DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

ST JOSEPH ENGINEERING COLLEGE, MANGALURU-28

Computer Networks(18CS52/17CS52/15CS52) Assignment Report

2020-2021

Topic Name:	Mail Client Application – Bulk Mailer	
Team No:	11	
Team Members:	**USN**	**Name**
	4SO18CS068	Laureen Maria Fernandes
	4SO18CS092	Raksharaj Shetty
	4SO18CS103	Shanwill Pinto
	4SO18CS123	Vignesh
Submission Date:	04/01/2021	
Total Marks Awarded:		
Staff Signature:		

COMPUTER NETWORKS ASSIGNMENT RUBRICS

	4	3	2	1	0	Marks Awarded
Meets Computational Specifications (2 x) **Max: 8 marks**	The program meets all of the computational specifications	The program produces the correct results and displays them correctly for almost all computational specifications	The program produces correct results for most computational specs, has a few bugs	The program produces incorrect results, has several bugs	The program does not work or has many bugs	
Readability **(1 X)** **Max: 4 marks**	The code is well organized and very easy to understand, with clear comments both in-line and in headers	The code is pretty well organized, fairly easy to read, and has good comments	The code has some organization, is a challenge to read, and has minimal comments	The code is readable only by someone who knows what it is supposed to do, has few comments	The code is poorly organized and very difficult to read, with no comments	
Error Handling **(1 X)** **Max: 4 marks**	The program checks for all error conditions and handles them appropriately	The program checks for most error conditions and handles them appropriately	The program checks for some error conditions and handles them appropriately	The program checks for few error conditions and doesn't handle them appropriately	The program does not check error conditions	
Individual Contribution **(1 X)** **Max: 4 marks**	The individual contributed in a valuable way to the project. The individual is able to articulate the working of the program.	The individual did not contribute as heavily as other members but did meet all the responsibilities. Able to articulate the program	The individual did not contribute. Did meet all responsibilities but not able to articulate the working of program	The individual did not contribute neither did meet all the responsibilities but is able to articulate the working of program	The individual did not contribute nor met all responsibilities. Failed to articulate the working of program	

Total Marks:

CONTENTS

CHAPTER 1: INTRODUCTION

1.1 Overview of the Topic:

Bulk Mailer is a Mail Client web application that can be used by organisations to send bulk emails for different groups of subscribers. In general, a bulk email service is a company that allows its customers to send mass email messages to multiple lists of recipients at a specified time. With this service, you can send a single message to thousands of people on a mailing list or a personalized email to each address on a list that can be of any size.

Today, marketers prefer to use bulk email services to deliver important messages with minimal effort. Unlike junk emails sent without the recipients' permission, bulk emails are legal marketing campaigns since the recipients subscribe to receive them. However, if bulk email marketing is not properly managed, users may consider it spam, and consequently, it may hurt sender reputation.

Most of the bulk email service providers price their offerings based on the number and frequency of the emails one wants to send. But, after registering with Bulk Mailer, you can send bulk emails free of charge!

Our application has a feature-rich email builder that lets you build beautiful and responsive emails in minutes. It supports adding and using different email templates as well which ensures consistency and reduced human efforts.

1.2 Scope and Importance:

Bulk mailing is an incredibly useful tool for any business as it aims to promote a business or sell goods or even develop relationships. Sending thousands or tens of thousands of messages to even just a couple of email addresses would be draining due to the amount of time and effort required. Moreover, the cost of running such a campaign would not be sustainable for any business. Using a bulk email service is cheaper, faster, and much more convenient.

This service is a prime example of how you can utilise technology to enhance traditional marketing methods. Time-saving is, of course, one of the big advantages with bulk mailing, but there are plenty of other benefits too like the ability to spark engagement. More businesses are now seeing the benefits of combining direct mail campaigns with digital marketing methods.

CHAPTER 2: SOFTWARE REQUIREMENTS SPECIFICATION

2.1 Functional Requirements:

This section documents the operation and activities that the application must be able to perform, which are,

- Register – User should be able to create an account on the website. On successful registration, he/she must contact his/her organisation for account activation.
- Login - User should be able to login using the login portal. On successful login, he/she will be redirected to the dashboard.
- Bulk Mailing – User should be able to send organisational mails in bulk to the specified group of subscribers.
- Template Management – User should be able to create and use existing email templates according to his/her convenience.
- Groups – User should be able to maintain groups of subscribers based on different bulk mailing categories.
- Logout – User should be able to logout using the logout button.
- API – Interested public should be able to subscribe and unsubscribe using API calls.

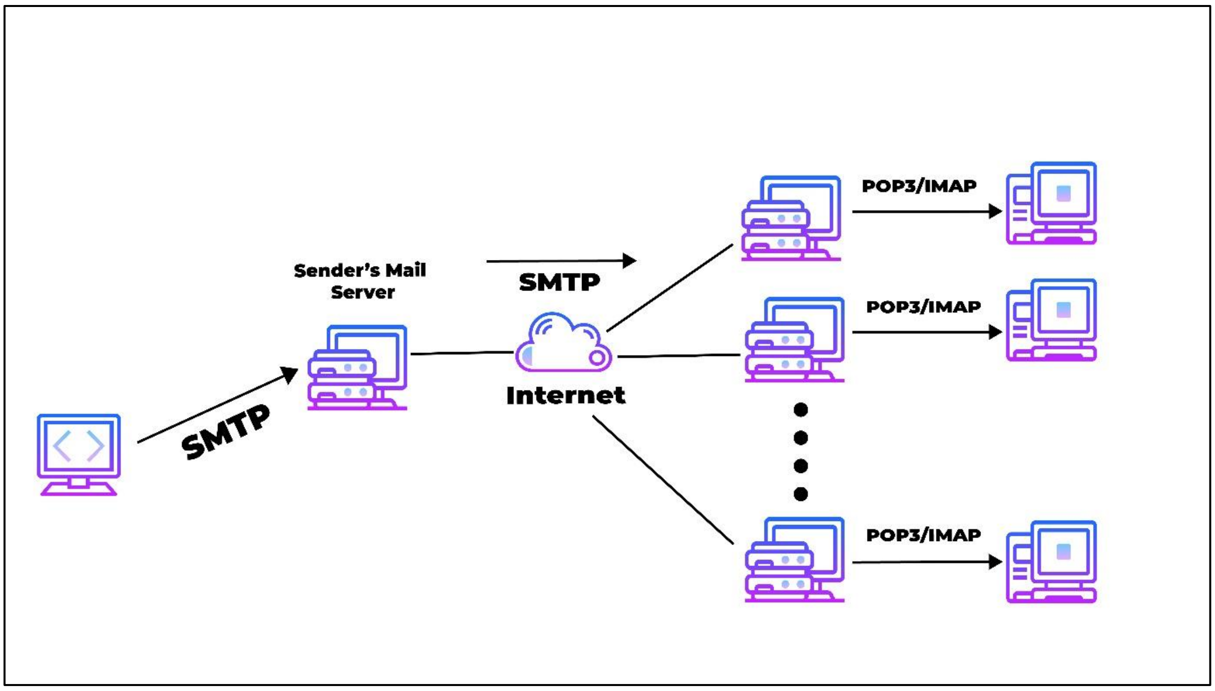
2.2 Non-functional Requirements:

This section deals with the quality of the web application rather than the functionalities by describing the non-functional requirements which are,

- Usability requirements – Web app should provide a user-friendly interface with a GUI and so, must be easy to use.
- Efficiency requirements – Web app should respond to user input promptly and display the necessary output.
- Space requirements – Since it is a web app, there should be no need to install anything.
- Portability requirements – Responsive design should be implemented so that the web app can be used on any device.

CHAPTER 3: DESIGN

3.1 Abstract Design:



CHAPTER 4: IMPLEMENTATION

4.1 Pseudo Code:

```
Step 1: Start
Step 2: Get username, name, subject, group, email content from the form
Step 3: Generate the from email i.e., from_email = username + domain_name
Step 4: Generate the mail list by extracting the emails of all subscribers in the specified group.
Step 5: Generate the mail message to be sent by using from_email, mail list, subject and email content.
Step 6: Using SendGrid API, send the email. Email is sent using the SMTP protocol and TCP connection.
Step 7: Stop
```

4.2 Code Implementation:

```python
from flask import Flask, render_template, request, redirect, url_for, flash
from flask_login import LoginManager, UserMixin, current_user, login_user, log
out_user, login_required
from flask_sqlalchemy import SQLAlchemy
from datetime import datetime
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import Mail
from passlib.hash import sha256_crypt
import json, random, string, psycopg2


#load import.json file containing database uri, admin email and other impt inf
o
with open('import.json', 'r') as c:
    json = json.load(c)["jsondata"]

#create a Flask app and setup its configuration
app = Flask(__name__)
app.secret_key = "76^)(HEY,BULK-MAILER-
HERE!)(skh390880213%^*&%6h&^&69lkjw*&kjh"
app.config['SQLALCHEMY_DATABASE_URI'] = json["databaseUri"]
db = SQLAlchemy(app)

#use LoginManager to provide login functionality and do some initial confg
login_manager = LoginManager(app)
login_manager.login_view = 'login'
login_manager.login_message_category = 'info'

#function to load the currently active user
```

```
@login_manager.user_loader
def load_user(user_id):
    return Organization.query.get(user_id)

'''DATABASE MODELS'''

#represents a group of users to whom a specific email can be sent
class Group(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(50), nullable=False)
    date = db.Column(db.String(50), nullable=False)
    subscribers = db.relationship('Subscriber',cascade = "all,delete", backref
='subscribers')

#represents a subscriber that belongs to a group
class Subscriber(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    email = db.Column(db.String(50), nullable=False)
    date = db.Column(db.String(50), nullable=False)
    group_id = db.Column(db.Integer, db.ForeignKey('group.id'))

#represents an email template
class Template(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(50), nullable=False)
    link = db.Column(db.String(100), nullable=False)
    content = db.Column(db.String(500), nullable=False)
    date = db.Column(db.String(50), nullable=False)

#represents a user in an organisation
#currently only one organisation with multiple users is supported
class Organization(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(50), nullable=False)
    email = db.Column(db.String(50), nullable=False)
    password = db.Column(db.String(500), nullable=False)
    status = db.Column(db.Integer , nullable=False)
    date = db.Column(db.String(50), nullable=False)

'''END OF DATABASE MODELS'''

#generate a random 8 lettered password for forgot password
letters = string.ascii_letters
new_password = ''.join(random.choice(letters) for i in range(8))

#convert the current datetime to string, to be stored in the db
x = datetime.now()
time = x.strftime("%c")
```

```
#domain name
domain='@bulkmailer.cf'

#login route
@app.route('/login', methods = ['GET', 'POST'])
def login():
    #check if user is authenticated
    if current_user.is_authenticated:
        #if true, go to the dash page
        return redirect(url_for('dash_page'))
    #check if a form has been submitted i.e., user has tried to login
    if (request.method == 'POST'):
        #get the data in the email, password, and remember me fields
        email = request.form.get('email')
        password = request.form.get('password')
        remember = request.form.get('remember')
        #get user with the email entered by querying the database
        user = Organization.query.filter_by(email=email).first()
        #check if user exists
        if not user:
            #if user doesn't exist i.e., email not found, flash an error
            flash('Valid account not found!', 'danger')
            return render_template('login.html', json=json)
        elif ( sha256_crypt.verify(password, user.password ) == 1) and (user.s
tatus == 1):
            #if user exists and correct password has been entered and the user
's account has been activated
            #update the last login to current date and add it to the db
            user.date = time
            db.session.add(user)
            db.session.commit()
            #log the user in using login_user
            login_user(user, remember=remember)
            #go to the page that the user tried to access if exists
            #otherwise go to the dash page
            next_page = request.args.get('next')
            return redirect(next_page) if next_page else redirect(url_for('das
h_page'))
        else:
            #user doesn't exist so flash an error
            flash('Account not activated or invalid credentials!', 'danger')
    return render_template('login.html', json=json)

#logout route
@app.route('/logout')
@login_required
def logout():
    #log the user out using logout_user, flash a msg and go to the login page
    logout_user()
    flash('Logged Out Successfully!', 'success')
    return redirect(url_for('login'))

#register route
@app.route('/register',methods = ['GET', 'POST'])
def register_page():
    #check if form has been submitted i.e., user has tried to register
    if (request.method == 'POST'):
        #get the data in name, email, and password fields
        name = request.form.get('name')
        email = request.form.get('email')
        password = request.form.get('password')
        password2 = request.form.get('password2')
        #check if passwords match
        if(password!=password2):
            #if not, flash an error msg
            flash("Password unmatched!", "danger")
            return render_template('register.html', json=json)
        else:
            #generate the hashed password
            password = sha256_crypt.hash(password)
            response = Organization.query.filter_by(email=email).first()
            #check if the email already exists in the db
            if not response:
                #add the user to the db using the details entered and flash a
msg
                entry = Organization(name=name, email=email, password=password
, date=time, status=0)
                db.session.add(entry)
                db.session.commit()
                flash("Now contact your organization head for account activati
on!", "success")
                #generate the welcome email to be sent to the user
                subject = "Welcome aboard " + name + "!"
                content = '''<!DOCTYPE html><html xmlns="http://www.w3.org/199
9/xhtml" xmlns:v="urn:schemas-microsoft-com:vml" xmlns:o="urn:schemas-
microsoft-com:office:office"><head><title>Reset Your Password</title> <!--
[if !mso]> --><meta http-equiv="X-UA-Compatible" content="IE=edge"> <!--
<![endif]--><meta http-equiv="Content-Type" content="text/html; charset=UTF-
8"><meta name="viewport" content="width=device-width, initial-
scale=1.0"><style type="text/css">#outlook a{padding:0}.ReadMsgBody{width:100%
}.ExternalClass{width:100%}.ExternalClass *{line-
height:100%}body{margin:0;padding:0;-webkit-text-size-adjust:100%;-ms-text-
size-adjust:100%}table,td{border-collapse:collapse;mso-table-lspace:0pt;mso-
table-rspace:0pt}img{border:0;height:auto;line-height:100%;outline:none;text-
decoration:none;-ms-interpolation-
mode:bicubic}p{display:block;margin:13px 0}</style><style type="text/css">@med
ia only screen and (max-width:480px){@-ms-
viewport{width:320px}@viewport{width:320px}}</style><style type="text/css">@me
dia only screen and (min-width:480px){.mj-column-per-
100{width:100%!important}}</style></head><body style="background: #f0f0f0;"><d
iv class="mj-container" style="background-
color:#f0f0f0;"><table role="presentation" cellpadding="0" cellspacing="0" sty
le="background:#f0f0f0;font-
size:0px;width:100%;" border="0"><tbody><tr><td><div style="margin:0px auto;ma
x-
width:600px;"><table role="presentation" cellpadding="0" cellspacing="0" style
="font-
size:0px;width:100%;" align="center" border="0"><tbody><tr><td style="text-
align:center;vertical-align:top;direction:ltr;font-
size:0px;padding:0px 0px 0px 0px;"><div class="mj-column-per-100 outlook-
group-fix" style="vertical-align:top;display:inline-block;direction:ltr;font-
size:13px;text-
align:left;width:100%;"><table role="presentation" cellpadding="0" cellspacing
="0" width="100%" border="0"><tbody><tr><td style="word-wrap:break-word;font-
size:0px;"><div style="font-size:1px;line-height:30px;white-
space:nowrap;">&#xA0;</div></td></tr></tbody></table></div></td></tr></tbody><
/table></div></td></tr></tbody></table><div style="margin:0px auto;max-
width:600px;background:#FFFFFF;"><table role="presentation" cellpadding="0" ce
llspacing="0" style="font-
size:0px;width:100%;background:#FFFFFF;" align="center" border="0"><tbody><tr>
<td style="text-align:center;vertical-align:top;direction:ltr;font-
size:0px;padding:9px 0px 9px 0px;"><div class="mj-column-per-100 outlook-
group-fix" style="vertical-align:top;display:inline-block;direction:ltr;font-
size:13px;text-
align:left;width:100%;"><table role="presentation" cellpadding="0" cellspacing
="0" width="100%" border="0"><tbody><tr><td style="word-wrap:break-word;font-
size:0px;padding:25px 25px 25px 25px;" align="center"><table role="presentatio
n" cellpadding="0" cellspacing="0" style="border-collapse:collapse;border-
spacing:0px;" align="center" border="0"><tbody><tr><td style="width:204px;"> <
img alt="" title="" height="100px" width="100px" src="https://cdn.discordapp.c
om/attachments/577137963985534994/791571694803353610/favicon.ico" style="borde
r:none;border-radius:0px;display:block;font-size:13px;outline:none;text-
decoration:none;width:100%;height:auto;" width="204"></td></tr></tbody></table
></td></tr><tr><td style="word-wrap:break-word;font-
size:0px;padding:0px 15px 0px 15px;" align="center"><div style="cursor:auto;co
lor:#333333;font-family:Helvetica, sans-serif;font-size:15px;line-
height:22px;text-align:center;"><h3 style="font-family: Helvetica, sans-
serif; font-size: 24px; color: #333333; line-
height: 50%;">Hey, Welcome!</h3></div></td></tr><tr><td style="word-
wrap:break-word;font-
size:0px;padding:0px 50px 0px 50px;" align="center"><div style="cursor:auto;co
lor:#333333;font-family:Helvetica, sans-serif;font-size:15px;line-
height:22px;text-
```

```
align:center;"><p>Now contact your organization head for account activation.</
p></div></td></tr><tr><td style="word-wrap:break-word;font-
size:0px;padding:20px 25px 20px 25px;padding-top:10px;padding-
left:25px;" align="center"><table role="presentation" cellpadding="0" cellspac
ing="0" style="border-
collapse:separate;" align="center" border="0"><tbody><tr><td style="border:non
e;border-
radius:5px;color:#FFFFFF;cursor:auto;padding:10px 25px;" align="center" valign
="middle" bgcolor="#4DAA50"><a href="https://bulkmailer.cf" style="text-
decoration: none; background: #4DAA50; color: #FFFFFF; font-
family: Helvetica, sans-serif; font-size: 19px; font-weight: normal; line-
height: 120%; text-
transform: none; margin: 0px;" target="_blank">Visit Website</a></td></tr></tb
ody></table></td></tr><tr><td style="word-wrap:break-word;font-
size:0px;padding:0px 47px 0px 47px;" align="center"><div style="cursor:auto;co
lor:#333333;font-family:Helvetica, sans-serif;font-size:15px;line-
height:22px;text-align:center;"><p><span style="font-
size:14px;"><strong>Questions?&#xA0;</strong><br>Email us at <a href="mailto:e
mail@bulkmailer.cf" style="color: #555555;">email@bulkmailer.cf</a>.&#xA0;</sp
an></p></div></td></tr></tbody></table></div></td></tr></tbody></table></div><
table role="presentation" cellpadding="0" cellspacing="0" style="background:#f
0f0f0;font-
size:0px;width:100%;" border="0"><tbody><tr><td><div style="margin:0px auto;ma
x-
width:600px;"><table role="presentation" cellpadding="0" cellspacing="0" style
="font-
size:0px;width:100%;" align="center" border="0"><tbody><tr><td style="text-
align:center;vertical-align:top;direction:ltr;font-
size:0px;padding:0px 0px 0px 0px;"><div class="mj-column-per-100 outlook-
group-fix" style="vertical-align:top;display:inline-block;direction:ltr;font-
size:13px;text-
align:left;width:100%;"><table role="presentation" cellpadding="0" cellspacing
="0" width="100%" border="0"><tbody><tr><td style="word-wrap:break-word;font-
size:0px;padding:0px 98px 0px 98px;" align="center"><div style="cursor:auto;co
lor:#777777;font-family:Helvetica, sans-serif;font-size:15px;line-
height:22px;text-align:center;"><p><span style="font-
size:12px;"><a href="https://bulkmailer.cf" style="color: #555555;">TERMS OF S
ERVICE</a> | <a href="https://bulkmailer.cf" style="color: #555555;">PRIVACY P
OLICY</a><br>&#xA9; 2020 Bulk Mailer<br><a href="https://bulkmailer.cf/unsubsc
ribe" style="color: #555555;">UNSUBSCRIBE</a></span></p></div></td></tr></tbod
y></table></div></td></tr></tbody></table></div></td></tr></tbody></table></di
v></body></html>'''
                message = Mail(
                    from_email=('register@bulkmailer.cf', 'Bulk Mailer Registe
r'),
                    to_emails=email,
                    subject=subject,
                    html_content=content)
                try:
                    #using the sendgrid api, send the email to the user's emai
l
                    sg = SendGridAPIClient(json['sendgridapi'])
                    response = sg.send(message)
                    # flash('Email Sent Successfully!', success)
                    # print(response.status_code)
                    # print(response.body)
                    # print(response.headers)
                except Exception as e:
                    #if an error occurs flash a msg
                    print("Error")
                    flash("Error while sending mail!", "danger")
                return redirect(url_for('login'))
            else:
                #user exists so flash an error
                flash("User exists!", "danger")
                return render_template('register.html', json=json)
    return render_template('register.html', json=json)

#forgot password route
@app.route('/forgot', methods = ['GET', 'POST'])
@login_required
def forgot_password_page():
    #check if form has been submitted
    if (request.method == 'POST'):
        #get the email entered
        email=request.form.get('email')
        #get the user from the db
        post = Organization.query.filter_by(email=email).first()
        if post:
            #if user exists
            if(post.email==json["admin_email"]):
                #if user tried to reset admin password
                flash("You can't reset password of administrator!", "danger")
                return render_template('forgot-password.html', json=json)
            else:
                #hash the new password generated
                passwordemail = new_password
                post.password = sha256_crypt.hash(new_password)
                db.session.commit()
                #generate the forgot password email to be sent to the user
                subject = "Password Generated : " + passwordemail
                content = '''<!DOCTYPE html><html xmlns="http://www.w3.org/199
9/xhtml" xmlns:v="urn:schemas-microsoft-com:vml" xmlns:o="urn:schemas-
microsoft-com:office:office"><head><title>Reset Your Password</title> <!--
[if !mso]> --><meta http-equiv="X-UA-Compatible" content="IE=edge"> <!--
<![endif]--><meta http-equiv="Content-Type" content="text/html; charset=UTF-
```

```
8"><meta name="viewport" content="width=device-width, initial-
scale=1.0"><style type="text/css">#outlook a{padding:0}.ReadMsgBody{width:100%
}.ExternalClass{width:100%}.ExternalClass *{line-
height:100%}body{margin:0;padding:0;-webkit-text-size-adjust:100%;-ms-text-
size-adjust:100%}table,td{border-collapse:collapse;mso-table-lspace:0pt;mso-
table-rspace:0pt}img{border:0;height:auto;line-height:100%;outline:none;text-
decoration:none;-ms-interpolation-
mode:bicubic}p{display:block;margin:13px 0}</style><style type="text/css">@med
ia only screen and (max-width:480px){@-ms-
viewport{width:320px}@viewport{width:320px}}</style><style type="text/css">@me
dia only screen and (min-width:480px){.mj-column-per-
100{width:100%!important}}</style></head><body style="background: #f0f0f0;"><d
iv class="mj-container" style="background-
color:#f0f0f0;"><table role="presentation" cellpadding="0" cellspacing="0" sty
le="background:#f0f0f0;font-
size:0px;width:100%;" border="0"><tbody><tr><td><div style="margin:0px auto;ma
x-
width:600px;"><table role="presentation" cellpadding="0" cellspacing="0" style
="font-
size:0px;width:100%;" align="center" border="0"><tbody><tr><td style="text-
align:center;vertical-align:top;direction:ltr;font-
size:0px;padding:0px 0px 0px 0px;"><div class="mj-column-per-100 outlook-
group-fix" style="vertical-align:top;display:inline-block;direction:ltr;font-
size:13px;text-
align:left;width:100%;"><table role="presentation" cellpadding="0" cellspacing
="0" width="100%" border="0"><tbody><tr><td style="word-wrap:break-word;font-
size:0px;"><div style="font-size:1px;line-height:30px;white-
space:nowrap;">&#xA0;</div></td></tr></tbody></table></div></td></tr></tbody><
/table></div></td></tr></tbody></table><div style="margin:0px auto;max-
width:600px;background:#FFFFFF;"><table role="presentation" cellpadding="0" ce
llspacing="0" style="font-
size:0px;width:100%;background:#FFFFFF;" align="center" border="0"><tbody><tr>
<td style="text-align:center;vertical-align:top;direction:ltr;font-
size:0px;padding:9px 0px 9px 0px;"><div class="mj-column-per-100 outlook-
group-fix" style="vertical-align:top;display:inline-block;direction:ltr;font-
size:13px;text-
align:left;width:100%;"><table role="presentation" cellpadding="0" cellspacing
="0" width="100%" border="0"><tbody><tr><td style="word-wrap:break-word;font-
size:0px;padding:25px 25px 25px 25px;" align="center"><table role="presentatio
n" cellpadding="0" cellspacing="0" style="border-collapse:collapse;border-
spacing:0px;" align="center" border="0"><tbody><tr><td style="width:204px;"> <
img alt="" title="" height="100px" width="100px" src="https://cdn.discordapp.c
om/attachments/577137963985534994/791571694803353610/favicon.ico" style="borde
r:none;border-radius:0px;display:block;font-size:13px;outline:none;text-
decoration:none;width:100%;height:auto;" width="204"></td></tr></tbody></table
></td></tr><tr><td style="word-wrap:break-word;font-
size:0px;padding:0px 15px 0px 15px;" align="center"><div style="cursor:auto;co
lor:#333333;font-family:Helvetica, sans-serif;font-size:15px;line-
```

```
height:22px;text-align:center;"><h3 style="font-family: Helvetica, sans-
serif; font-size: 24px; color: #333333; line-
height: 50%;">Your password has been reset successfully</h3></div></td></tr><t
r><td style="word-wrap:break-word;font-
size:0px;padding:0px 50px 0px 50px;" align="center"><div style="cursor:auto;co
lor:#333333;font-family:Helvetica, sans-serif;font-size:15px;line-
height:22px;text-
align:center;"><p>Forgot your password or need to change it? No problem.</p></
div></td></tr><tr><td style="word-wrap:break-word;font-
size:0px;padding:20px 25px 20px 25px;padding-top:10px;padding-
left:25px;" align="center"><table role="presentation" cellpadding="0" cellspac
ing="0" style="border-
collapse:separate;" align="center" border="0"><tbody><tr><td style="border:non
e;border-
radius:5px;color:#FFFFFF;cursor:auto;padding:10px 25px;" align="center" valign
="middle" bgcolor="#4DAA50"><a href="#" style="text-
decoration: none; background: #4DAA50; color: #FFFFFF; font-
family: Helvetica, sans-serif; font-size: 19px; font-weight: normal; line-
height: 120%; text-
transform: none; margin: 0px;" target="_blank">Login</a></td></tr></tbody></ta
ble></td></tr><tr><td style="word-wrap:break-word;font-
size:0px;padding:0px 47px 0px 47px;" align="center"><div style="cursor:auto;co
lor:#333333;font-family:Helvetica, sans-serif;font-size:15px;line-
height:22px;text-align:center;"><p><span style="font-
size:14px;"><strong>Questions?&#xA0;</strong><br>Email us at <a href="mailto:e
mail@bulkmailer.cf" style="color: #555555;">email@bulkmailer.cf</a>.&#xA0;</sp
an></p></div></td></tr></tbody></table></div></td></tr></tbody></table></div><
table role="presentation" cellpadding="0" cellspacing="0" style="background:#f
0f0f0;font-
size:0px;width:100%;" border="0"><tbody><tr><td><div style="margin:0px auto;ma
x-
width:600px;"><table role="presentation" cellpadding="0" cellspacing="0" style
="font-
size:0px;width:100%;" align="center" border="0"><tbody><tr><td style="text-
align:center;vertical-align:top;direction:ltr;font-
size:0px;padding:0px 0px 0px 0px;"><div class="mj-column-per-100 outlook-
group-fix" style="vertical-align:top;display:inline-block;direction:ltr;font-
size:13px;text-
align:left;width:100%;"><table role="presentation" cellpadding="0" cellspacing
="0" width="100%" border="0"><tbody><tr><td style="word-wrap:break-word;font-
size:0px;padding:0px 98px 0px 98px;" align="center"><div style="cursor:auto;co
lor:#777777;font-family:Helvetica, sans-serif;font-size:15px;line-
height:22px;text-align:center;"><p><span style="font-
size:12px;"><a href="https://bulkmailer.cf" style="color: #555555;">TERMS OF S
ERVICE</a> | <a href="https://bulkmailer.cf" style="color: #555555;">PRIVACY P
OLICY</a><br>&#xA9; 2020 Bulk Mailer<br><a href="https://bulkmailer.cf/unsubsc
ribe" style="color: #555555;">UNSUBSCRIBE</a></span></p></div></td></tr></tbod
```

```
y></table></div></td></tr></tbody></table></div></td></tr></tbody></table></di
v></body></html>'''
                message = Mail(
                    from_email=('resetpassword@bulkmailer.cf', 'Bulk Mailer Re
set Password'),
                    to_emails=email,
                    subject=subject,
                    html_content=content)
                try:
                    #using the sendgrid api, send the email to the user's emai
l
                    sg = SendGridAPIClient(json['sendgridapi'])
                    response = sg.send(message)
                    flash("You will receive a mail shortly. Password rested su
ccessfully!", "success")
                    # print(response.status_code)
                    # print(response.body)
                    # print(response.headers)
                except Exception as e:
                    #if error occurs flash a msg
                    print("Error!")
        else:
            #user doesn't exist
            flash("We didn't find your account!", "danger")
            return render_template('forgot-password.html', json=json)

    return render_template('forgot-password.html', json=json)

#route to view groups
@app.route('/view/groups')
@login_required
def group_page():
    #get all the groups in the db ordered by id
    groups = Group.query.order_by(Group.id).all()
    return render_template('group_list.html', groups=groups)

#route to add a new group
@app.route('/new/group', methods=['POST'])
@login_required
def submit_new_group():
    #check if form has been submitted
    if(request.method=='POST'):
        #add the group with the entered name to the db and redirect to view gr
oups page
        group_name = request.form.get('groupname')
        entry = Group(name=group_name, date=time)
        db.session.add(entry)
        db.session.commit()
        flash("New group added successfully!", "success")
    return redirect('/view/groups')

#route to delete group with specified id
@app.route("/delete/group/<int:id>", methods = ['GET'])
@login_required
def delete_group(id):
    #get the record of the group to be deleted
    delete_group = Group.query.filter_by(id=id).first()
    if(delete_group.id == 3):
        #if default group flash an error msg
        flash("You can not delete default group!", 'warning')
        return redirect('/view/groups')
    else:
        #otherwise, delete the record and redirect to view groups page
        db.session.delete(delete_group)
        db.session.commit()
        flash("Group deleted successfully!", "danger")
        return redirect('/view/groups')

#route to activate/deactivate a user's account
@app.route("/activate/user/<int:id>", methods = ['GET'])
@login_required
def activate_user(id):
    #get the record of the user with the specified id
    activate_user = Organization.query.filter_by(id=id).first()
    if(activate_user.status == 1):
        #if user's status is active then deactivate account
        activate_user.status = 0
        flash("User deactivated successfully!", "warning")
    else:
        #otherwise, activate account
        activate_user.status = 1
        flash("User activated successfully!", "success")
    db.session.commit()
    #redirect to view users page
    return redirect('/view/users')

#route to delete a user
@app.route("/delete/user/<int:id>", methods = ['GET'])
@login_required
def delete_user(id):
    #get the record of the user with the specified id
    delete_user = Organization.query.filter_by(id=id).first()
    #if user tries to delete admin, flash an error
    if(delete_user.email == json["admin_email"]):
        flash("You cannot delete administrator", "warning")
    #otherwise check if user is admin
    elif current_user.email == json["admin_email"]:
        #delete specified user
        db.session.delete(delete_user)
        db.session.commit()
        flash("User deleted successfully!", "danger")
    else:
        #flash an error msg that only admins can delete users
        flash('Only Admin can delete users!', 'warning')
    #redirect to view users page
    return redirect('/view/users')

#route to delete a template
@app.route("/delete/template/<int:id>", methods = ['GET'])
@login_required
def delete_template(id):
    #get the record of the template with the specified id and delete it
    delete_template = Template.query.filter_by(id=id).first()
    #check if template exists
    if delete_template:
        #delete template
        db.session.delete(delete_template)
        db.session.commit()
        flash("Template deleted successfully!", "danger")
    else:
        #flash an error msg that template doesn't exist
        flash('Template does not exist!', 'danger')
    return redirect('/view/templates')

#route to view subscribers of a particular group
@app.route('/view/subscribers/<int:number>')
@login_required
def subscribers_page(number):
    #get the records of all the subscribers of the group and display
    post = Subscriber.query.filter_by(group_id=number).all()
    response = Group.query.order_by(Group.id).all()
    return render_template('group_members.html', post=post, response=response)

#route to add a new subscriber
@app.route('/new/subscribers', methods=['POST'])
@login_required
def submit_new_subscribers():
    if(request.method=='POST'):
        #using the data entered, create a subscriber in the specified group an
d add to db
        email = request.form.get('email')
        gid = request.form.get('gid')
        entry = Subscriber(email=email, date=time, group_id=gid)
        db.session.add(entry)
        db.session.commit()
        flash("New subscriber added successfully!", "success")
    #redirect to view subscribers of the group page
    return redirect('/view/subscribers/' + str(gid))

#route to delete a subscriber
@app.route('/delete/subscriber/<int:gid>/<int:number>', methods=['GET'])
@login_required
def delete_subscriber(gid, number):
    #get the record of the subscriber to be deleted
    delete_subscriber = Subscriber.query.filter_by(id=number).first()
    #check if subscriber exists
    if delete_subscriber:
        #delete subscriber
        db.session.delete(delete_subscriber)
        db.session.commit()
        flash("Subscriber deleted successfully!", "danger")
    else:
        #flash an error msg
        flash('Subscriber not found!', 'danger')
    #redirect to view subscribers page
    return redirect('/view/subscribers/'+str(gid))

#route to compose and send the bulk email
@app.route('/mail', methods=['POST', 'GET'])
@login_required
def mail_page():
    #check if form has been submitted
    if(request.method=='POST'):
        #get the email fields entered
        username = request.form.get('username')
        name = request.form.get('name')
        subject = request.form.get('subject')
        group = request.form.get('group')
        html_content = request.form.get('editordata')
        html_content = html_content + '''<table role="presentation" cellpaddin
g="0" cellspacing="0" style="background:#f0f0f0;font-
size:0px;width:100%;" border="0"><tbody><tr><td><div style="margin:0px auto;ma
x-
width:600px;"><table role="presentation" cellpadding="0" cellspacing="0" style
="font-
size:0px;width:100%;" align="center" border="0"><tbody><tr><td style="text-
align:center;vertical-align:top;direction:ltr;font-
size:0px;padding:0px 0px 0px 0px;"><div class="mj-column-per-100 outlook-
group-fix" style="vertical-align:top;display:inline-block;direction:ltr;font-
size:13px;text-
align:left;width:100%;"><table role="presentation" cellpadding="0" cellspacing
="0" width="100%" border="0"><tbody><tr><td style="word-wrap:break-word;font-
```

```
size:0px;padding:0px 98px 0px 98px;" align="center"><div style="cursor:auto;co
lor:#777777;font-family:Helvetica, sans-serif;font-size:15px;line-
height:22px;text-align:center;"><p><span style="font-
size:12px;"><a href="https://bulkmailer.cf" style="color: #555555;">TERMS OF S
ERVICE</a> | <a href="https://bulkmailer.cf" style="color: #555555;">PRIVACY P
OLICY</a><br>© 2020 Bulk Mailer<br><a href="https://bulkmailer.cf/unsubscribe"
 style="color: #555555;">UNSUBSCRIBE</a></span></p></div></td></tr></tbody></t
able></div></td></tr></tbody></table></div></td></tr></tbody></table>'''
        #generate the from email
        fromemail = username + domain
        #generate the mail list by extracting the emails of all the subscriber
s in the specified group
        mailobj = Subscriber.query.filter_by(group_id=group).all()
        maillist = []
        for mailobj in mailobj:
            maillist = maillist + [mailobj.email]
        #generate the mail
        message = Mail(
            from_email=(fromemail, name),
            to_emails=maillist,
            subject=subject,
            html_content=html_content)
        try:
            #send the email
            sg = SendGridAPIClient(json['sendgridapi'])
            response = sg.send(message)
            flash("Mail has been sent successfully!", "success")
        except Exception as e:
            #flash an error msg if exception occurs
            flash("Error due to invalid details entered!", "danger")
    #get all the groups and templates in the db to display to the user
    group = Group.query.order_by(Group.id).all()
    mailtemp = Template.query.order_by(Template.id).all()
    return render_template('mail.html', group=group, template=mailtemp)

#route to use a template
@app.route("/use/template/<int:id>", methods = ['GET'])
@login_required
def use_template(id):
    #get the record of the template to be used
    post = Template.query.filter_by(id=id).first()
    #get all groups and templates to be displayed
    group = Group.query.order_by(Group.id).all()
    mailtemp = Template.query.order_by(Template.id).all()
    #redirect to mail with the specified template in the content
    return render_template('mail2.html', group=group, template=mailtemp, post=
post)
```

```
#route to select a group
@app.route("/use/group/<int:id>", methods = ['GET'])
@login_required
def use_group(id):
    #get the record of the group to whom the email has to be sent
    post = Group.query.filter_by(id=id).first()
    #get all the templates to display
    mailtemp = Template.query.order_by(Template.id).all()
    #redirect to mail with the specified group as the recipient
    return render_template('mail3.html', template=mailtemp, post=post)

#route to view all the templates
@app.route('/view/templates')
@login_required
def template_page():
    #get the records of all the templates and display them
    template = Template.query.order_by(Template.id).all()
    return render_template('templates.html', template=template)

#route to add a template
@app.route('/add/template', methods=['POST'])
@login_required
def add_template():
    #if form has been submitted
    if (request.method == 'POST'):
        #get the data in the form fields
        link = request.form.get('link')
        name = request.form.get('name')
        editordata = request.form.get('editordata')
        #use the data to create a record and add it to the db
        entry = Template(name=name, date=time, content=editordata, link=link)
        db.session.add(entry)
        db.session.commit()
        #flash a msg and redirect to view all templates page
        flash('Template added successfully!', 'success')
        return redirect('/view/templates')

#api to subscribe using user's email
@app.route('/subscribe', methods=['GET', 'POST'])
def sub_page():
    #if form has been submitted
    if (request.method == 'POST'):
        #get the user's email
        email = request.form.get('email')
        #check if the subscriber already exists in the db
        check = Subscriber.query.filter_by(email=email).first()
        if not check:
            #if subscriber doesn't exist, create a record (add user to default
 group)
            entry = Subscriber(email=email, date=time, group_id=3)
            #commit to the db
            db.session.add(entry)
            db.session.commit()
            # flash('Newsletter subscribed successfully!', 'success')
            return render_template('thankyou.html')
        else:
            #if user exists then flash an error msg
            flash('You have already subscribed!', 'danger')
            return render_template('error.html')

#api to unsubscribe from a group
@app.route('/unsubscribe', methods=['GET', 'POST'])
def unsub_page():
    #check if form has been submitted
    if (request.method == 'POST'):
        #get the user's email
        email = request.form.get('email')
        #get the subcriber's record from the db
        delete_subscriber = Subscriber.query.filter_by(email=email).first()
        if not delete_subscriber:
            #if subscriber doesn't exist, flash an error msg
            flash('We did not find your data in our database!', 'danger')
            return render_template('error.html')
        else:
            #if subscriber exists, delete from db
            db.session.delete(delete_subscriber)
            db.session.commit()
            flash('Newsletter unsubscribed  successfully!', 'success')
            return render_template('error.html')


#main page
@app.route('/')
@login_required
def dash_page():
    #get the number of groups, subscribers, and templates; display them to the
 user
    glen = len(Group.query.all())
    slen = len(Subscriber.query.all())
    tlen = len(Template.query.all())
    return render_template('index.html', glen=glen, slen=slen, tlen=tlen)

#route to view list of users
@app.route('/view/users')
@login_required
def users_page():
    #get the records of all the users and display to the user
    users = Organization.query.order_by(Organization.id).all()
    return render_template('user_list.html', users=users)

#route to handle page not found
@app.errorhandler(404)
def page_not_found(e):
    # note that we set the 404 status explicitly
    return render_template('404.html'), 404

#execute if file is the main file i.e., file wasn't imported
if __name__ == '__main__':
    app.run(debug=True)
```

CHAPTER 5: TESTING

Test case ID	Test case objective	Test case description	Result
1	Check if login page functions	Valid credentials are entered and the user is logged into his/her account.	Pass
2	Check if the web app is able to send bulk mails	A group of email IDs is selected and mail is sent to all the email IDs successfully.	Pass
3	Check if the user is able to create and add templates	The user provides the html code for the template and the CDN link of the template preview. The template is then stored in the data base and is ready to be used when needed.	Pass
4	Check if the user is able to create groups and add members into the group	The user can create a group by giving it a name along with the email IDs of the subscribers belonging to the group and then, the group can be selected to send mail to all the group members at once.	Pass
5	Check if the user is able to register and create a new account	The user enters valid credentials and is able to create an account.	Pass
6	Check if the all-input fields accept input	All the input fields in the webapp accept the appropriate data and display warning messages if invalid data is entered.	Pass
7	Check if the user is able to logout	The user is able to click on logout and then, he/she is logged out of his/her account.	Pass
8	Check if the webapp is user friendly	The user is able to navigate through the webapp with ease and locate all the features easily.	Pass

9	Check if the user is able subscribe and unsubscribe	The user enters email ID on the client’s page and the user’s mail ID is automatically added into the group. The user is also able to unsubscribe from the client’s page by which, the mail ID is removed from the group.	Pass

CHAPTER 6: SCREENSHOTS

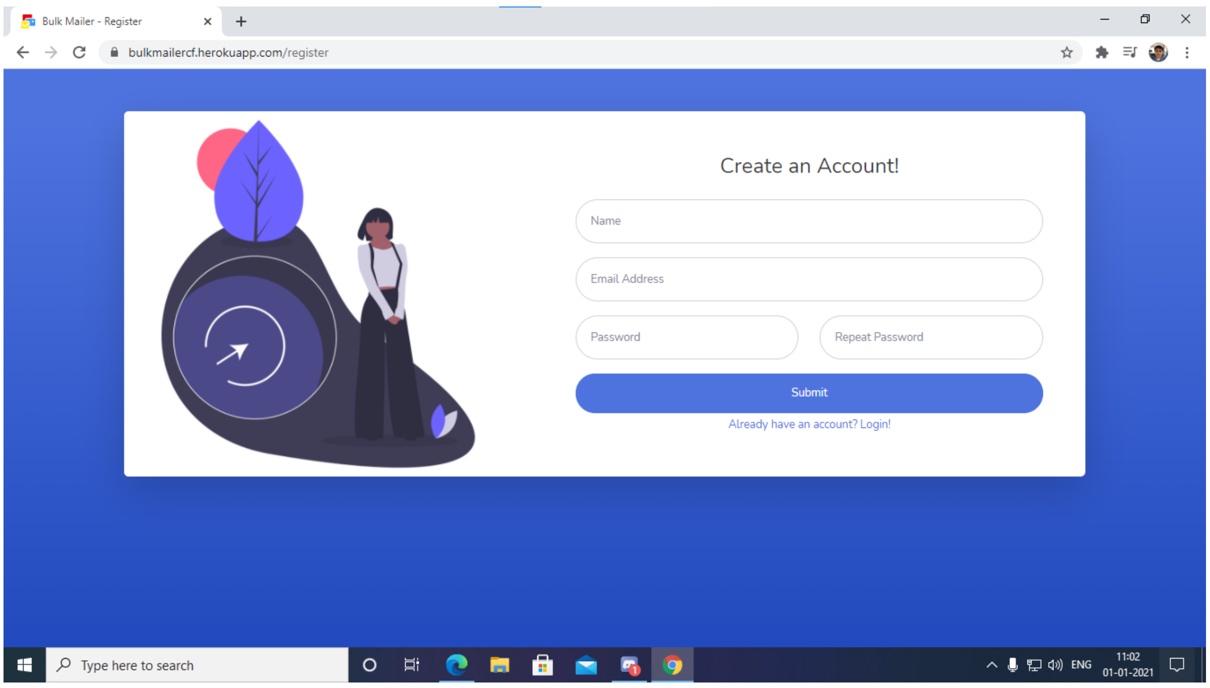


Fig 1: Register Page

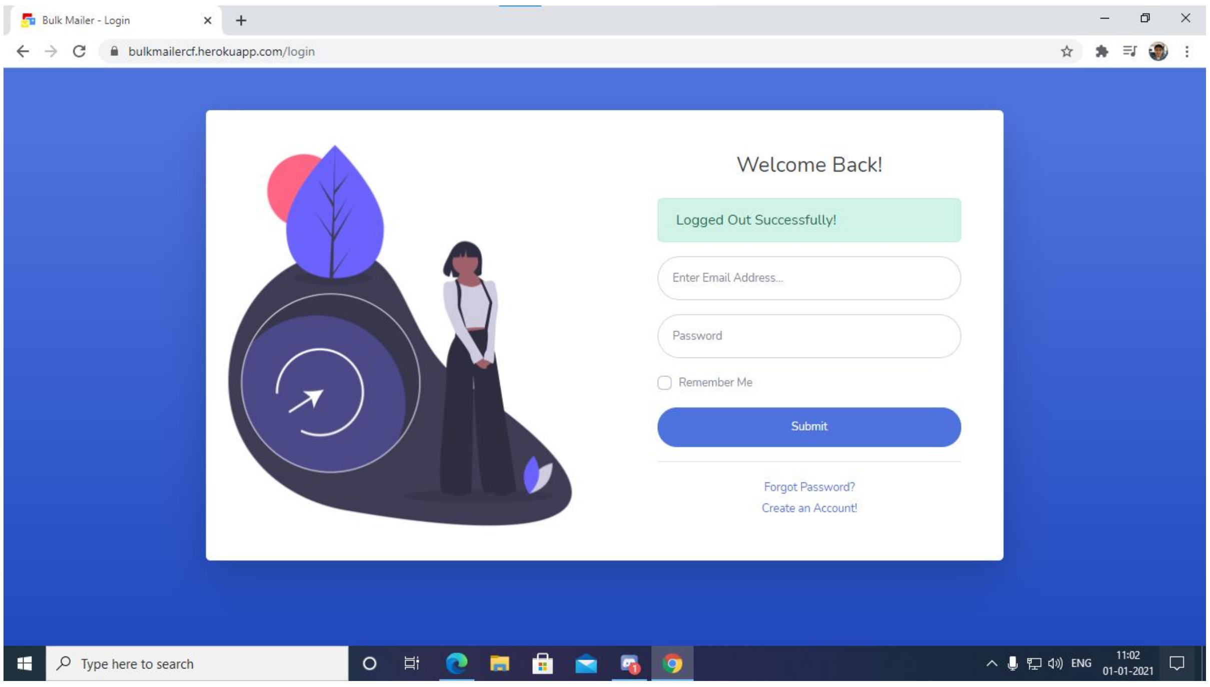


Fig 2: Login Page

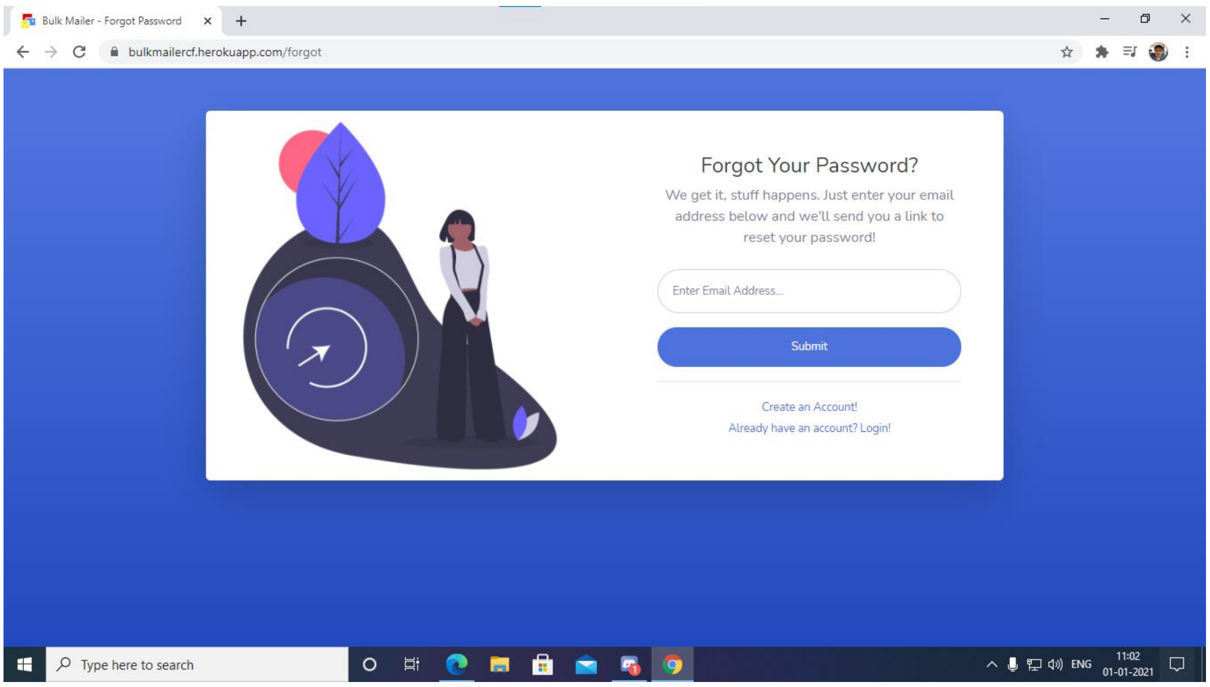

Bulk Mailer - Forgot Password
bulkmailercf.herokuapp.com/forgot
Forgot Your Password?
We get it, stuff happens. Just enter your email address below and we'll send you a link to reset your password!
Enter Email Address...
Submit
Create an Account!
Already have an account? Login!
Type here to search
ENG
11:02
01-01-2021

Fig 3: Forgot Password Page

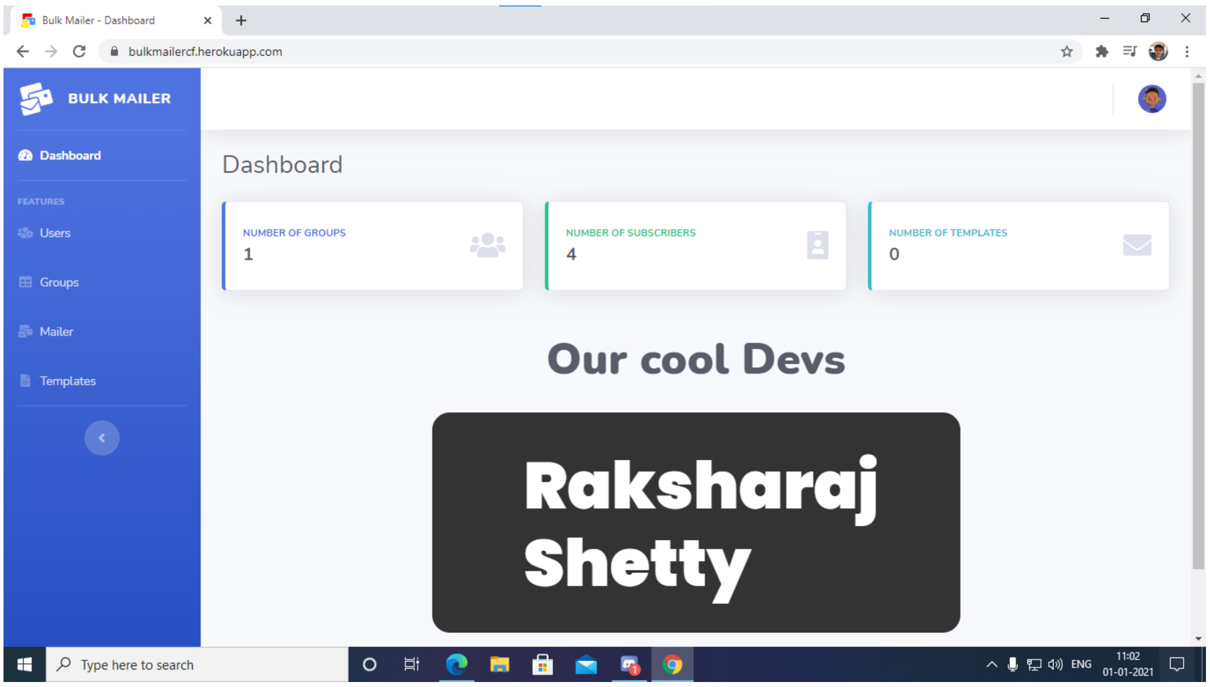
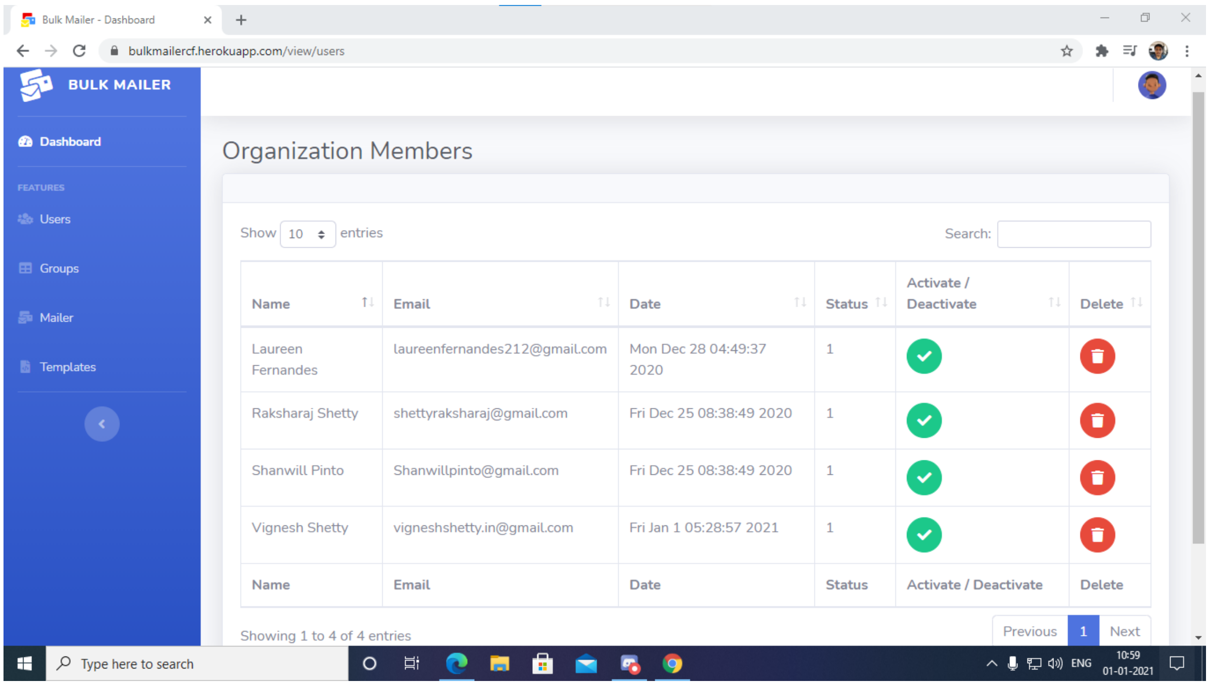


Fig 4: Dash Page



Bulk Mailer - Dashboard

bulkmailercf.herokuapp.com/view/users

BULK MAILER

Dashboard

FEATURES

Users

Groups

Mailer

Templates

Organization Members

Show 10 entries Search:

Name	Email	Date	Status	Activate / Deactivate	Delete
Laureen Fernandes	laureenfernandes212@gmail.com	Mon Dec 28 04:49:37 2020	1	✓	🗑
Raksharaj Shetty	shettyraksharaj@gmail.com	Fri Dec 25 08:38:49 2020	1	✓	🗑
Shanwill Pinto	Shanwillpinto@gmail.com	Fri Dec 25 08:38:49 2020	1	✓	🗑
Vignesh Shetty	vigneshshetty.in@gmail.com	Fri Jan 1 05:28:57 2021	1	✓	🗑
Name	Email	Date	Status	Activate / Deactivate	Delete

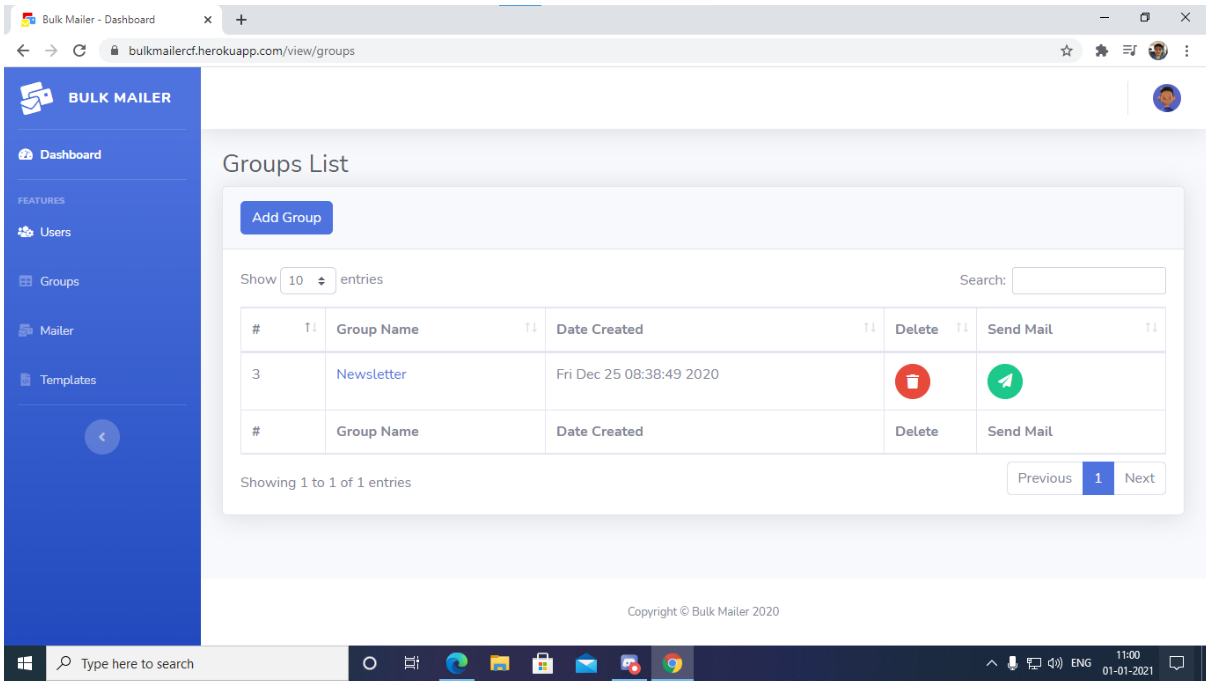
Showing 1 to 4 of 4 entries

Previous 1 Next

Type here to search

ENG 10:59 01-01-2021

Fig 5: View Users Page



Bulk Mailer - Dashboard

bulkmailercf.herokuapp.com/view/groups

BULK MAILER

Dashboard

FEATURES

Users

Groups

Mailer

Templates

Groups List

Add Group

Show 10 entries Search:

#	Group Name	Date Created	Delete	Send Mail
3	Newsletter	Fri Dec 25 08:38:49 2020	🗑	✈
#	Group Name	Date Created	Delete	Send Mail

Showing 1 to 1 of 1 entries

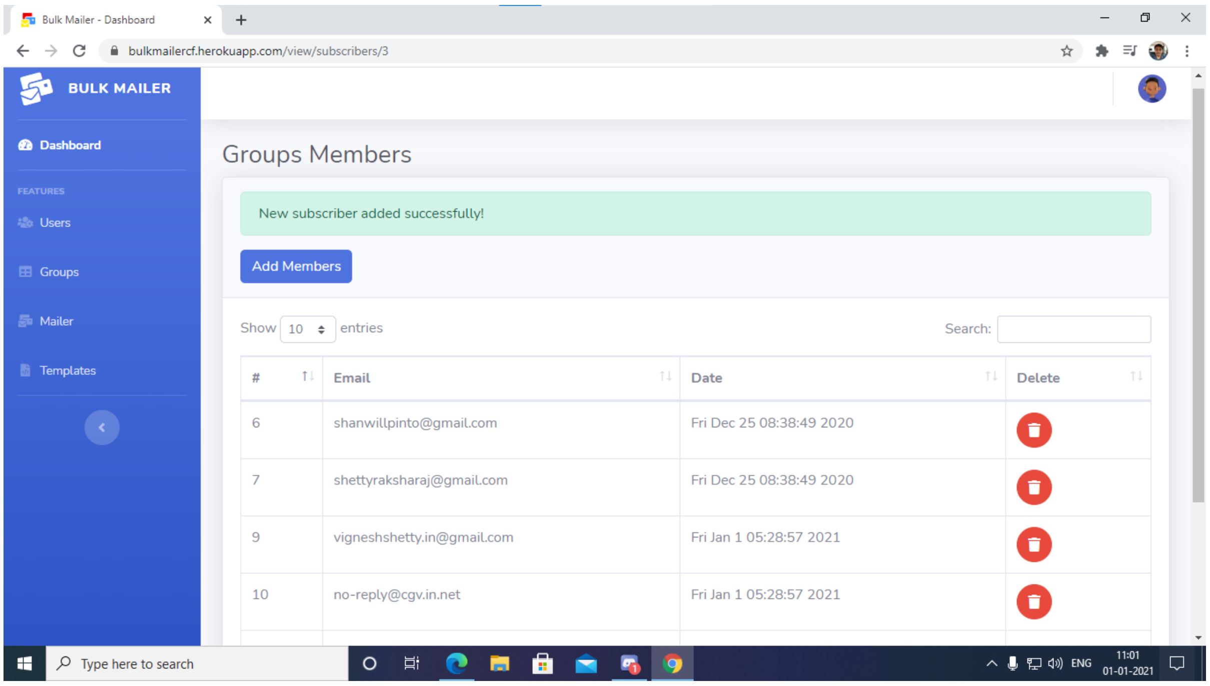
Previous 1 Next

Copyright © Bulk Mailer 2020

Type here to search

ENG 11:00 01-01-2021

Fig 6: View Groups Page



Bulk Mailer - Dashboard

bulkmailercf.herokuapp.com/view/subscribers/3

BULK MAILER

Dashboard

FEATURES

Users

Groups

Mailer

Templates

Groups Members

New subscriber added successfully!

Add Members

Show 10 entries

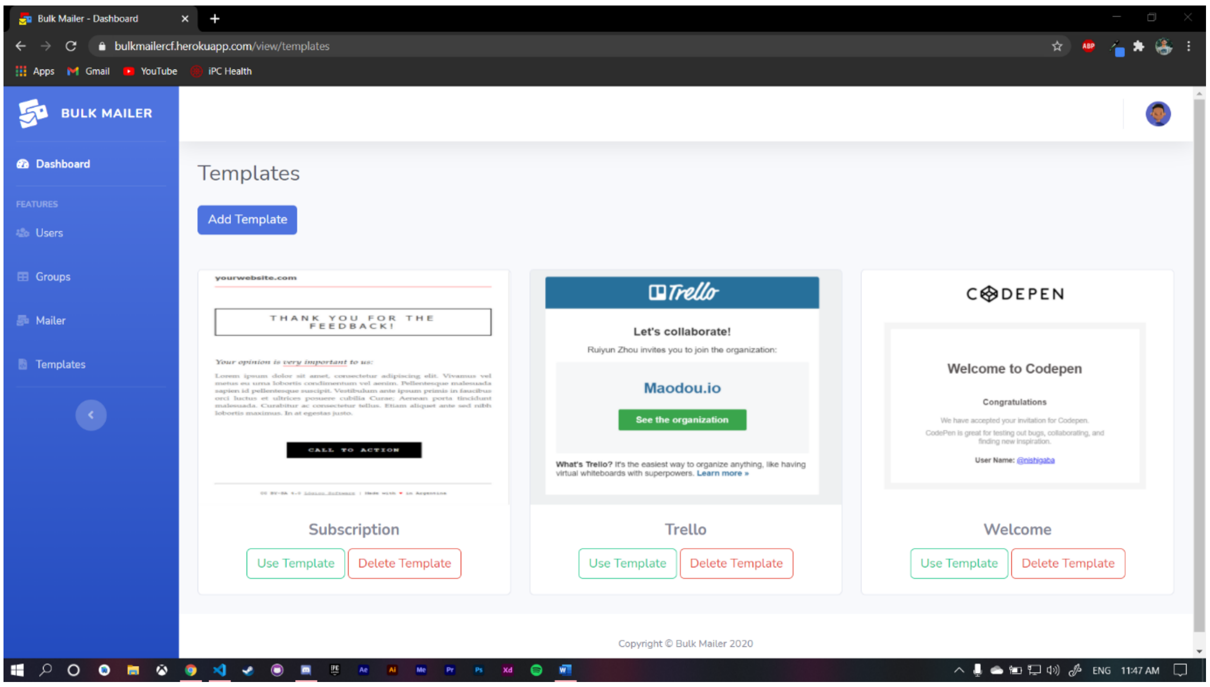
Search:

#	Email	Date	Delete
6	shanwillpinto@gmail.com	Fri Dec 25 08:38:49 2020	
7	shettyraksharaj@gmail.com	Fri Dec 25 08:38:49 2020	
9	vigneshshetty.in@gmail.com	Fri Jan 1 05:28:57 2021	
10	no-reply@cgv.in.net	Fri Jan 1 05:28:57 2021	

Type here to search

ENG 11:01 01-01-2021

Fig 7: View Subscribers Page



Bulk Mailer - Dashboard

bulkmailercf.herokuapp.com/view/templates

Apps Gmail YouTube iPC Health

BULK MAILER

Dashboard

FEATURES

Users

Groups

Mailer

Templates

Templates

Add Template

yourwebsite.com

THANK YOU FOR THE FEEDBACK!

Your opinion is very important to us:

CALL TO ACTION

Subscription

Use Template Delete Template

Trello

Let's collaborate!

Ruiyun Zhou invites you to join the organization:

Maodou.io

See the organization

What's Trello? It's the easiest way to organize anything, like having virtual whiteboards with superpowers. Learn more »

Trello

Use Template Delete Template

CODEPEN

Welcome to Codepen

Congratulations

We have accepted your invitation for Codepen.

CodePen is great for testing out bugs, collaborating, and finding new inspiration.

User Name: @nishigaba

Welcome

Use Template Delete Template

Copyright © Bulk Mailer 2020

ENG 11:47 AM

Fig 8: View Templates Page

CHAPTER 7: CONCLUSION AND FUTURE SCOPE

The main objective of the project is to develop a bulk mailing web application for organisations to send bulk mails to different groups of subscribers.

Bulk mailing has long been a cost-effective and efficient way for businesses to engage with their target market, and with email traffic increasing and many users having to cope with e-mail overload, it makes perfect sense to ensure you’re also focusing on proven, traditional methods.

In future, the mailing service can be made more efficient by adding services like,

- Sentence completion feature.
- Extending this service to more than one organisation.
- Increasing the effectiveness of the application by using private SMTP servers.

CHAPTER 8: REFERENCES

- James F Kurose and Keith W Ross, Computer Networking, A Top-Down Approach, 6th edition, Pearson,2017.
- https://flask.palletsprojects.com/en/1.1.x/
- https://www.sqlalchemy.org/
- https://sendgrid.com/docs/