



Object Oriented Programming

[CT 451]

Bachelor's Degree in Electronics, Communication
and Information Engineering

(BEI/076)

Groups: A and B

Department of Electronics & Computer Engineering

Pulchowk Campus

Institute of Engineering

Tribhuvan University

TABLE OF CONTENT

Front page -----	3
Abstract -----	4
Acknowledgement -----	5
List of Figures -----	5
Abbreviations -----	5
Chapter One: INTRODUCTION -----	6
1.1. Background and problem statements -----	6
1.2. Objectives -----	6
1.3. Limitations -----	7
1.4. Formatting guidelines -----	7
Chapter Two: PROBLEM ANALYSIS -----	7
2.1. Understanding the problem -----	7
2.2. Input Requirements -----	7
2.3. Output Requirements -----	7
2.4. Processing Requirements -----	8
2.5. Technical Feasibility -----	8
Chapter Three: Review of Related Literatures -----	8
Chapter Four: ALGORITHM DEVELOPMENT AND FLOWCHART -----	9
3.1. Algorithm -----	9
3.2. Flowchart -----	10
3.3. UML -----	11
3.3.1 Structural UML Diagrams -----	12
Chapter Four: IMPLEMENTATION AND CODING -----	13
4.1. Implementation -----	13
Figures -----	14
4.2. Coding of the project -----	17
Chapter Five: RESULTS AND DISCUSSION -----	18
Chapter Six: CONCLUSIONS -----	19
REFERENCES -----	20
APPENDIX A (Source Code) -----	21

**TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS**

Mass Mailing Using CPP

**A COURSE PROJECT SUBMITTED TO THE DEPARTMENT OF ELECTRONICS AND
COMPUTER ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE PRACTICAL COURSE ON
OBJECT ORIENTED PROGRAMMING [CT 451]**

Submitted by:

Jeevan Koiri (PUL076BEI016)

Bhupendra Chaulagain(PUL076BEI011)

Vikrant Panjiyar (PUL076BEI048)

Submitted to:

Department of Electronics and Computer Engineering, Pulchowk Campus

Institute of Engineering, Tribhuvan University

Lalitpur, Nepal

December - 16, 2020

Abstract

This project removed the necessity of payments for email marketing. In simple words, mass mailing became possible using a simple CPP console based application and totally free to use and send bulk emails with so ease. The main purpose of this research was to prevent the requirements of AWS or other expensive web applications (Cloud computing services) and make mass mailing easy, free and automated using CPP. We found that using the same SMTP components which are used by web applications, we can make our own web application to send mails. Using the simple coding idea and looping techniques we came up with a great project which is able to send different types of beautiful mails i.e. both html mails with embed images and simple text email so easily and to thousands of recipients at once. We also added features to attach files with the email body, typing html directly in console which will automatically be converted into required form. Our result showed that it is now so reliable and flexible that the user no longer has to know or understand the backend code and simply select features he/she want to use to enjoy the mass mailing for free. We came to the conclusion that we made a project just as same as AWS mass mailing system with even more features to send bulk emails like using our project the recipients won't know that the same mail is sent to others as well. This feature is not provided by any mass mailing web services as even by using those services, recipients still can see the total recipients list.

Keywords: object oriented programming, console, AWS, Mass mailing, web application, SMTP, CPP, Cloud computing, looping, bulk emails, embed image, backend code

ACKNOWLEDGEMENTS

We would like to express a deep sense of thanks & gratitude to our Professor, Project guide Mr. Bikal Adhikari for his patient guidance, enthusiastic encouragement and useful critiques of this project. The materials and the lectures he provided to us are really awesome that helped a lot while doing the project. The guidelines he gives us and the manner he teaches us are the source of motivation for us. The online materials that we get from the stackoverflow really help us for solving the problems that arise during the project.

We are highly indebted to teachers, both from the past and the present, for their valuable suggestions and their painstaking efforts to make this subject easier for students.

We must thank our classmates for their timely help & support for the compilation of this project.

Last but not the least, we would like to thank the emailarchitect and its whole team for providing us such an awesome service at free of cost which allows us to send mails to anyone.

LIST OF FIGURES

1. Inheritance Diagram
2. Variables used in our Projects that are inherited in Mails class.
3. Functions that can be called using the object of Mails class.
4. Screenshots of Execution of program
5. Varieties types of mail sent using our software
6. Flowcharts

ABBREVIATIONS

AWS - Amazon Web service

CPP/ C++ - C Plus Plus

CRM - Customer Relationship Management

G Suite - Google Suite

SMTP - Simple Mail Transfer Protocol

IEEE -Institute of Electrical and Electronics Engineers

OOP - Object Oriented Programming

MS – Microsoft

HTTPS- Hypertext Transfer Protocol Secure

Chapter One: INTRODUCTION

This project arose when one of the members of this team faced a real life problem and wanted to solve using CPP code. This Project is based on mass mailing. This means sending bulk emails or simply sending mail to many recipients at once. It might be little different than mass mailing but better than that as well as when mass mailing is done using G Suite account or other paid CRM services, recipients know that the mail is sent to others as well i.e. namelist is visible to every recipients. This project gives a simple platform where we can just copy and paste the namelist of hundreds of thousands of recipients from google sheet and send mails by typing body content in the console and even html mail i.e. we can design very complex mail with embed images in google sheet and just download it as html and use that to send. We can attach any number of attachments as well with email body. No complexity is needed. Here it gives lots of features like sending mail one by one to every recipients with little changes in respective format or sending same email to all recipients by single click through console or even sending mails using readymade html format. All we have to do is to enter email address and password, enter name one by one in console or even read and write multiple addresses from text file at once, enter name of attachment, enter name of html file (If html mail is to be sent) or simply typing mail in console at the moment and press enter key. The main feature it provides is that the user has nothing to do with backend code. It is as simple as logging in using mail.google.com and all these facilities are for free. The greatest feature of this project that makes it perfect and different from mass mailing is that no recipient knows that the same mail is sent to other people as well as the namelist is not visible to any recipients.

1.1. Background and problem statements:-

The idea of mass mailing using CPP came when one of the team members was already working in an NGO as sponsorship manager searching for sponsors by mailing hundreds of companies everyday and his main task was to send mails daily which were 200 + a day. But the problem was mass mailing could make every recipient know that the same mail was sent to others as well and only paid web services could be used to do that like AWS or G Suite account. And the other problem was that in some of the cases every recipient needed to receive a simple change in mail content. He didn't like attaching files every time with every mail and writing contents with simple changes again and again. The simple solution was a gmail template which also had a problem that even the template worked with body but still attachments must be attached every time. So he prepared a code to automate the process using python which was simple but little slow. So the same implementation could be done with CPP.

1.2. Objectives:-

Looking into the problem we came up with a solution by writing a code to automate the process. This project is able to send mails to hundreds of thousands of recipients without making anyone know that the same mail is sent to others as well. The main objectives of this projects are:

- ☐ Send mails to any number of recipients with automation in attaching mail contents
- ☐ No recipient should be able to know if the same mail is sent to another as well
- ☐ No paid web services should be required

- ☐ Any type of mail format with embed image must be simply sent by just attaching html file with no decoding problems
- ☐ Two way input for recipients address either directly into the console or from .txt file
- ☐ Two way body content input either directly from console or directly attaching html file
- ☐ Enable login from any email as required to send mails
- ☐ No knowledge of backend code should be required for users of this services
- ☐ To send beautiful emails with attachments with so ease to hundreds of recipients at once
- ☐ Automation in sending mails and attachment process

1.3. Limitations:-

As always, code is to be upgraded as time passes. So there always exist some problem with code when working with web interface. Some of the limitations of this project are:

- ☐ This project is still in adding-more-features stage so visual effects are not added yet
- ☐ The attachments must be in same folder
- ☐ We cannot paste in the console the list of recipients. So, file is used where list to be copied
- ☐ Mail is sent using loop technique i.e. mail is sent to each recipient one by one once started automatically. So, for thousands of recipients it might happen that a last recipient will receive the mail an hour lately

Chapter Two: PROBLEM ANALYSIS

2.1. Understanding the problem

As mentioned earlier, mass mailing brought problems like it could make every recipient know that the same mail was sent to every individual and it could be done only through paid web services. So in order to minimize the problems caused by those services we decided to make these projects with the view of sending more than thousands of mails at a time to different people without letting them know that the same mail was sent to every individual.

2.2. Input Requirements

As the title 'Mass Mailing' suggests the inputs required for this program includes almost all the inputs we give while sending mail that includes the mail to whom we send email, subject, body and the attachments.

2.3. Output

We were able to send beautiful emails using CPP to hundred or even more recipients without making them know that the same mail was sent to others as well. The email body content could be either html with embed image or simply text based. We also added features of adding any number of attachments with

email. The output is based on errors or success i.e. if mail is sent the message will be displayed for each success sent and if any types of errors occur like wrong email address or password then the respective message will be displayed in the console.

2.4. Processing Requirements

Our project doesn't really require any expensive tools or softwares for its purpose. But indeed it requires a small executable (.exe file) which is located at the email architect website. It requires a server address to send the mail which is already included in the project code. SMTP server address with some port is its one requirement that should be fulfilled. Other basic processing requirements are email address of the sender, his password, subject he wants to enter, body of the text, attachments and Recipients List. These are taken during the input.

2.5. Technical Feasibility

The current project can be set as the good solution for the real life problem that can provide an assist in various fields including banking sector, hospitals, schools and many more. The project seems feasible and reliable in these areas due to its simplicity and its great use for sending the same mail to many people at once. It on the one hand saves time and on the other it does not require any special technical knowledge to send mail using this project as users of the project need not to know the backend code and simply play with its features just by selecting numbers.

Chapter Three: Review of Related Literatures

Important dates for the development of emails

The first intra-network mail was sent in 1971, since then email has been an integral part of our daily life.

1971-1989: Age of Plain Text Emails

1989- Early 2000's: HTML email take over and email becomes public

2001-2009 Mobile users increased and emails became responsive

2009- Present days CSS has developed and enhanced the quality of mail

Server Address

SMTP server: *SMTP is an acronym for Simple Mail Transfer Protocol. This is used for routing messages around the internet. Smtplib.com is used here as smtp server address And port address for sending mails are: 587, 465. Out of these two ports port 587 is used in our mail so the Server Address becomes smtp.gmail.com:587*

Chapter Four: ALGORITHM DEVELOPMENT, FLOWCHART and UML Diagrams

In this chapter students are expected to present the algorithm and flowchart developed to tackle the problem at hand following the problem analysis. This chapter will require some figures. These figures must be properly numbered and labelled. They must also be included in the List of Figures section.

In this chapter students are required to include the algorithm and flowchart of the project with sufficient description in following format:

3.1. Algorithm

Step 1: Start

Step 2: Make an object m of class Mails

Step 3: Show the front page to the user.

Step 4: Ask to press any key.

Step 5: Go to the next page which shows the headers and ask for the Email and Password of the sender.

Step 6: After entry go to the next page and show the header and session of the user and ask for the subject entry.

Step 7: Get the body and No. of attachment and type the names of each attachment.

Step 8: Get the Recipients of the mail using console I/O or using File I/O.

Step 9: Show the recipients to the user so that the user can edit them in file I/O mode.

Step 10: Show the variables to the user so that user can know to whom the mail is going to be sent.

Step 11: Take each variable containing the Recipient email and send same email to each mail.

Step 12: End

3.2. Flowchart:

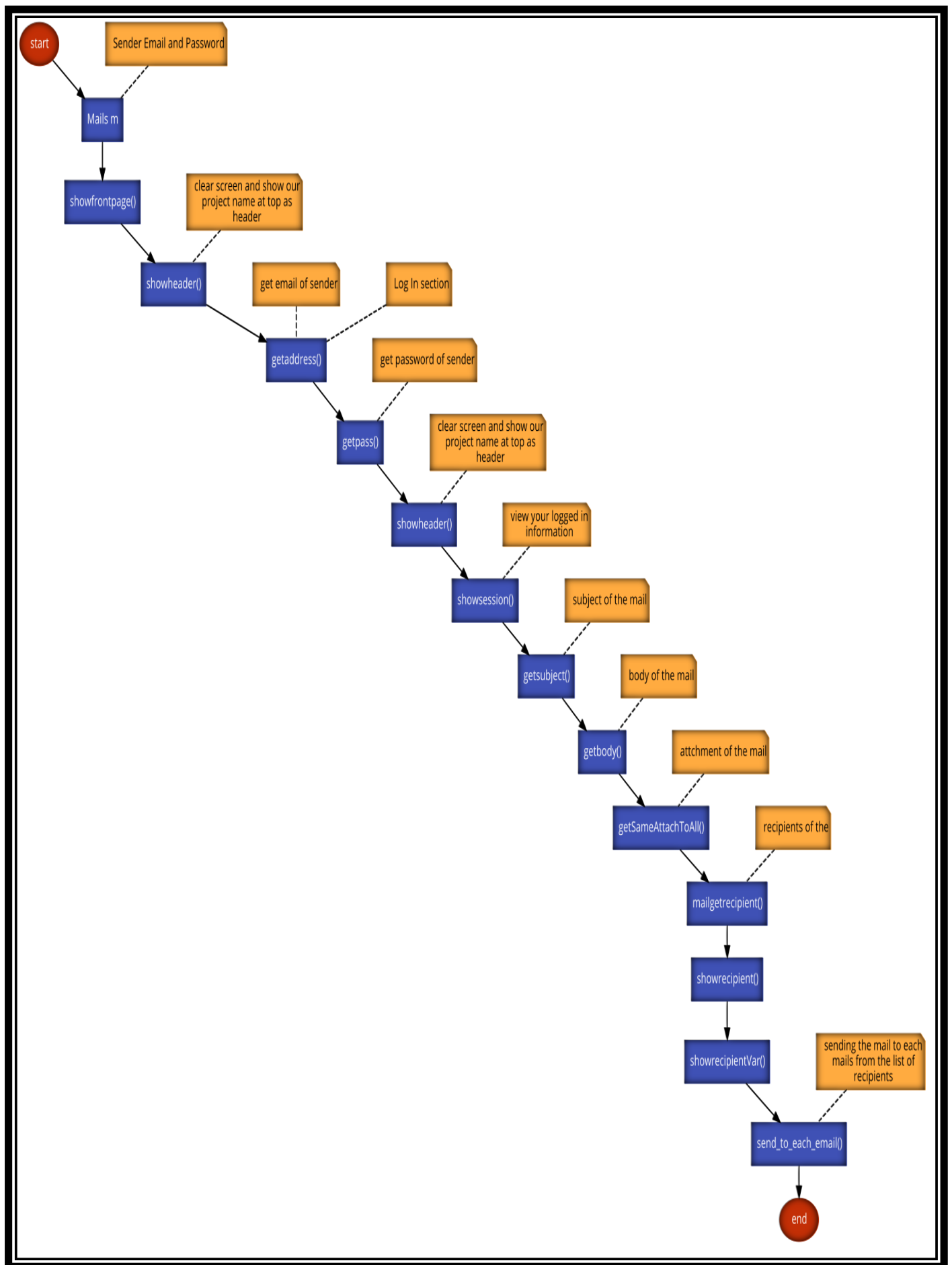


Figure a: Flowchart for the main Program

3.3. UML

The **Unified Modelling Language (UML)** is a general-purpose, developmental, modelling language in the field of software engineering that is intended to provide a standard way to visualize the design of a system.

3.3.1. Structural UML diagrams

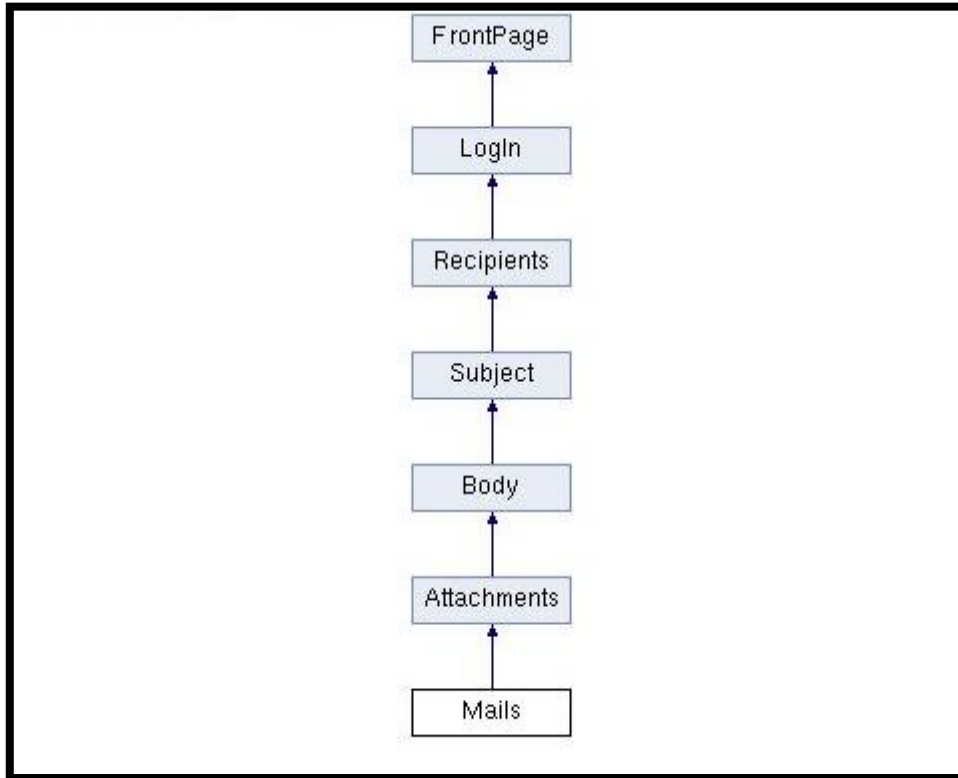


Fig: Multilevel Inheritance Diagram of Our project (Mail Sending Using C++)

Protected Attributes

int choice_body

▼ Protected Attributes inherited from Attachments

int no_of_attachments

char your_attachments [100][50]

▼ Protected Attributes inherited from Body

char your_body [100000]

char ch

int choice_body

▼ Protected Attributes inherited from Subject

char your_subject [100]

char subject [100]

char ch

▼ Protected Attributes inherited from Recipients

int no_of_recipients

int list_choice

int consoleOrfile

char recipients [1000][50]

▼ Protected Attributes inherited from Login

char your_address [100]

char your_password [50]

Fig: List of all Variables of our Project

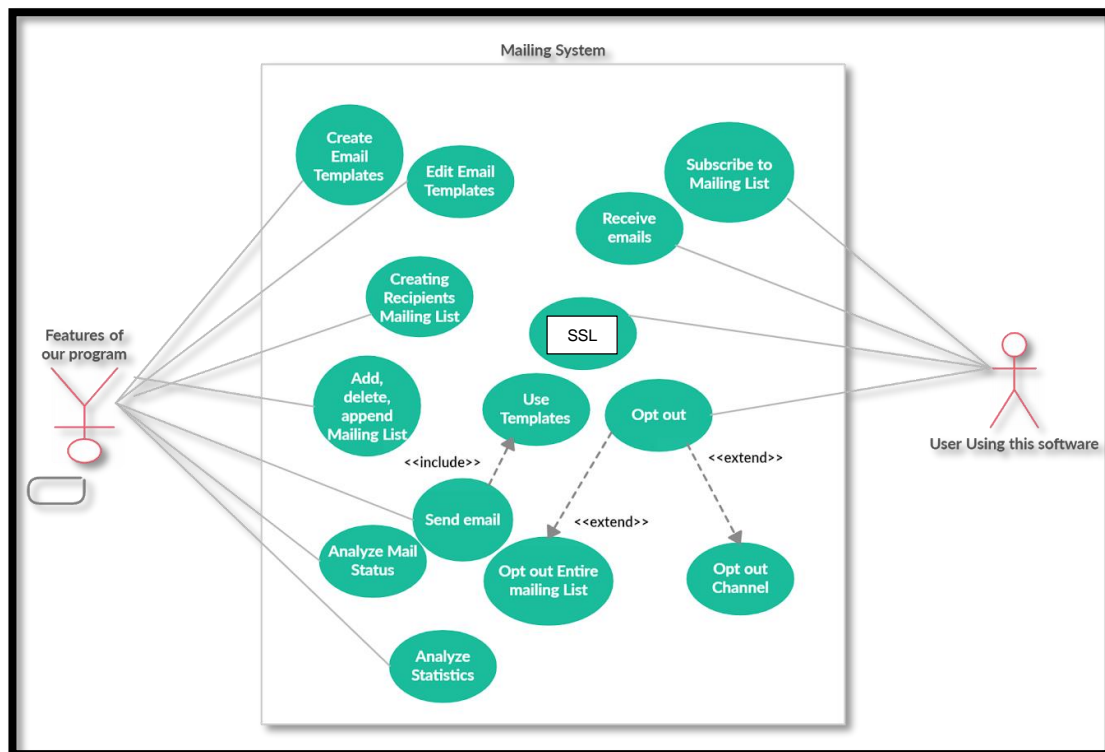
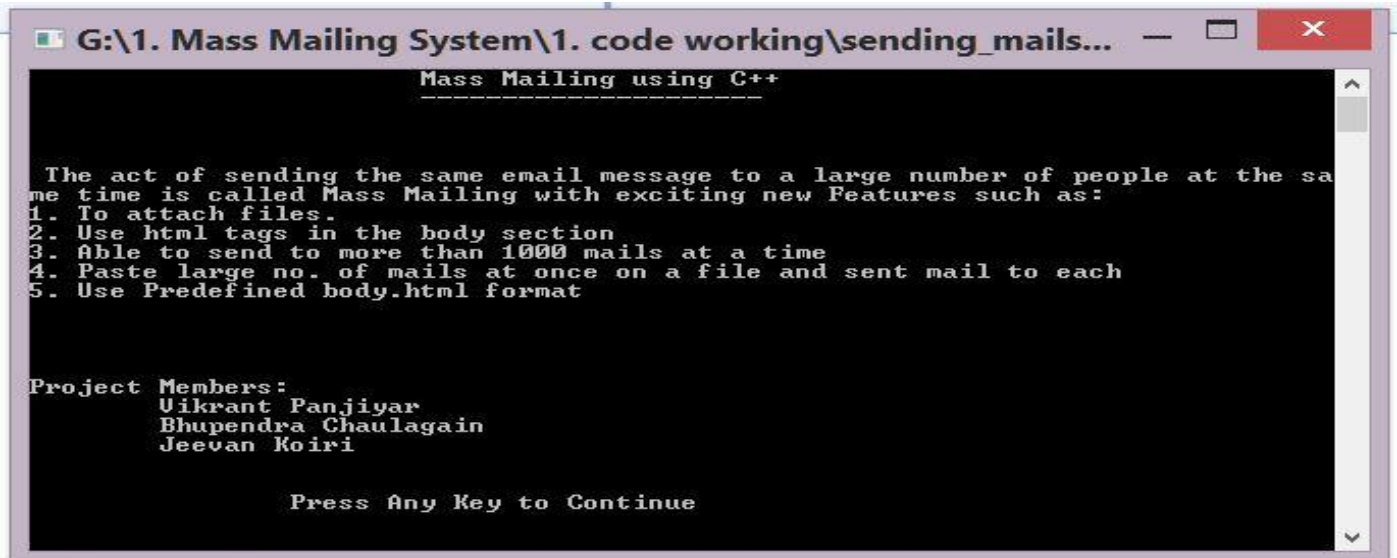


Fig: Mailing System Features

Chapter Four: IMPLEMENTATION AND CODING

4.1. Implementation:

The implementation of our software can be seen in the snapshot below:

A screenshot of a Windows command prompt window titled "G:\1. Mass Mailing System\1. code working\sending_mails...". The window displays the text "Mass Mailing using C++" followed by a list of features: "The act of sending the same email message to a large number of people at the same time is called Mass Mailing with exciting new Features such as:", "1. To attach files.", "2. Use html tags in the body section", "3. Able to send to more than 1000 mails at a time", "4. Paste large no. of mails at once on a file and sent mail to each", "5. Use Predefined body.html format". Below this, it lists "Project Members:" followed by "Uikrant Panjiyar", "Bhupendra Chaulagain", and "Jeevan Koiri". At the bottom, it says "Press Any Key to Continue".

```
G:\1. Mass Mailing System\1. code working\sending_mails...
Mass Mailing using C++

The act of sending the same email message to a large number of people at the same time is called Mass Mailing with exciting new Features such as:
1. To attach files.
2. Use html tags in the body section
3. Able to send to more than 1000 mails at a time
4. Paste large no. of mails at once on a file and sent mail to each
5. Use Predefined body.html format

Project Members:
    Uikrant Panjiyar
    Bhupendra Chaulagain
    Jeevan Koiri

Press Any Key to Continue
```

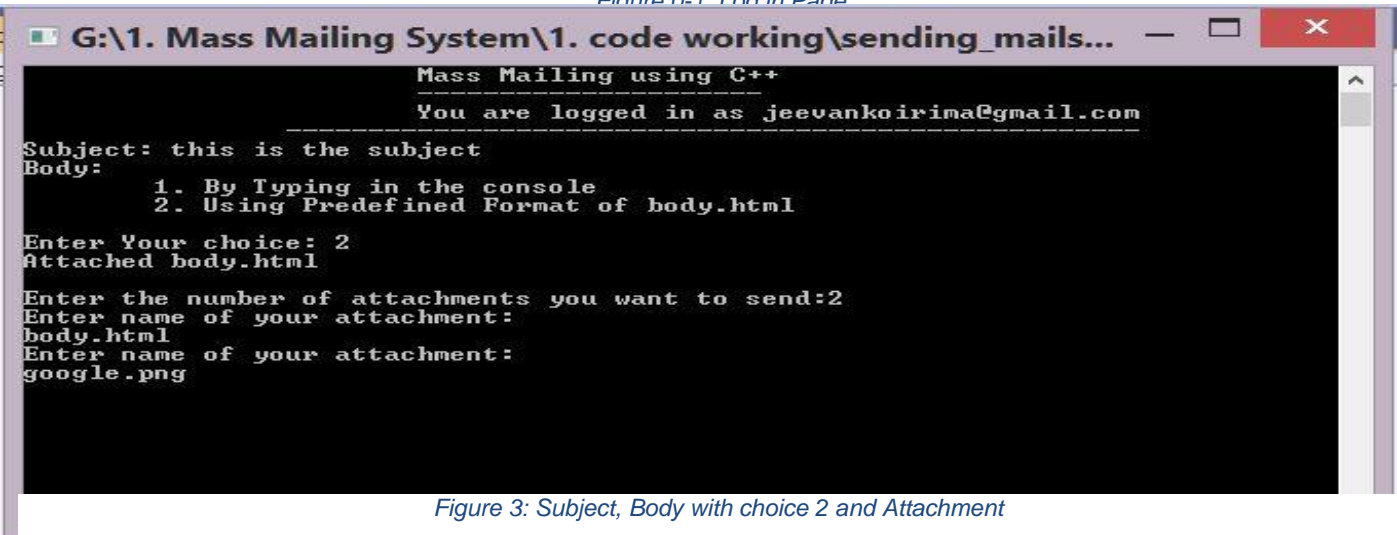
Figure 1: Front Page

A screenshot of a Windows command prompt window titled "G:\1. Mass Mailing System\1. code working\sending_mails...". The window displays the text "Mass Mailing using C++" followed by "Log In", "Email:jeevankoirima@gmail.com", and "Password: *****".

```
G:\1. Mass Mailing System\1. code working\sending_mails...
Mass Mailing using C++

Log In
Email:jeevankoirima@gmail.com
Password: *****
```

Figure 0-1: Log in Page

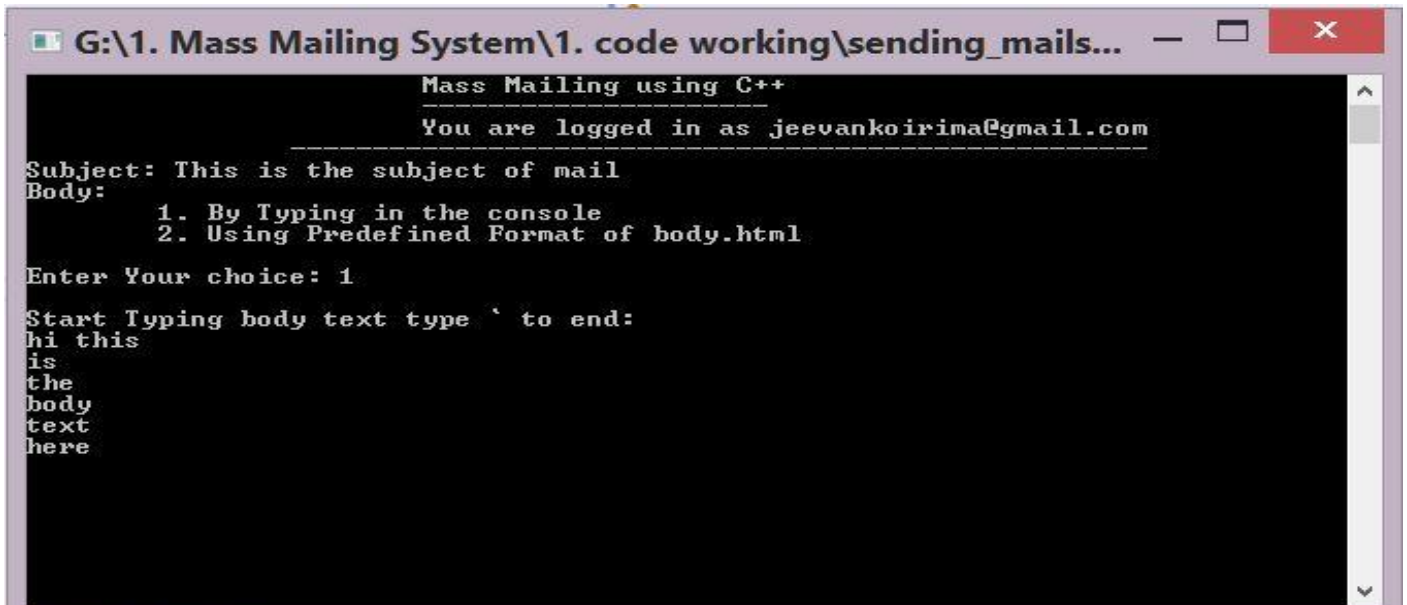
A screenshot of a Windows command prompt window titled "G:\1. Mass Mailing System\1. code working\sending_mails...". The window displays the text "Mass Mailing using C++" followed by "You are logged in as jeevankoirima@gmail.com". Below this, it prompts for "Subject: this is the subject" and "Body:". The body options are "1. By Typing in the console" and "2. Using Predefined Format of body.html". The user has chosen "2". It then prompts for "Enter Your choice: 2", "Attached body.html", "Enter the number of attachments you want to send:2", "Enter name of your attachment: body.html", and "Enter name of your attachment: google.png".

```
G:\1. Mass Mailing System\1. code working\sending_mails...
Mass Mailing using C++

You are logged in as jeevankoirima@gmail.com

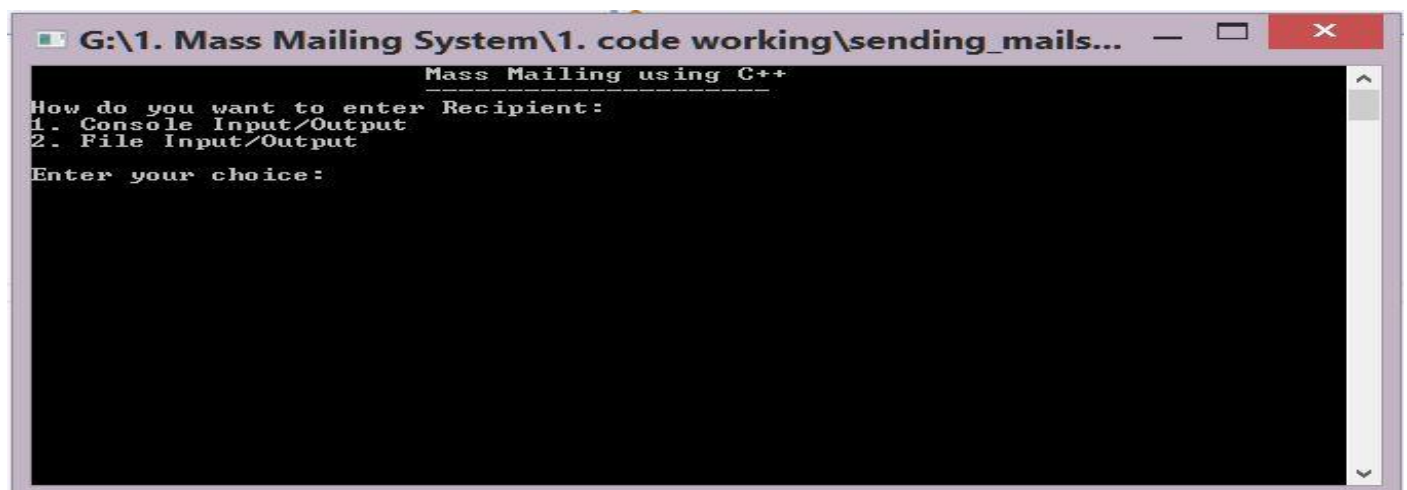
Subject: this is the subject
Body:
    1. By Typing in the console
    2. Using Predefined Format of body.html
Enter Your choice: 2
Attached body.html
Enter the number of attachments you want to send:2
Enter name of your attachment:
body.html
Enter name of your attachment:
google.png
```

Figure 3: Subject, Body with choice 2 and Attachment



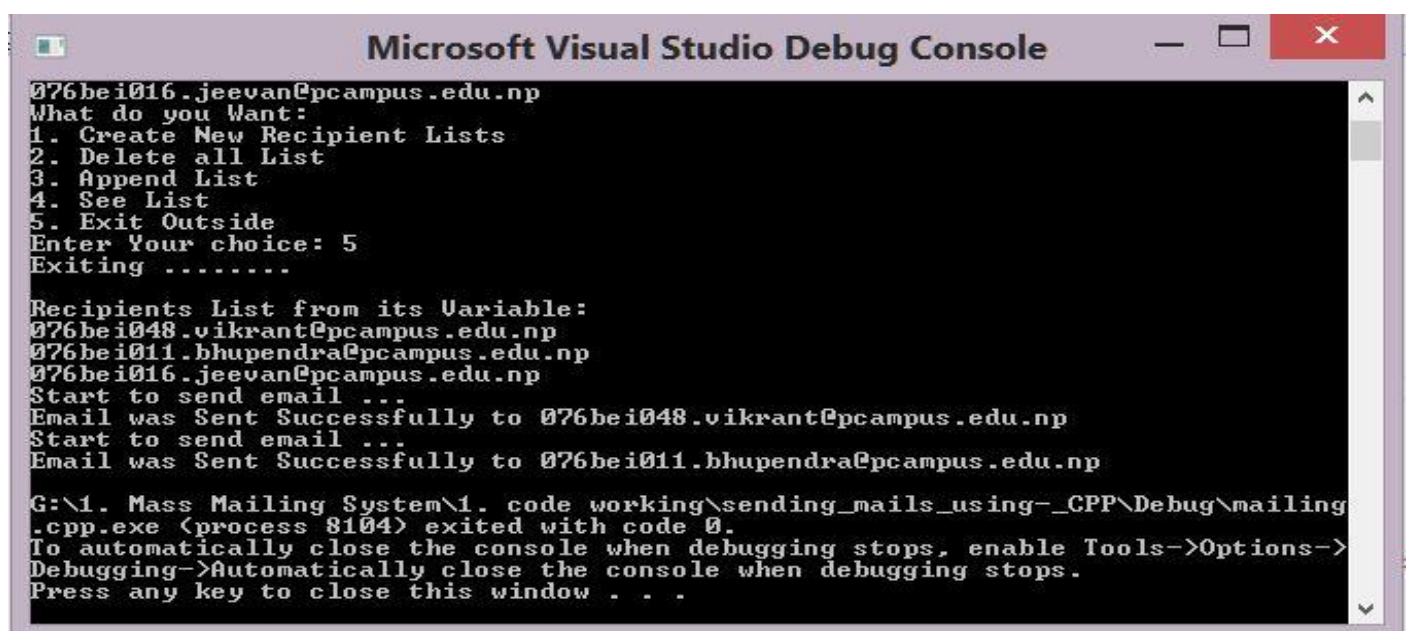
```
Mass Mailing using C++
-----
You are logged in as jeevankoirina@gmail.com
-----
Subject: This is the subject of mail
Body:
    1. By Typing in the console
    2. Using Predefined Format of body.html
Enter Your choice: 1
Start Typing body text type ` to end:
hi this
is
the
body
text
here
```

Figure 4: Body with choice 1



```
Mass Mailing using C++
-----
How do you want to enter Recipient:
1. Console Input/Output
2. File Input/Output
Enter your choice:
```

Figure 5: Entering the Recipients List



```
Microsoft Visual Studio Debug Console
076bei016.jeevan@pcampus.edu.np
What do you Want:
1. Create New Recipient Lists
2. Delete all List
3. Append List
4. See List
5. Exit Outside
Enter Your choice: 5
Exiting .....

Recipients List from its Variable:
076bei048.vikrant@pcampus.edu.np
076bei011.bhupendra@pcampus.edu.np
076bei016.jeevan@pcampus.edu.np
Start to send email ...
Email was Sent Successfully to 076bei048.vikrant@pcampus.edu.np
Start to send email ...
Email was Sent Successfully to 076bei011.bhupendra@pcampus.edu.np

G:\1. Mass Mailing System\1. code working\sending_mails_using_CPP\Debug\mailing
.cpp.exe (process 8104) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->
Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

Figure 6: Sending mails to each of the Recipients

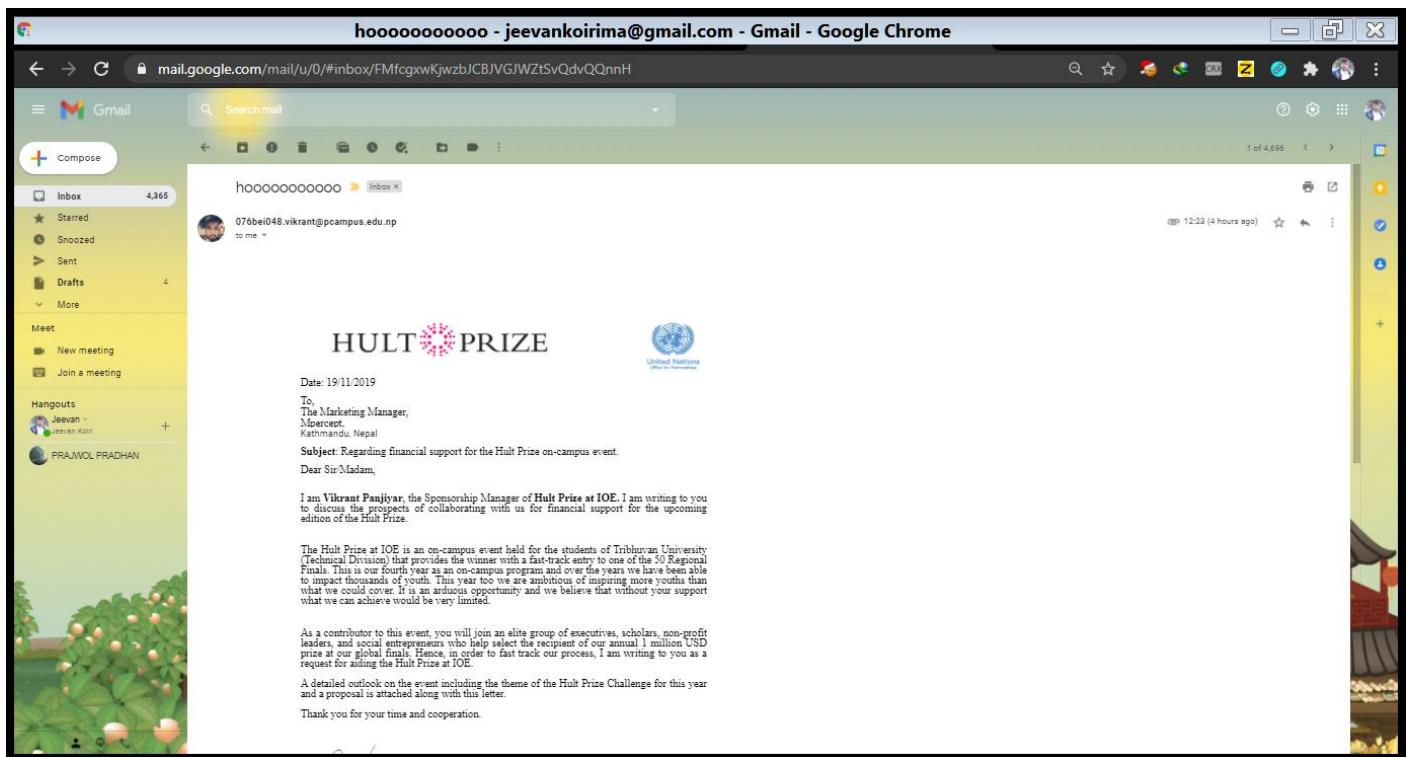


Figure7: Mail sent using the body.html and using Embedded Image of Hult Prize

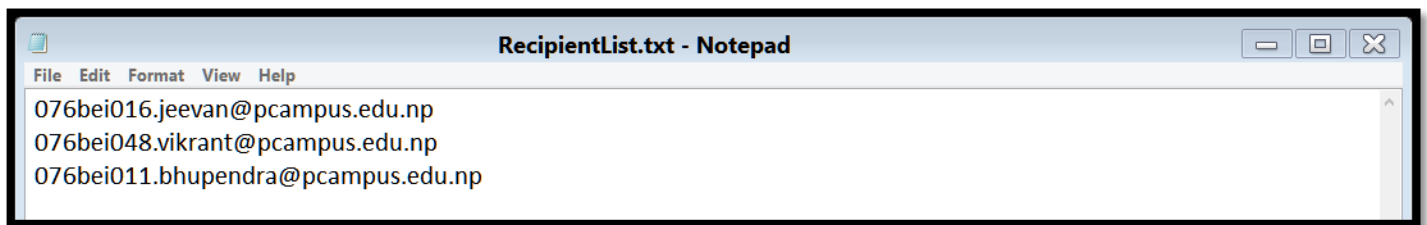


Figure 7: File Containing the Recipients List

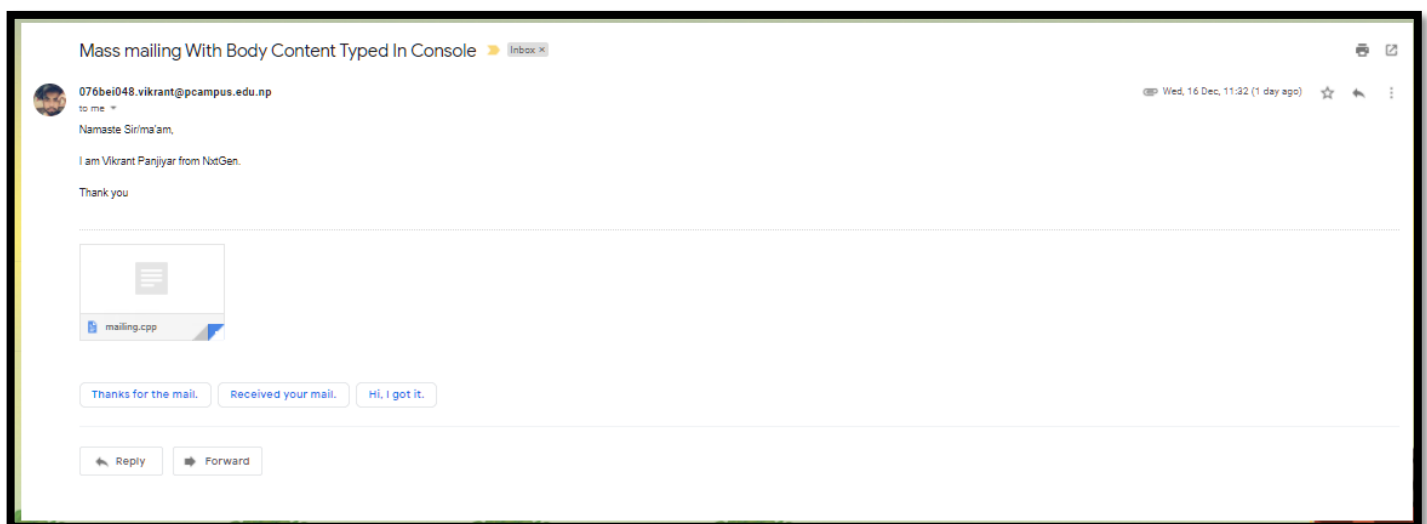


Figure 9: Mail Sent using a Attachment and By typing in the console

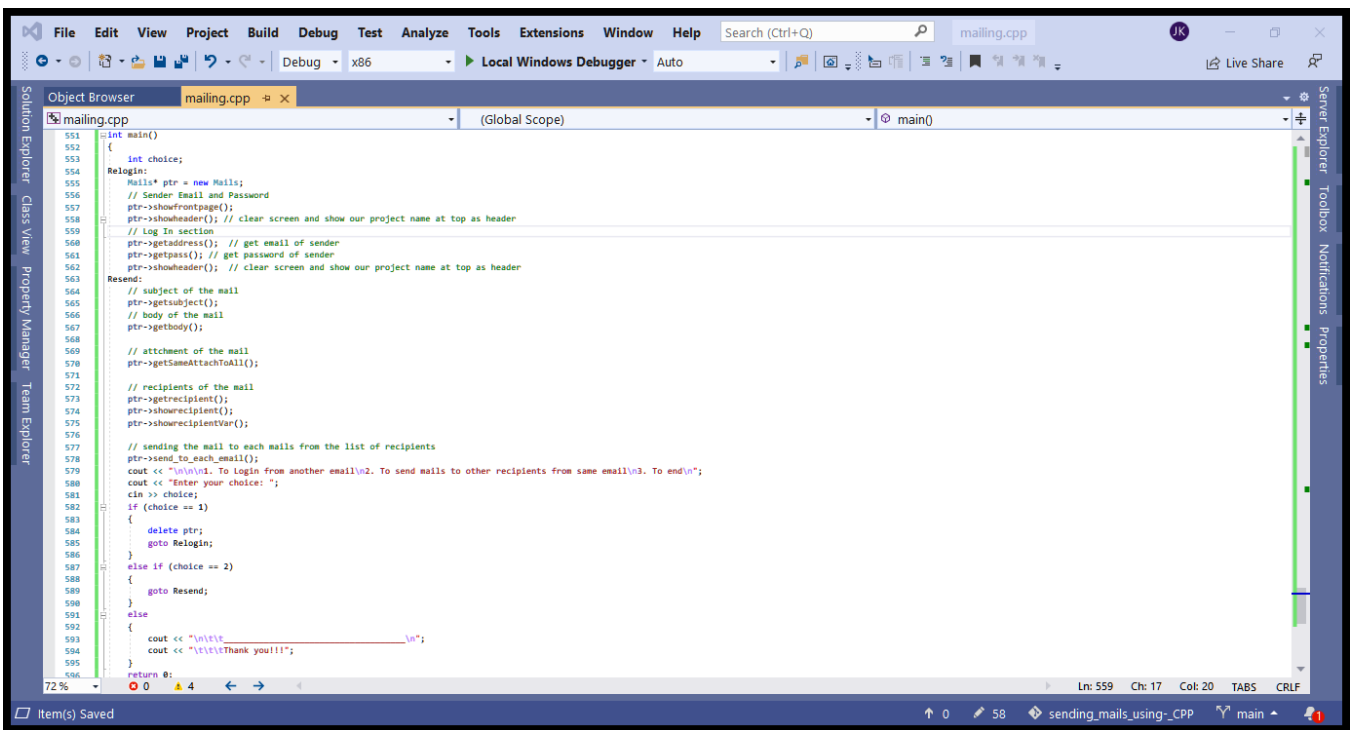


Figure10: Main program in Visual Studio



Figure 11: Mail Sent using the Signature

4.2. Coding of the project:

For the code we use the following sources:

<https://www.emailarchitect.net/easendmail/kb/vc.aspx?cat=0>

From this we downloaded the executable file for our pc that is required for sending the mail. We use the resources provided by this website and we are finally able to send mail having capability such as sending mails, sending html body, sending embedded image with the mail. This source really makes our project successful. Finally we are able to send mail to any number of recipients using the code we write.

We first write the success from which we are able to send the mail. Then later on we extend the functionality of our project. For Mass Mailing we simply use the Loop to send the same subject, same body text to a lot of persons. This enables us to make our project dynamic.

Further we came to a problem that when we type the email and password it is visible to everyone. So, we make a function for removing visibility of password and use * in place of a character typed. The code

Further we use the functionality of not seeing the password while typing in the console. When

The source code for our project (Mass Mailing Using C++) is provided at the end of this documentation.

For the code of this project refer to the Appendix A that is located at the end of this documentation.

Chapter Five: RESULTS AND DISCUSSION

The result was in accordance with our expectations. We were able to send mail to more thousand people at once using this project. The mail sent through a certain email was successfully received by the group of people which indicated the completion of the project. The project could be the useful result in various fields for sending mails and it drastically reduces the time of the user for sending the same mail.

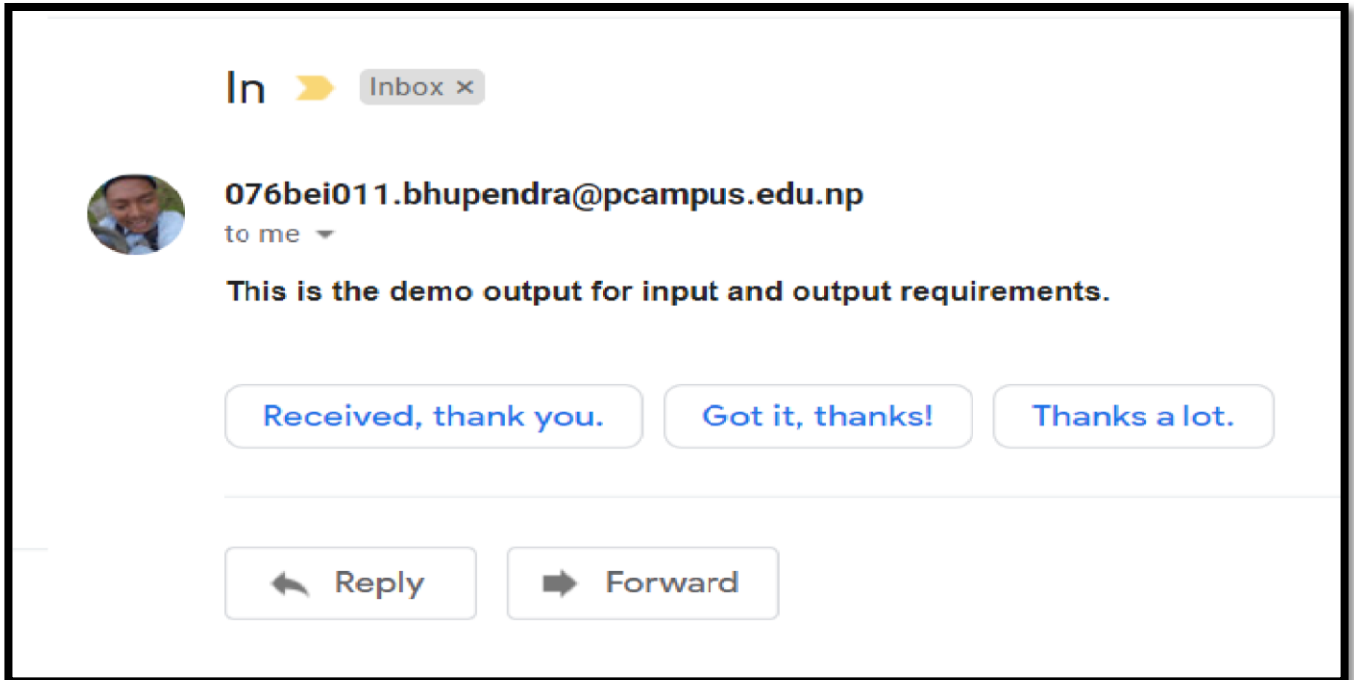


Fig: Demo of the result received.

Chapter Six: CONCLUSIONS

The project 'Mass Mailing' holds a great importance in sending mail to the collection of people at once that brought easiness as well as saves time that people need to spend to send mail individually. With the growth of technical advancement, mailing has become one of the essential factors whereas mass mailing has become one of the greatest needs in email marketing. To overcome this problem many people are engaging themselves in getting a better platform for making the task easier by sending bulk mails to all the recipients at once without letting them know it was the same mail sent to every person. Thus, through this project we have been able to handle the problem of mass mailing.

REFERENCES

ELECTRONIC/DIGITAL SOURCES

- ❑ Emailarchitect [Online Blog]

<https://www.emailarchitect.net/easendmail/kb/vc.aspx?cat=0>

- ❑ Github vcpkg Package manager for Visual Studio 2019

<https://github.com/microsoft/vcpkg>

- ❑ Vcpkg Youtube tutorial (video)

https://www.youtube.com/watch?v=URzE4V_kbb4

HARDCOPY SOURCES

- ❑ Object-oriented programming in Microsoft C++ a Book by Robert Lafore

UNPUBLISHED SOURCES

- ❑ Lectures by Our teacher Bikal Adhikari
- ❑ Lecture notes provided by our teacher Bikal Adhikari

APPENDIX A (Source Code)

```
// Change these first two lines according to your computer locations //
#define BODY_LOCATION "G:\\1. Mass Mailing System\\1. code working\\sending_mails_using-_CPP\\body.html" // Note
the double back slash here. It is required
#include "C:\\vcpkg\\downloads\\tools\\perl\\5.30.0.1\\c\\lib\\gcc\\i686-w64-mingw32\\8.3.0\\include\\stdfix.h"//SEARCH WHERE IS
STDFIX.H AND COPY THE PATH HERE TO INCLUDE ok

#include <tchar.h>
#include <Windows.h>
#include <iostream>
#include <conio.h> // for _getch() and _getche() // Here getche()
#include <cstdlib> // for system("cls");
#include <fstream>
#include <cstring>
#define size_recipients 1000

using namespace std;
const int ConnectNormal = 0;
const int ConnectSSLAuto = 1;
const int ConnectSTARTTLS = 2;
const int ConnectDirectSSL = 3;
const int ConnectTryTLS = 4;

#include "EASendMailObj.tlh"
#include <time.h>
#include <string>
using namespace EASendMailObjLib;

using namespace std;

class FrontPage
{
protected:
public:
    void showfrontpage()
    {
        system("cls");
        cout << "\\t\\tMass Mailing using C++" << endl;
        cout << "\\t\\t-----" << endl;
        cout << "\\n\\n\\n";
        cout << " The act of sending the same email message to a large number of people at the same time is called
Mass Mailing with exciting new Features such as:" << endl;
        cout << "1. To attach files." << endl;
        cout << "2. Use html tags in the body section" << endl;
        cout << "3. Able to send to more than 1000 mails at a time" << endl;
        cout << "4. Paste large no. of mails at once on a file and sent mail to each" << endl;
        cout << "5. Use Predefined body.html format" << endl;
        cout << "\\n\\n\\n";
        cout << "Project Members:\\n";
        cout << "\\tVikrant Panjiyar\\n";
        cout << "\\tBhupendra Chaulagain\\n";
        cout << "\\tJeevan Koiri\\n";
        cout << "\\n\\n\\t\\tPress Any Key to Continue";
        getch();
    }
};

class LogIn : public FrontPage
```

```

{
protected:
    char your_address[100];
    char your_password[50];
public:
    void showheader()
    {
        // this is the header
        system("cls");
        cout << "\t\tMass Mailing using C++" << endl;
        cout << "\t\t-----" << endl;
        // this is the session
    }
    //void showsession()
    //{
    //    cout << "\t\tYou are logged in as " << your_address << endl;
    //    cout << "\t\t-----" << endl;
    //}

    void getaddress()
    {
        cout << "\n\n\t\tLog In" << endl;
        // Set your sender email address
        cout << "\tEmail:";
        cin >> your_address;
    }

    // to make the password * and while typing in the console
    void getpass()
    {
        int i = 0; // used in the while loop
        char ch = '\0';
        char password[50]; // to store each character enter by the user

        // main loop for implementating password entering
        cout << "\tPassword: ";
        while (1)
        {
            ch = _getch(); // getting a character at each iteration

            if (ch != 8 && ch != 13) //
            {
                cout << "*";
                password[i] = ch;
                i++;
            }
            else if (ch == 8) // if backspace is presssed
            {
                cout << "\b \b"; // moves cursor to the left print <space> again move cursor to left
                i--;
            }
            else if (ch == 13) // 13 represents enter
            {
                password[i] = '\0'; // if enter is pressed, last character in match[] becomes null
                break; // for end of string
            }
            else
            {
                break;
            }
        }
        cout << endl;
        //cout<< endl << password;
        //doing this using the loop ---> your_pass = password;
        for (int i = 0; password[i] != '\0'; i++)

```

```

        {
            your_password[i] = password[i];
        }
        your_password[i] = '\0';
    }
};

```

```

class Recipients : public LogIn
{
protected:
    int no_of_recipients;
    int list_choice;
    int consoleOrfile;
    char recipients[1000][50];
public:
    void getrecipient()
    {
        system("cls");
        LogIn l;
        l.showheader();
        cout << "How do you want to enter Recipient:" << endl;
        cout << "1. Console Input/Output\n";
        cout << "2. File Input/Output\n";
        cout << "\nEnter your choice:";
        cin >> consoleOrfile;

        if (consoleOrfile == 2) // two represent file I/O
        {
            while (true)
            {
                // read the RecipientList.txt file and display the content
                system("cls");
                LogIn l1;
                l1.showheader();

                char ch;
                ifstream infile;
                infile.open("RecipientList.txt", ios::in);
                if (infile.fail()) //Error handling while opening a file
                {
                    cout << "The File has no Recipient List." << endl;
                    goto down_label;
                }
                cout << "\nThe File contains the following recipients:" << endl;
                no_of_recipients = 0;
                while (infile.get(ch))
                {
                    if (ch == '\n')
                    {
                        no_of_recipients++;
                    }
                    cout << ch;
                }
                infile.close();
            down_label:
                cout << endl << "What do you Want:" << endl;
                cout << "1. Create New Recipient Lists" << endl;
                cout << "2. Delete file" << endl;
                cout << "3. Append List" << endl;
                cout << "4. Search and delete a Recipient" << endl;
                cout << "5. Exit Outside" << endl;
                cout << "Enter Your choice: ";
                cin >> list_choice;
            }
        }
    }
};

```

```

if (list_choice == 1)
{
    // code for creating a file and ask to add the list using File IO
    char ch;
    ofstream outfile;
    outfile.open("RecipientList.txt", ios::out);
    cout << "Start writing the Recipients type ` to exit" << endl;
    while ((ch = cin.get()) != '\n')
    {
        outfile.put(ch);
    }
    outfile.close();
    cout << "File written!" << endl;
}
else if (list_choice == 2)
{
    remove("RecipientList.txt");
}
else if (list_choice == 3)
{
    // code for opening file in append mode and writing the content of the file
    char ch;
    ofstream out;
    out.open("RecipientList.txt", ios::app);
    cout << "Start writing the Recipient. Type ` to exit" << endl;
    while ((ch = cin.get()) != '\n')
    {
        out.put(ch);
    }
    out.close();
    cout << "File written!" << endl;
}
else if (list_choice == 4)
{
    char ch;
    char choiceyn;
    ifstream cinf;
    cinf.open("RecipientList.txt", ios::in);
    char name[50];
    char search_name[50];
    cout << "Enter the name of the recipient you want to search:";
    cin >> search_name;
    bool found;
    while (!cinf.eof())
    {
        cinf >> name;
        if (strcmp(name, search_name) == 0)
        {
            cout << "Found:";
            found = true;
        }
    }
    cinf.close();

    // deleting
    if (found)
    {
        cout << "Are you sure you want to delete(y/n): ";
        cin >> choiceyn;
        if (choiceyn == 'y')
        {
            cout << "deleting..." << endl;
            ////
            string line;

```

```

        ifstream myfile;
        ofstream temp;
        myfile.open("RecipientList.txt");
        temp.open("temp.txt");
        while (getline(myfile, line))
        {
            if (line != search_name)
                temp << line << endl;
        }
        cout << "The record with the name " << search_name << " has
been deleted" << endl;

        myfile.close();
        temp.close();
        remove("RecipientList.txt"); // to delete the file directly
        rename("temp.txt", "RecipientList.txt"); // to rename the file
        ///
        cout << "Deleted" << search_name << endl;
    }
}
else
{
    cout << "Not Found\n";
}
}
else if (list_choice == 5)
{
    cout << "Exiting ..... \n";
    break;
}
else
{
    cout << "You typed wrong." << endl;
    system("cls");
}
}
}
else // if(consoleOrfile == 1 or anything by default)
{
    int i = 0;
    no_of_recipients = 0;
    cout << "Enter Address of Recipients. Type stop to stop \n";
    while (true)
    {
        cin >> recipients[i];
        if (strcmp(recipients[i], "stop") == 0)
        {
            break;
        }

        i++;
        no_of_recipients++;
    }
}
}
}
void showrecipient()
{
    if (consoleOrfile == 1)
    {
        cout << "Recipients List:" << endl;
        for (int i = 0; i <= no_of_recipients; i++)
        {
            cout << recipients[i] << endl;
        }
    }
}

```



```

    }
else // for file if (consoleOrfile == 2)
{

    // Creation of ifstream class object to read the file
    ifstream fin;
    // by default open mode = ios::in mode
    fin.open("RecipientList.txt");
    int i = 0;
    // Execute a loop until EOF (End of File)
    while (fin) {

        // Read a Line from File and store it in a variable

        fin >> recipients[i];
        i++;
    }
    // Close the file
    fin.close();
}

} // end of the function
void showrecipientVar()
{
    cout << "\nRecipients List from its Variable:" << endl;
    for (int i = 0; i <= no_of_recipients; i++)
    {
        cout << recipients[i] << endl;
    }
}

};

// class to deal with the subject entry
class Subject : public Recipients
{
protected:
    char your_subject[100];
    char subject[100];
    char ch;
public:
    void getsubject()
    {
        int j = 0; // used in the while loop
        char ch = '\0';
        // main loop for implementating your_subject entering
        cout << "Subject: ";
        while (1)
        {
            ch = _getche(); // getting a character at each iteration

            if (ch != 8 && ch != 13)
            {
                your_subject[j] = ch;
                j++;
            }
            if (ch == 8) // if backspace is pressed
            {
                cout << "\b"; // backspace means clearing the screen from the console
                j--;
            }
            else if (ch == 13) // 13 represents enter
            {
                your_subject[j] = '\0'; // if enter is pressed, last character in match[] becomes null
                break; // for end of string
            }
        }
    }
}

```

```

    }
}
void showsubject()
{
    cout << "Subject:" << your_subject << endl;
}
};

// for handling the body
class Body : public Subject
{
protected:
    char your_body[100000];
    char ch;
    int choice_body;
public:

    void showbody()
    {
        if (choice_body == 1)
            cout << "Body:" << your_body << endl;
        else
            cout << "Attached body.html\n Go and edit the body.html in the directory to change the format of the
mail.\n";
    }
};

class Attachments : public Body
{
protected:
    int no_of_attachments;
    char your_attachments[100][50];
public:
    //set attachment
    void getSameAttachToAll()
    {
        cout << "\nEnter the number of attachments you want to send:";
        cin >> no_of_attachments;
        for (int i = 0; i < no_of_attachments; i++)
        {
            cout << "Enter name of your attachment:" << endl;
            cin >> your_attachments[i];
        }
    }
};

// Deals with sending the mail to every recipients
class Mails : public Attachments
{
protected:
    int choice_body;
public:

    void send_to_each_email()
    {
        for (int i = 0; i < no_of_recipients; i++)
        {
            mailSenderWithSameAttachmentToAll(recipients[i], your_address, your_password, your_body,
your_attachments, no_of_attachments);
        }
    }

    void mailSenderWithSameAttachmentToAll(char recipient[50], char your_address[100], char your_password[50], char
your_body[100000], char your_attachments[100][50], int no_of_attachments)

```

```

{

::CoInitialize(NULL);

IMailPtr oSmtmp = NULL;
oSmtmp.CreateInstance(__uuidof(EASendMailObjLib::Mail));
oSmtmp->LicenseCode = _T("TryIt");
oSmtmp->FromAddr = _T(your_address);


oSmtmp->AddRecipientEx(_T(recipient), 0); //to whom You want to send
// Set email body

//set attachment
for (int i = 0; i < no_of_attachments; i++)
{
    oSmtmp->AddAttachment(your_attachments[i]);
}


// Set HTML body format. BodyFormat property must be set to 1 (text/html) to take effect.
oSmtmp->BodyFormat = 1;
if (choice_body == 1)
{
    oSmtmp->BodyText = your_body;
}
else
{
    // Set HTML body text
    // google.png will be imported as embedded image in this email
    oSmtmp->ImportMailEx(_T(BODY_LOCATION));
}
// Your SMTP server address
oSmtmp->ServerAddr = _T("smtp.gmail.com");

oSmtmp->Subject = _T(your_subject);

// User and password for ESMTP authentication, if your server doesn't
// require User authentication, please remove the following codes.
oSmtmp->UserName = _T(your_address);
oSmtmp->Password = _T(your_password);
// Most modern SMTP servers require SSL/TLS connection now.
// ConnectTryTLS means if server supports SSL/TLS, SSL/TLS will be used automatically.
oSmtmp->ConnectType = ConnectTryTLS;


// If your SMTP server uses 587 port
oSmtmp->ServerPort = 587;


// If your SMTP server requires SSL/TLS connection on 25/587/465 port
//oSmtmp->ServerPort = 25 or 587 or 465;
// oSmtmp->ConnectType = ConnectSSLAuto;

cout << "Start to send email ...\r" << endl;

if (oSmtmp->SendMail() == 0)
{
    cout << "Email was Sent Successfully to " << recipient << endl;
}
else
{
    cout << "Failed to send email with the following error:\n" << (const TCHAR*)oSmtmp-
>GetLastErrDescription();
}
} // end of mail sender

```

```

// get body
void getbody()
{
    cout << "\n";
    // main loop for implementating your_subject entering
    cout << "Body: " << endl;
    cout << "\t1. By Typing in the console" << endl;
    cout << "\t2. Using Predefined Format of body.html" << endl;
    cout << "\nEnter Your choice: ";
    cin >> choice_body;

    // Set HTML body format. BodyFormat property must be set to 1 (text/html) to take effect.
    //oSmt->BodyFormat = 1;

    if (choice_body == 1)
    {
        int j = 0; // used in the while loop
        char ch = '\0';
        cout << "\nStart Typing body text type ` to end:" << endl;
        while (1)
        {
            ch = _getche(); // getting a character at each iteration

            if (ch != 8 && ch != 13 && ch != '^')
            {
                your_body[j] = ch;
                j++;
            }
            else if (ch == 8) // if backspace is pressed
            {
                cout << " \b"; // backspace means clearing the screen from the console
                j--;
            }
            else if (ch == 13) // 13 represents enter
            {
                cout << "\n";
                your_body[j] = '<'; // if enter is pressed, last character in match[] becomes null
                j++;
                your_body[j] = 'b';
                j++;
                your_body[j] = 'r';
                j++;
                your_body[j] = '>'; // <br> in html represents a new line
                j++;
            }
            else
            {
                your_body[j] = '\0';
                //oSmt->BodyText(your_body);
                break; // for end of loop
            }
        } // end of while loop
    } // end of if condition
    else
    {
        //oSmt->ImportMailEx(_T("G:\\1. Mass Mailing System\\1. code working\\sending_mails_using-
_CPP\\body.html"));
        cout << "Attached body.html" << endl;
    }
}

};

int main()
{

```

```

    int choice;
Relogin:
    Mails* ptr = new Mails;
    // Sender Email and Password
    ptr->showfrontpage();
    ptr->showheader(); // clear screen and show our project name at top as header
    // Log In section

    ptr->getaddress(); // get email of sender
    ptr->getpass();    // get password of sender

    ptr->showheader(); // clear screen and show our project name at top as header

    //ptr->showsession(); // view your logged in information
Resend:

    // subject of the mail
    ptr->getsubject();
    // body of the mail
    ptr->getbody();

    // attchment of the mail
    ptr->getSameAttachToAll();

    // recipients of the mail
    ptr->getrecipient();
    ptr->showrecipient();
    ptr->showrecipientVar();

    // sending the mail to each mails from the list of recipients
    ptr->send_to_each_email();
    cout << "\n\n1. To Login from another email\n2. To send mails to other recipients from same email\n3. To end\n";
    cout << "Enter your choice: ";
    cin >> choice;
    if (choice == 1)
    {
        delete ptr;
        goto Relogin;
    }
    else if (choice == 2)
    {
        goto Resend;
    }
    else
    {
        cout << "\n\t\t_____ \n";
        cout << "\t\tThank you!!!";
    }

    return 0;

}

```