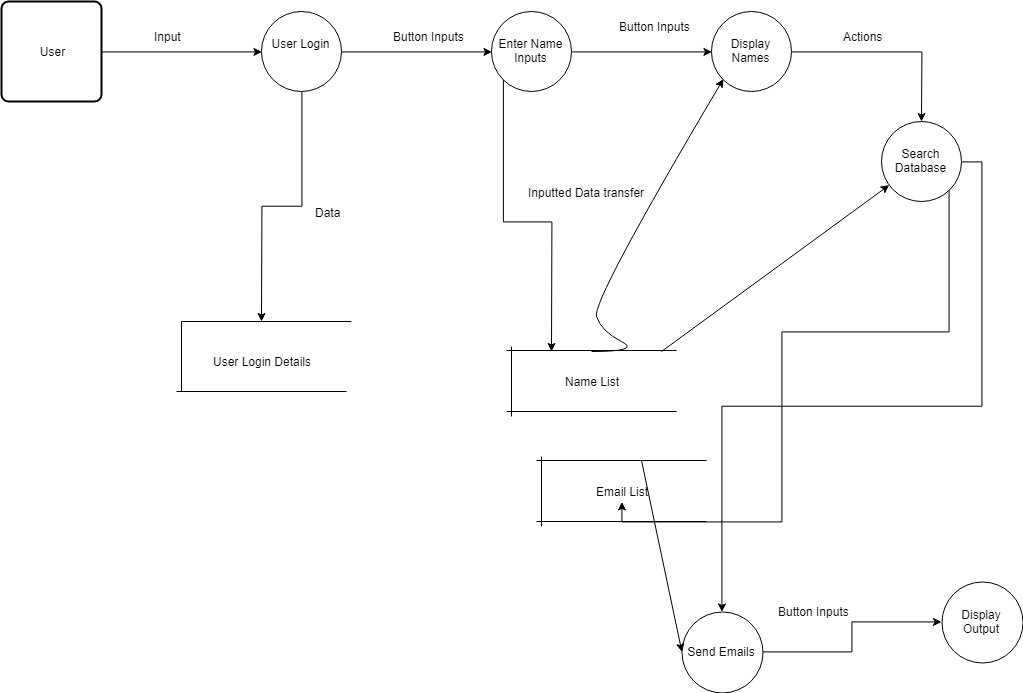
After the Check Names button is pressed another TextDisplay appears as such: 

After the Delete last entered name button Is pressed, the last entered name is removed from the name list. However when the Send Emails button is pressed a pop up shows ensuring with the user that they are 100% sure about sending the emails to the entered recipients.



The colour coding is done to connect the decisions on a psychological level.

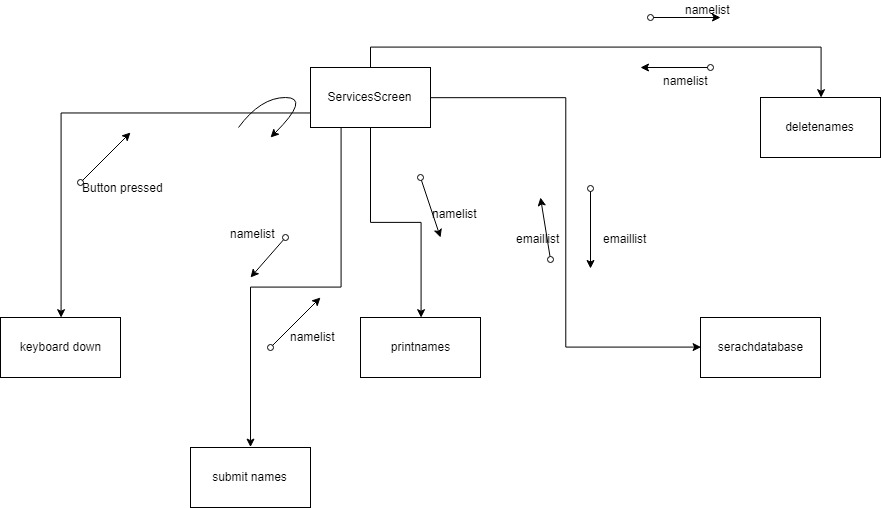
### Dataflow Diagram



### Structure Diagrams

#### LoginScreen

#### ServicesScreen



## Communication issues

### Communicating with the client

Communication between the client and the developer during the project is a very important aspect to forwarding the process of the solution. Having good communication with the client will allow for a better understanding of the problem and issues they face whilst also understanding the clients needs. During the creation of the solution, having communication and feedback between you and the client is very important to develop the best version of the solution possible, catering to their needs.

To ensure that I can maintain good communication with my client, I will:

* Interview my client before starting the production to ensure I understand their needs
* Organize weekly meetings with my client to discuss current work, issues that are being faced and future work
* Always send new improvements made to the solution (i.e. prototypes)
* Always keep them updated and posted on what work I am doing and if they have any suggestions as to what I can add or remove

Having good communication between you and the client allows them to formally address any issues that they have with the solutions. The connection between the developer and the client is very important as it will allow for easier communication. I will provide all the information necessary to them, such as the GANTT Chart.

## Quality assurance

### Guaranteeing a quality product

To ensure the quality of a product, it must be tested thoroughly to make sure they are no issues with the coded solution. The program must meet a certain criterion of specifications when testing in order to be considered a quality product. The criteria for this project are as follows:

* Interactive GUI
* Inbuilt file explorer
* The code will read and analyse information given in the file
* Will feature an inbuilt local database (with sample information)
* When a user accesses my program, when first launched they will be prompted with a login page
* The instructions will be very clear and well set out
* Uses python 3.0 and Kivy
* When the user activates the send message button, it will send the message to the list of people provided
* User will control all aspects of what emails get sent out
* The code will automatically connect to the server and login with an email account, and bulk send the emails to the listed recipients

By following these criteria and making sure all outcomes are met, then it can be assured that the project will be of high quality.

### Judging the quality of the product

The main judgement of quality is based on whether the product meet or excels comparing to the original specifications. These checks are done during the development of the program and at the final stage. The judgement can only be accurate if the you have gained a competent understanding of the clients needs and their requirements.

In the case of this project, the requirements are very achievable and broad. After investigating and researching on what to include in the solution, I understood for the project there are a few easy to complete requirements. These requirements are ones that the syllabus outlined as good practices the requirements are:

* Document your development process following the software development life cycle
* Ease of use
* Efficient
* Reliable
* Design and develop a software project from an idea into a fully implemented software program
* Runs as expected
* Testable
* Understandable
* Easy to be maintained

These outcomes are found on the marking criteria given with the project, however these outcomes are extracted directly from the syllabus.

When developing the program, there must be a standard of usability to ensure that the project will function to the requirements. The standard of usability can be testing by multiple methods including a desk check. However, for this program it will be using a benchmark to compare the expected theoretical result to the actual result exerted from the program. This will allow for the program to detect if it will work on the given device.

# Stage 2: Planning & Designing Software Solutions

## Standard Algorithm

Standard algorithms are sets of code that are very useful and code that is used regularly through your code. There are a number of standard algorithms included in this project. A few of these standard algorithms are displayed below:

* Connecting to email servers and sending emails, *smtplib* class
* Reading from an excel file, *Pandas* class
* Read and correlate information in an excel file, *Dataframe* class
* Opening the excel files, *xlrd* class

The smtplib class uses standard logic of sending data over the internet via email. The smtplib class is responsible for taking in data through parameters in its constructor and sending that data to the Gmail server, which forwards it to the inputted emails. The smtplib class uses the smptlib library to run certain modules of code.

The Pandas class uses a large amount of standard logic. Its main purpose is to read data from an Excel file, pass it into an array separated by commas. Those variables are then able to be accessed from other objects through accessors.

The Dataframe class uses standard logic reading and comparing the files objects to ones that are given by the user. The Dataframe class opens the Excel file and selects the information that you want to select and is stored to then be used later on to compare to the information given by the user.

The xlrd class allows the project to understand the .xlrx extension (Excel file) and thus allowing my program to function. This is the foundation class that allows my code to run.

## Custom-designed logic used in software solutions

### Psuedocode

BEGIN searchdatabase(self)

self.emlist.text = ''

self.emlist1.text = ''

nl = allnames

e = pd.read\_excel("Database.xlsx")

df = pd.DataFrame(e, columns = ['Names', 'Emails'])

search = df[df['Names'].isin(nl)]

em = search['Emails'].values

print(em)

if len(em) == len(allnames)

self.emlist1.text = "All Correct"

else

self.emlist.text = "Error! One of the names was misspelt or not a Student"

END searchdatabase(self)

BEGIN deletename(self)

if len(allnames) > 0:

del allnames[-1]

print(allnames)

else:

print("There are no names left to remove")

END deletename(self)

BEGIN submit\_names(self)

namez = self.names\_text\_input.text

allnames.append(namez)

print(namez)

print(allnames)

self.printnames()

END submit\_names(self)

BEGIN \_on\_keyboard\_down(self, instance, keyboard, keycode, text, modifiers)

if self.names\_text\_input.focus and keycode == 40:

self.submit\_names()

self.names\_text\_input.text = ''

self.names\_text\_input.focus = True

END submit\_names(self)

BEGIN loginscript(self)

if self.room\_input.text == "" or self.name\_input.text == "":

print("Username and Password Cannot be blank!")

accounts = open("details.txt", "r")

accountsr = accounts.read()

accounts.close()

accountsr = accountsr.splitlines()

enter = False

for i in range(len(accountsr)):

up = accountsr[i].split(":")

if self.room\_input.text == up[0]:

if self.name\_input.text == up[1]:

enter = True

if enter == True:

print("Success")

self.loginstatus.text = ""

self.parent.current = 'services'

else:

print("Incorrect username or password!")

self.loginstatus.text = "Incorrect username or password!"

END loginscript(self)

BEGIN show\_it(self)

self.box=FloatLayout()

self.lab=(Label(text="Are you sure you want to send the email to the selected contacts?",font\_size=15,color=(1,1,1,1),

size\_hint=(None,None),pos\_hint={'x':.4,'y':.6}))

self.box.add\_widget(self.lab)

self.no=(Button(text="No",size\_hint=(None,None),color=(1,0,0,1) ,

width=200,height=50,pos\_hint={'x':0,'y':0}))

self.box.add\_widget(self.no)

self.yes = (Button(text="Yes",size\_hint=(None,None),color=(0,1,0,1) ,

width=200,height=50,pos\_hint={'x':.5,'y':0}))

self.box.add\_widget(self.yes)

self.main\_pop = Popup(title="Confirmation box",content=self.box,

size\_hint=(None,None),size=(500,300),auto\_dismiss=False,title\_size=15)

self.no.bind(on\_press=self.main\_pop.dismiss)

self.yes.bind(on\_press=ServicesScreen.sendemails)

self.yes.bind(on\_release=self.main\_pop.dismiss)

self.main\_pop.open()

END show\_it(self)

BEGIN sendemails(self)

server = smtplib.SMTP("smtp.gmail.com", 587)

server.starttls()

server.login("bphslatealerts@gmail.com", "Latealerts11")

msg = "Hello sir, we have realised that you have been away on Monday the 4/2/19. If you could please notify Mr Karko about your abscence and discuss actions that would be great. Thank you very much"

subject = "Away Date"

body = "Subject: {}\n\n{}".format(subject,msg)

for email in ServicesScreen.getEM(self):

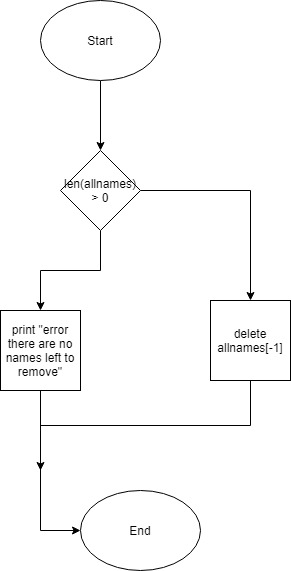
server.sendmail("test@gmail.com",email, body)

server.quit()

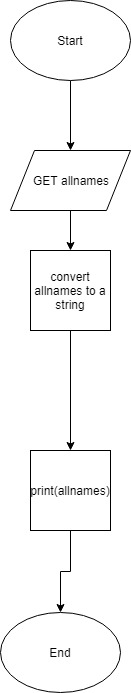
END sendemails(self)

### Flowcharts

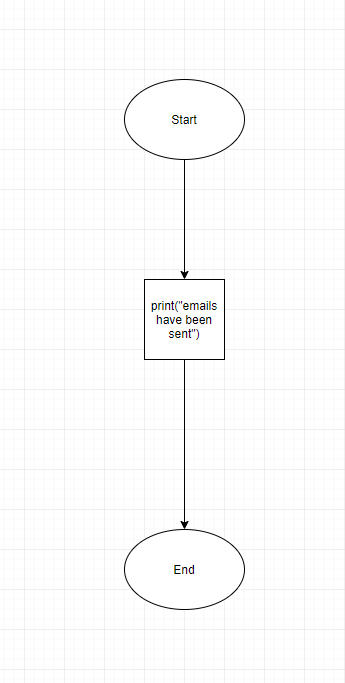
#### Def(deletename):



#### Def(printnames):



#### Def(popyes):



## Standard libraries of code used

During the development of this project, since I was using python it doesn’t have many built in libraries. However, you can install libraries that are verified by python through python that don’t originally come with the install. These libraries of code include open, reading and comparing excel files, connecting to the internet to send emails via the Gmail servers and creating the GUI (graphical user interface). These libraries have been very helpful as my project is task specific, however these libraries are open source and available to all python users.

I created 2 classes that utilise the libraries to open, read and compare the excel file to an array of data. These classes where generated from scratch to suit the solutions needs, this means that I couldn’t find anything similar online or from my previous work which increased my work load and decreased my programmer productivity as I had to build it from the ground up.

One class that I heavily used was the kivy class. The kivy library allows the programmer to generate a GUI for their project. This library has proven to be very helpful and intuitive. The classes that were created from the kivy library were very modular. It can be copied into any python code as long as they have the kivy library installed. The library of code allowed me to control aspects of projects GUI such as:

* + Colour
  + Window size
  + Window colour
  + Font
  + Button placement

It is a very abstract class design, utilising the kivy library. An upgrade thaty I can make to this class is being able to change the window colour and other aspects of the GUI during the runtime of the code.

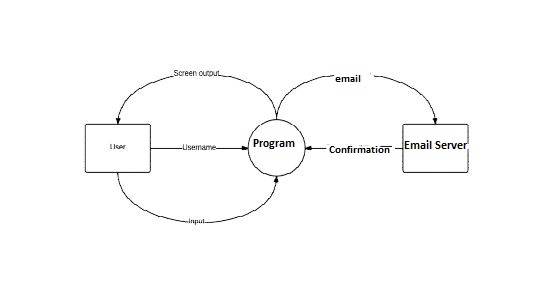
Finally I used the smtplib library to allows me to communicate with the internet and send data to the GMAIL server, then to be forwarded to the chosen recipients. These libraries proved to be very useful in the development of my project. I used these libraries to allow my solution to function they way it was discussed with the clients. These libraries and routines of code are very helpful in the implementation of my project, as without them, I would have to program them myself, and that would severely decrease programmer productivity.

## Test Data

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data set | Category | Sample | Reason for inclusion | Expected output | My output |
| 1 | Wrong print statement | All correct | Code kept printing out wrong statement | All correct | Nothing |
| 2 | No student names where entered | Empty | Code errored | [‘matt’, ‘mark’] | Code error |
| 3 | Sending emails | Empty | Not all the emails were being sent, | All emails have been sent | All emails have been sent, but upon further investigation not all emails were sent |
| 4 | Opening about menu | Empty | The button wasn’t working properly at some stages in development | The about menu to be open | The about menu wouldn’t open |

## Documentation of the overall software solution

### Context Diagram



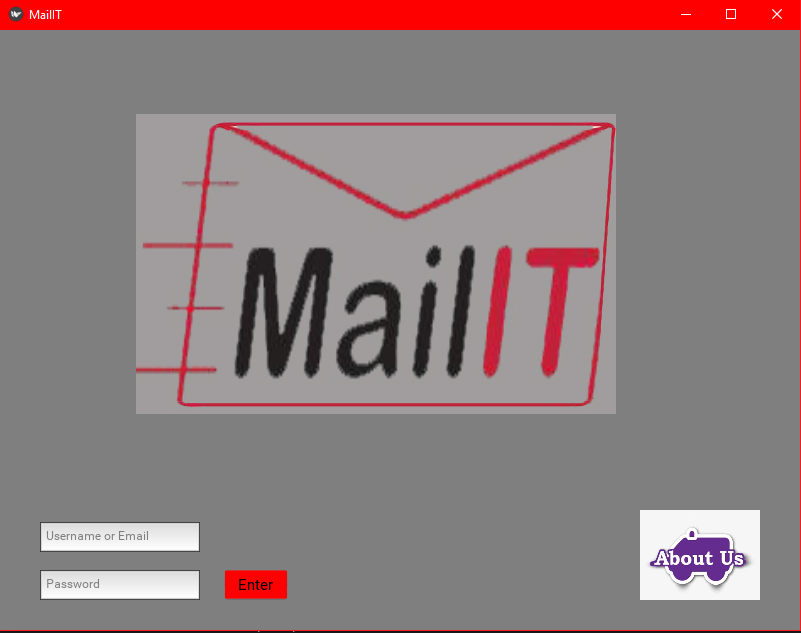
### IPO Diagram

|  |  |  |
| --- | --- | --- |
| Input | Process | Output |
| Enter | * Move to next TextInput * Activate button functions | * Add a error message if the input is invalid * Display the TextInput |
| Buttons are pressed | * Assigned functions for the button * Have multiple functions | * Print out what function is being executed |
| Email Sender | * Sends email to the GMAIL Server * Forwarded to recipients | * Dialogue box saying emails has been sent |
| About Page | * Opens up a new page * Calls image | * Image is displayed on the page |
|  |  |  |

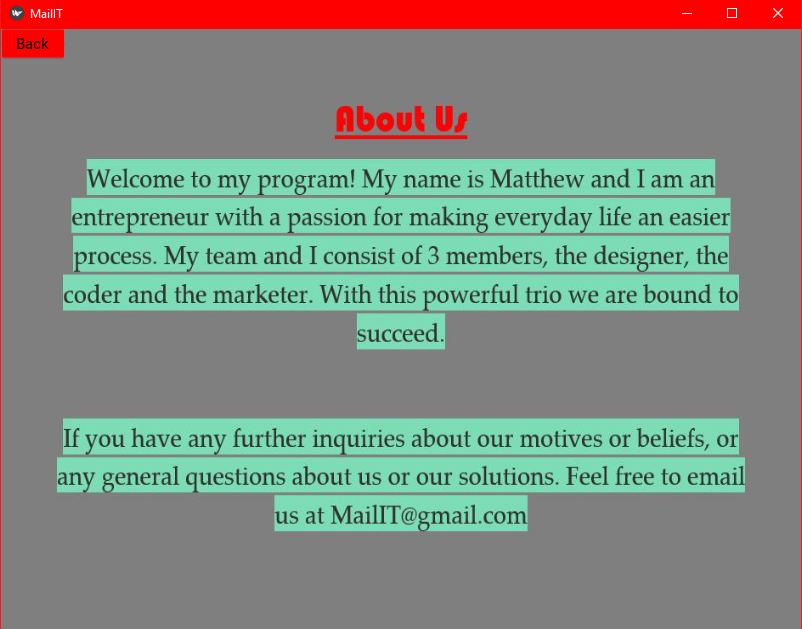
### Data Dictionary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identifiers | Data type | Scope | Description | Example |
| Allnames | Array | Public | Array of student names | Ahmed |
| Em | Array | Private | Array of student names | Michael |
| Len(allnames) | Integer | private | Length of an array | 23 |
| self.emlist1.text | String | Private | A string that is printed | All correct |
| self.emlist.text | String | Private | A string that is repeated | Hello how are you |
| self.box=FloatLayout() | Float | Private | The size of the screen | 400x400 |
| namez | Array | Private | List of inputted names | Jeff |
| e | File | Private | Opening the file to read it, reading it | “Database.xlsx” |
| Self.namelist.text | String | Private | Printing a string | “Hello world” |
| Df | Array | Private | Array of database names and emails | Smith, [s@gmail.com](mailto:s@gmail.com) |
| names\_text\_input | String | Public | Inputted names | Mark |
| Self.printnames | String | Public | Printing names on screen | Matthew  Mark |
| Nl | Array | Public | Array of file names global | Rufus  Earnie |
| Self.no | Button | Private | No button on popup information | n/a |
| Self.yes | Button | Private | YES button popup information | n/a |
|  |  |  |  |  |

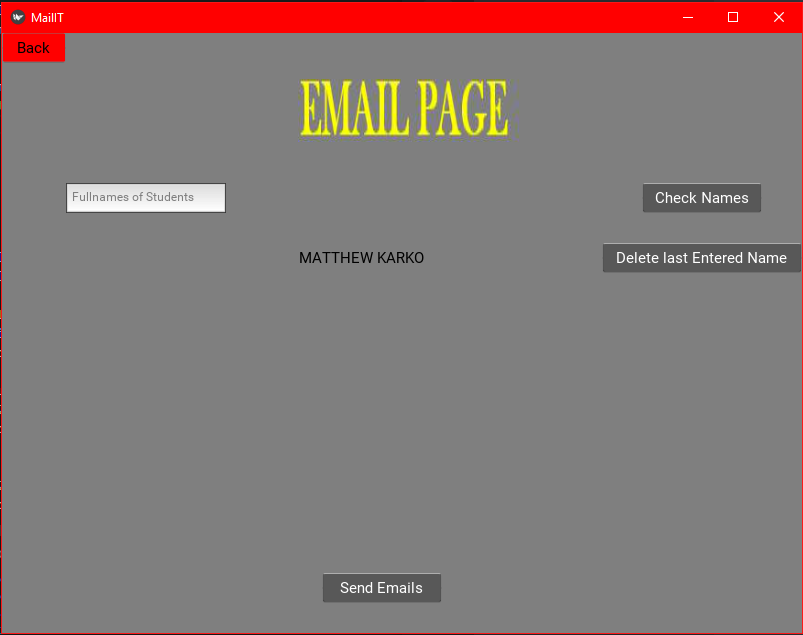
### Storyboarding



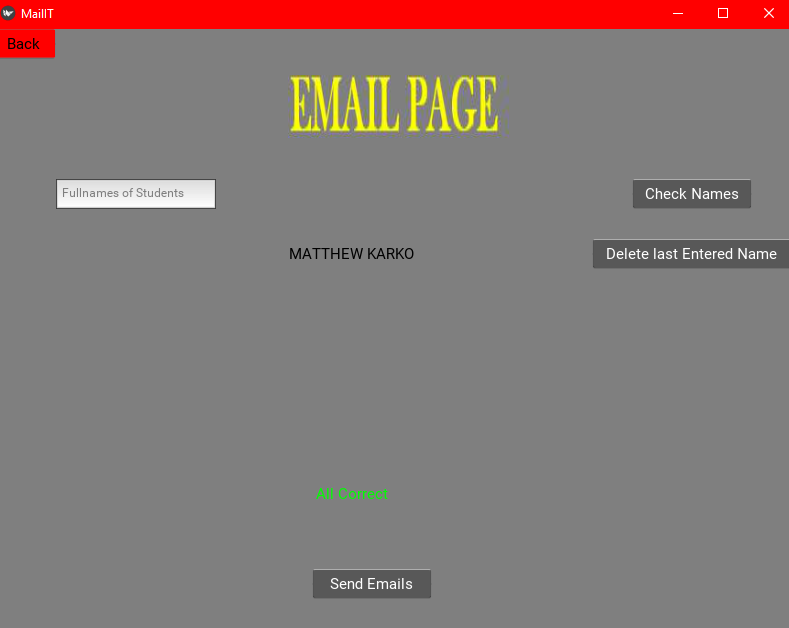
This is the login/introduction screen of the application. There is 2 Text inputs, 2 buttons and the logo of the program. The ‘About Us’ icon leads to a page explaining the people behind the project, their intentions and some characteristics about then. After you enter the correct details and press the ‘Enter’ button it will lead you to the email page.



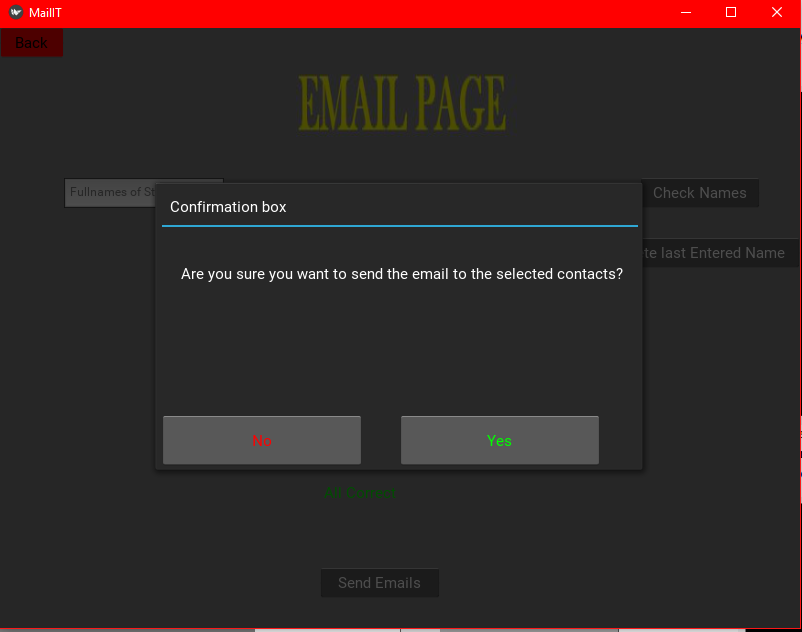
This is the about us page. There is a brief description of what the team is all about and how they came up with the idea for the program. Users are also able to press the back button that leads to the login screen.



This is the email page. It has one Text Input for the names of the students. Then it has 3 buttons all doing different funcitons. The Check Names button gives the user the option to double check his entries into the program. It also has a TextDisplay, being the name list that is entered.

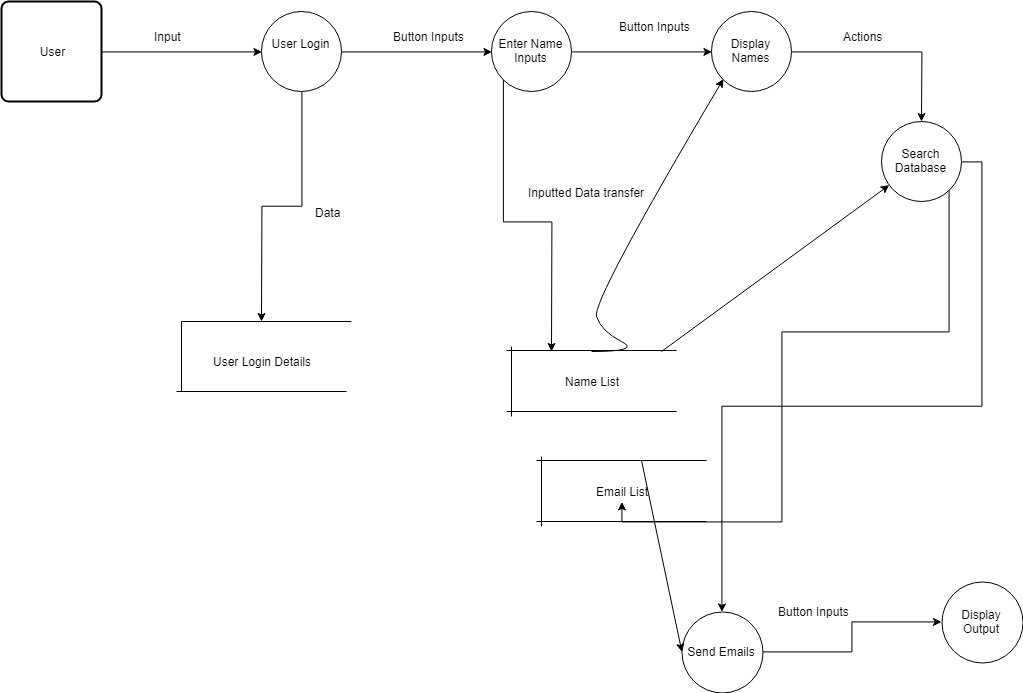
After the Check Names button is pressed another TextDisplay appears as such: 

After the Delete last entered name button Is pressed, the last entered name is removed from the name list. However when the Send Emails button is pressed a pop up shows ensuring with the user that they are 100% sure about sending the emails to the entered recipients.



The colour coding is done to connect the decisions on a psychological level.

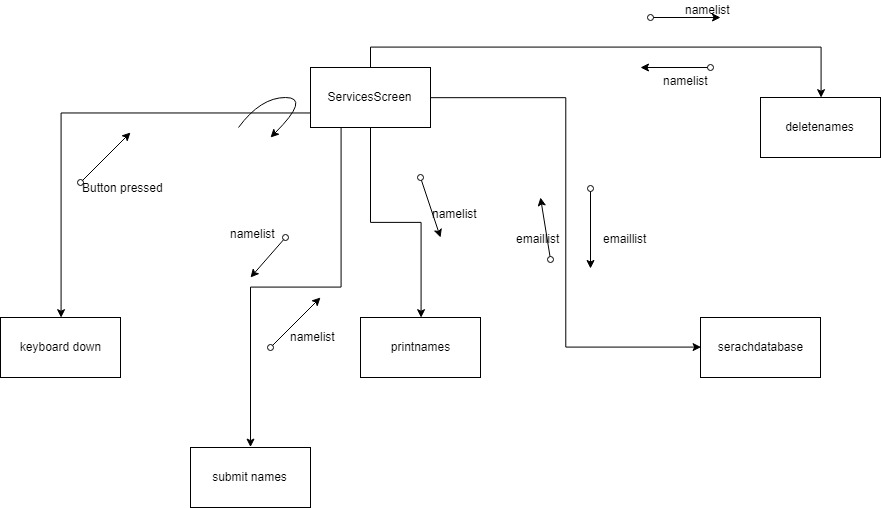
### Dataflow Diagram



### Structure Diagrams

#### LoginScreen

#### ServicesScreen



## Interface Design in Software solutions

### Issues that affect interface design

When designing the GUI, I had a few thoughts and speculations regarding the actual design. Here are some questions I had:

* What is my target audience?
* Is it easy enough to navigate through?
* How big do I make the screen?
* What colours do I make the buttons?
* Are the buttons big enough?
* What about people with disabilities?

This software solution is aimed at people who work in the office. After some collecting some primary data the youngest worker was 24 years old, so I am targeting an older audience. As a result of this, I decided to make the GUI less congested and have it very simple. This means that the menus aren’t detailed, and the buttons are clearly labelled and cannot be confused with other buttons. This simple GUI will allow for the users to very easily navigate and complete the task that they are set out to do. This also increases productivity within the workplace.

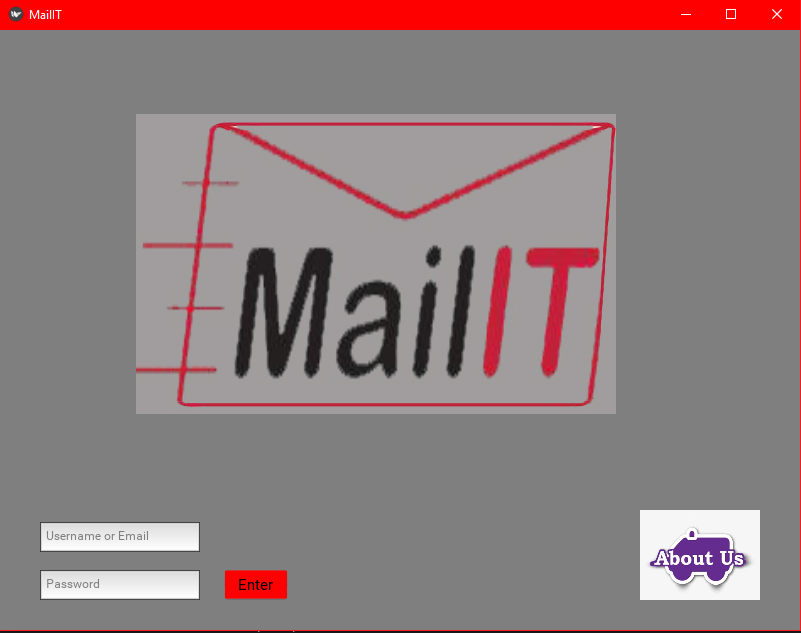
When deciding for the overall screen size of the application, I decided it was best to keep it windowed at a (400x400) window. This was done due to ease of access and being able to use the background apps alongside my product. This was also done to minimise the usage of the computer’s resources, leaving resources for other background processes. This also means that the minimum power requirements will be low allowing it to run on virtually any windows machine.

Currently user interface design is taking a minimalistic approach, with sleek buttons being implemented instead over complicated settings. I took inspiration from the interface design of the apple iPhone. They keep it very minimalistic with buttons and don’t overflow the user with unnecessary information allowing the user to feel comfortable and be able to understand what he is executing. This is the same figure that I wanted my GUI to be based on.

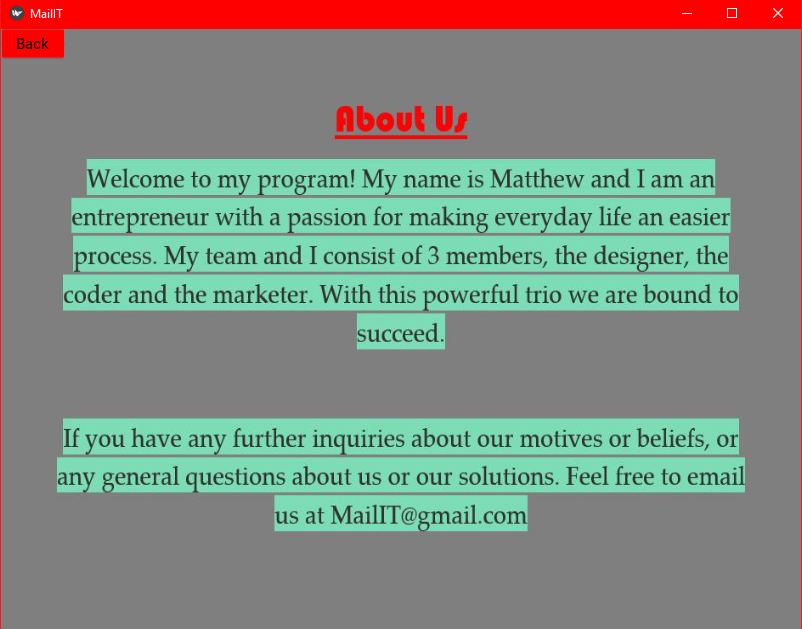
A challenge that I have found via research and primary data is maintaining consistency. Having a consistent user interface, users are able to know where to go to be able to complete the actions that they would like to complete. I have been able to maintain consistency through my GUI through experimenting with different designs and choosing the most consistence version that I had built.

This solution is designed for use of keyboard and mouse, it was not designed to be touchscreen compatible as when I was discussing with my clients about the sort of machines that this code will run on they, replied non-touchscreen. However, if they do decide to change to touchscreen, this code will be fully functional although it might require minor GUI changes to make it sleeker.

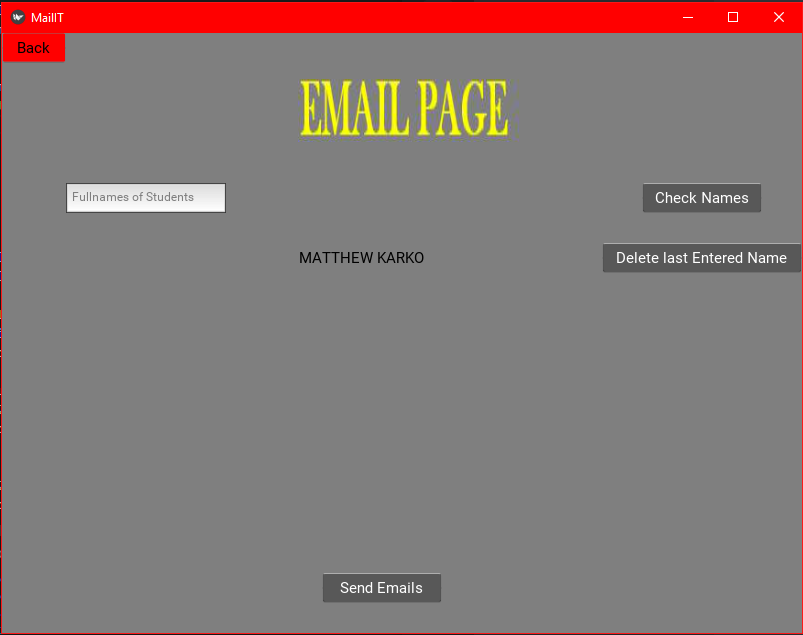
### Interface storyboard consideration



The MailIT Logo was placed in the centre of the screen to ensure that the user understands which program they are using. It is a different contrast of grey, so it stands out from the background making it more visible for the users. The ‘ABOUT US’ button was designed to be a different colour, so it clearly indicates where it is located on the screen, and to indicate that it is a button. It was placed in the bottom right hand corner as it is aesthetically pleasing.

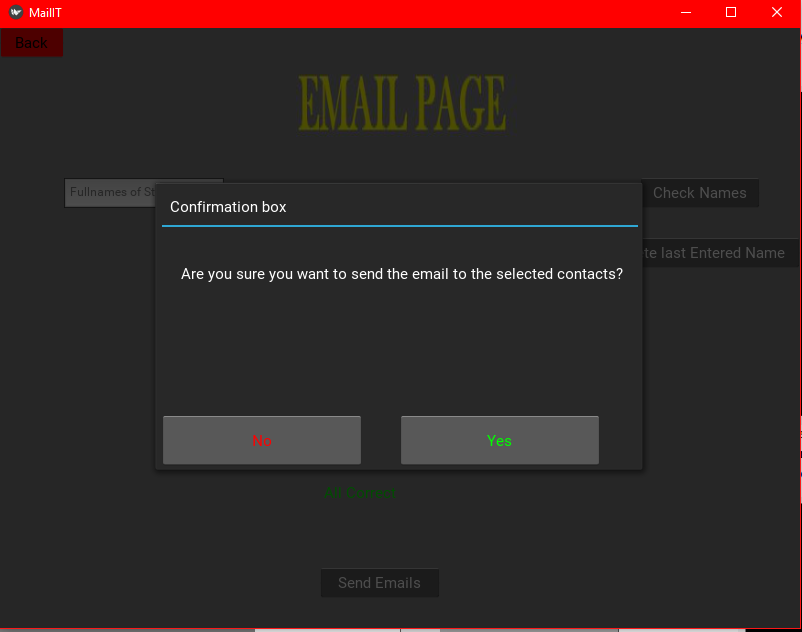


The ‘ABOUT US’ page information is in a bright green colour, to stand out from the window background making it easier for the user to read the text that is given. The back button allows the user to go back a page, it is red as red is an intense colour that suggests reversal, meaning the human’s brain psychologically relates the colour red to no, or back.



The buttons on the ‘EMAIL PAGE’ are grouped by their functions. This is so the user finds it easier to navigate and after consistent use. The user is able to immediately locate where he/she needs to go to do certain functions. This was the idea when designing the ‘EMAIL PAGE’. The send emails button is located on the bottom, and dead centre to show its importance. The check names and delete last entered names are placed together as their functions are a type that alter the name list. The email page title is placed top centre to clearly indicate where the user is in the program.

The name list prints in the centre of the page for the user to clearly identify what names he/she has entered and how many names they have entered. AS shown in the example above.



After pressing the Send emails button. A popup appears, the pop up is placed in the centre of the screen to grab the users attention. It dims the exterior of the popup to focus the users attention purely on the popup and none of its surroundings. Doing this it enables the user to clearly understand what the program is asking from him/her

## Factors to be considered when selecting the programming language to be used

### Factors to be considered

When deciding what programming language to use in the development of this project, there were a few factors that affected my decision. These factors include:

* Developer fluency in the language
* Language support (libraries, online resources)
* Performance of the language (runtimes, compile times, min requirements)
* Did it allow OOP (Object-oriented programming)

In my experience coding, I have only worked with HTML, and python. I would consider my self mediocre in HTML and fluent in python. I did not want to step out of my limits and do Java or C++ as I didn’t feel comfortable learning a entirely new language in the timeframe that I had. Between HTML and python, HTML didn’t have any of the features or functions that I needed for this specific project as it is a web-based language. So python was my next option. Python is very abstract and includes all the features and functions that I needed for this project. Also the support for this platform (python) is widespread as there are many developers out there on this platform.

The principles of OOP were very important in the development of this project. Python incorporates OOP as one of its language features, heavily supporting it.

The targeted platform for this program is a machine that has Windows OS running on it. Almost all machines have the minimum power requirements, meaning that the can all run python which was a positive and beneficial reason as to why I chose python. However, I will need to make sure that the Installation of python is done properly as there is external libraries that need to be installed separately in order for the program to run. In the event that this program was commercial, there would be a higher demand for pre-built binaries and pre-compiled software.

Support available for the chosen language is very important when developing. Python has very comprehensive guides, tutorials and documentation on the internet provided by python themselves and there is a lot of 3rd party assistance with python which deemed to be a very huge factor in the decision of the language. Due to this, it has allowed me to understand the language more and become more fluent in python, this also ables me to get better. High quality help by asking peers

## Factors to be considered when selecting the technology to be used

### Performance requirements

When developing the program, performance specifications must be put in place to ensure the program runs efficiently and effectively. These are the factors that will ensure optimum performance:

* The app must launch within 5 secs
* The connection must be made between the code and the GMAIL within 3secs
* Program shouldn’t stop responding
* No lag within the program

These requirements are enough to demonstrate if the app will run smoothly or crash.

### Benchmarking

To ensure that the program will run smoothly on a variety of PC’s, my project has been tested on multiple machines. Although the program doesn’t have an inbuilt benchmarking system, I have conducted and ran this program on several Windows machines and it has ran with no issues or stutters.

Benchmarking has allowed me to analyse the performance of my program and

# Stage 3: Implementation of Software Solutions

## Techniques in developing well written code

### Good programming practice

There are many ways a programmer can write high quality code. This is done by using a number of techniques and tools that are provided. Following good programming practice means that, errors will be reduced, code is easier to read and etc. There are a list of practices that should be adhered to when programming, this list may include:

* Commenting your code to understand what it is doing
* Version control (github)
* Writing to enable maintenance
* Separating logical tasks into subroutines
* Use of a clear modular structure
* Using control structures

Most projects, when developed use a top-down approach. Doing this allows the programmer to detect errors more efficiently and the whole SDC (Software development cycles) becomes more productive. For my project, I have used the top-down approach as it allows me to stay informed and understand what the code is doing at what time. It is very clear and very easy to understand, which was another reason as to why I chose the top-down approach. Updating the code has also become a lot easier, if it requires an update it will be very efficient.

When programming, it should be done in a way that allows for future updates to be very efficient and effective. Tools such as Documentation, Gantt charts and etc Allow for future developments and changes to be documented, also making the process of maintenance very productive.

Version control is a vital tool when it comes to working on any form of a programming project. The process of version control allows for backups of the code to be stored, which means in the case of an issue or mishap with the new update, you can easily fall back on the previous version until the bug is fixed. A technology that assists with version control is github.

### Errors faced

Errors occur when the program isn’t written correctly or doesn’t function as it was designed out to. There are 3 main types of errors in code:

* Syntax
* Logic
* Runtime

#### Error 1

This if statement is a logic error. A logic error is when the code’s syntax is correct; however, the code doesn’t perform the correct job.

if len(em) = len(allnames):

self.emlist1.text = "All Correct"

This if statement uses the “=” operator, instead of the “==” operator for checking if two integers are equal. To solve this error, I corrected the if statement, as shown below:

if len(em) == len(allnames):

self.emlist1.text = "All Correct"

#### Error 2

This statement is a form of a syntax error. Syntax errors occur when the code doesn’t make sense to the machine. This was the most common issue when I was developing for the project.

self.yes = (Button(text="Yes",size\_hint=(None,None),color=(0,1,0,1) ,

width=200,height=50,pos\_hint={'x':.5,'y':0})))

When I was programming the popup screen, I copy and pasted this line with minor variation several times. However, whilst copying the statement that I wrote I added an extra bracket which ended up causing 5 syntax errors. I solved it very simply as shown below:

self.yes = (Button(text="Yes",size\_hint=(None,None),color=(0,1,0,1) ,

width=200,height=50,pos\_hint={'x':.5,'y':0}))

#### Error 3

This if statement was another syntax error that I had run across during my programming. This error was caused due to me being unclear about how the len() class worked from python’s libraries. I had a very unclear understanding about the way the class worked.

if len[allnames] > 0:

del allnames[-1]

print(allnames)

Instead of the use of parentheses “( )” I used the square brackets “[ ]” which meant that it became a syntax error as that isn’t how the class works. It was easily solved as shown below:

if len(allnames) > 0:

del allnames[-1]

print(allnames)

#### Error 4

Another logic error that I encountered was when needing to search a file.

df = pd.dataframe(e, columns = ['Names', 'Emails'])

search = df[df['Names'].isin(nl)]

Dataframe was the external class that I was using, however I forgot to capitalise the DataFrame and it didn’t do what it was expected to do. However, the code still. It was a simple fix.

df = pd.DataFrame(e, columns = ['Names', 'Emails'])

search = df[df['Names'].isin(nl)]

#### Error 5

This was another syntax error. Syntax errors as shown were the most common form of erroring when I was programming my project.

def show\_it(self);

self.box=FloatLayout()

This error above is starting the definition, I used the “;” instead of the “:” which caused a syntax error. This was done by my mistake not pressing the shift key. It was a very simple fix,

def show\_it(self):

self.box=FloatLayout()

### Methods to detect errors

There are several ways to detect errors in code. The techniques that I used to detect errors in my project were:

* Debugging output statements
* Peer checking
* Desk checking

When debugging errors, i used the simple print statement that python has built in to create a debugging class that benefitted me in finding and solving errors in my code. There were many times where I was facing logic issues and I used this class to find and solve the issue. Overall it proved to be very effective.

Desk checking my program was able to provide me with valuable data about how my classes work. By desk checking my classes I was able to find errors within my loops and button classes. If I didn’t desk check my code, it would’ve remained the same and been spotted later in the SDC.

Peer checking my code allowed me to gain another perspective which is always beneficial. My peers were able to notice errors and things about my code that I was unable to figure out. This helped a lot in developing my code as I had some major flaws that I hadn’t noticed that they pointed out.

### Software debugging tools

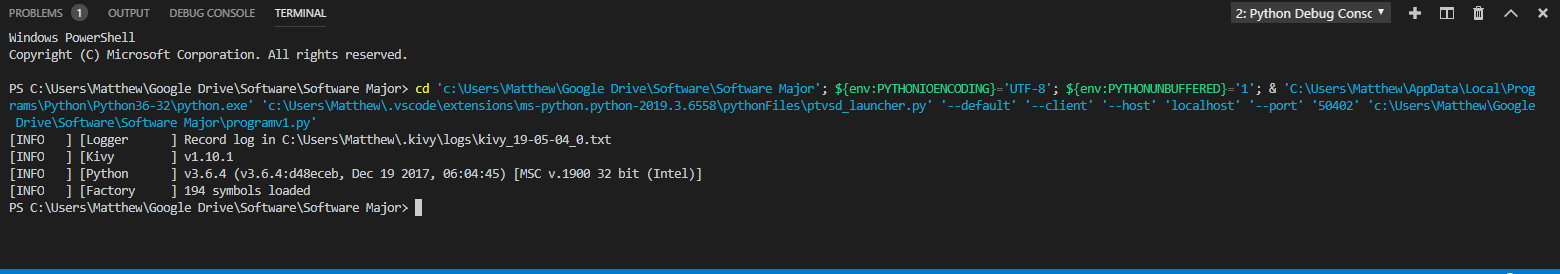
In the development of my project, I used debugging tools to solve issues. The 2 debugging tools that I used in python are:

* Visual Studio Code Problems
* Visual Studio Code Terminal

The VSC problems tabs helped me a lot in the development of my code. It pointed out all the issues that my code had. It would display the type of error and on what line it happened. It proved to be very useful when developing my project.



The VSC Terminal tab prints and displays everything that your program is doing at a given time. It was very helpful, as it showed what my code was doing and with the company of my error fixing methods it helped me solve a lot of issues that I had with prior versions of my code.



## Documentation of the overall software solution

### User documentation

The user documentation/installation guide is located as a separate file in the same directory. It will also be available on my blog <https://mksddmajorproject.wordpress.com/>

### Technical documentation

#### Log book

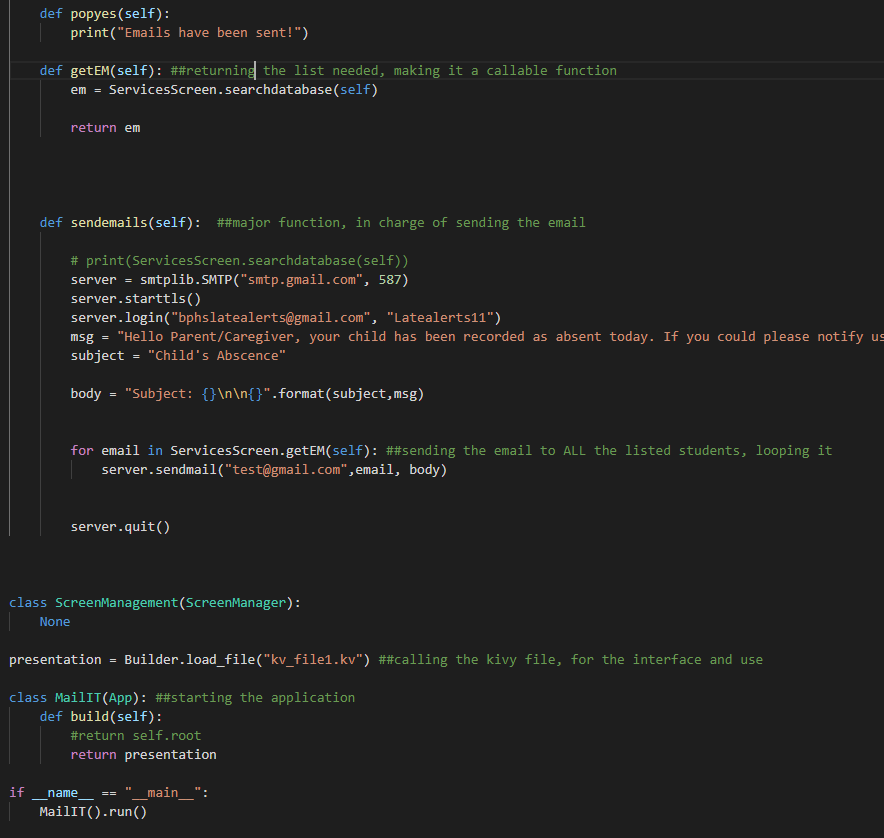
The log book can be found on <https://mksddmajorproject.wordpress.com/>. The logbook includes information on when certain parts of the project were completed, and what issues I encountered while programming.

#### System documentation and algorithms

System documentation, including descriptions on how the game works, can be viewed in this document. Some algorithms can also be viewed in this document. Flowcharts, DFD, context diagrams, class diagrams and storyboards are included in this document. As shown in the pla

#### In code comments

Within the code, there are comments explaining what each line does and how each bit works. These comments improve readability and increase ease of maintenance.



## Hardware requirements

Minimum hardware requirements are put in place so that the users can see whether their computers will be able to run the program or not. These specifications are as follows:

* Operating System: Windows running machine with Visual Studio Code (Python 3)
* CPU: Intel Pentium or greater
* RAM: 512mb minimum
* Storage: 1TB
* GPU: Intel HD integrated graphics 2000 or greater
* Screensize: 400x400 or greater
* A connection to the interwebs

## Testing the software solution

### Evaluate success of the project

The success of a project depends on how closely it fulfils criteria and specifications that it was developed for. The specifications for this project are as follows:

* Be able to read excel files
* Have a GUI
* Have a send button
* Be able to send emails to several people
* Be able to read and search files

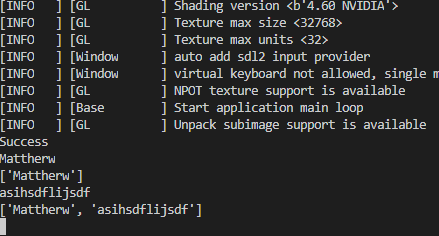
In my opinion my program exceeded the criterion that I had set out to complete. I need to check with the client to ensure that all of these criteria are fulfilled, and to get their signoff that the project was successful.

### Test data use

In the evaluation and testing of this project, I needed to ensure that It would always meet the expectations of the client. I would run through and check the code regularly and maintain my documentation as I proceeded with my project. To test my code I was able to get some of my peers to run through the code and also my parents who aren’t usual tech users.

When running through the code there were no new or any errors.





## Post implementation review

### Discussion with client

After trying to negotiate and organise a board meeting with NESA representatives (approx. 100000 people) I was unable to reach them. For future projects, I will use better methods of incorporating the client into the SDC of the project. However, I was able to discuss with my dad’s cousin who is an idol to me. He gave some pointers on the state of the game whilst he was wearing his dish dasha.

My dad’s cousin said that the program had a great idea, and it suited the criteria perfectly. They were satisfied with the work completed and he signed off.

## Modifying code to meet changed requirements

### Changes that may need to take place

In the future, certain updates to this project may need to take place. Some of the changes include:

* Bug fixes
* Additional functionally (new features)
* Better GUI Design

Bugs are common throughout all programs, and as a result, a method pf patching bugs is required. Usually you have a team of programmers that constantly update the program. The program is able to work better with the bugs removed

Most programs include the functionality to add new content. However, my code has been made so the updating process will require a lot of down time from the clients. Some features that I would like to add is to store the database on the cloud, as well as being able to instead of having office messenger to instead notify the student/ teacher directly.

Even though this game is designed for quick and easy access, I would like to make the GUI appear a lot better, as I believe it is too simplistic for people of our day and age. This would drastically improve the likeability of the program, making it suitable for young audiences as well as old.

‘

## User Documentation Modification

Every time I updated the project, there were certain aspects that needed to be changed. These include:

* Updating psueocode
* Updating classes
* Updating data flow diagrams
* Updating code comments
* Updating errors faced
* Updating data dictionary
* Updating storyboards

It is done so the documentation can be kept up to date and be valid for the client. Also, it is made easier for future developers to understand the problem and work on the solution, also making updates to the program a lot more straight forward.