ACKNOWLEDGEMENT

I wish to express my deep gratitude and sincere thanks to all my teachers for encouragement and the management for providing all facilities to successfully complete the project work.

I extend my sincere thanks to my Principal, Mrs. J Bhuvaneshwari and my Computer Science teacher, Mrs. Anupama whose valuable guidance helped me not only successfully complete the project but also appreciate the beauty of the computer science.

I extend my gratitude to my parents and classmates for their valuable support and time.

INDEX

SYSTEM SOFTWARE AND HARDWARE SPECIFICATIONS

SOFTWARE

The software used to run the program are:

- Python 3.10.2 [IDLE]
- MongoDB cloud database

HARDWARE:

- The hardware used to run the project are:
- Processor - Intel(R) Core (TM) i5-1035G1 CPU @ 1.00GHz 1.20 GHz
- 8GB RAM
- System type - Windows 11 64-bit operating system, x64-based processor
- OS edition - Windows 11 Home Single Language

PROJECT SYNOPSIS

Aim of “Cloud Messaging Service”:

Our project aims to make a cool and easy-to-use chat messaging system. We want to create a platform where people can chat with each other using text and share pictures and stuff in a way that's really smooth. We care a lot about keeping users' messages private and safe, so we're using strong security measures. Our goal is also to let users personalize the system and make it work for them. Overall, we just want to connect people in the classroom for better teaching as it stops the students from disturbing the class.

Introduction:

Ever wondered how a chat system actually works in the background? The amount of thought and hard work that goes behind connecting people successfully, giving them a space to talk to each other and become friends? With that thought in mind, this project was started, to connect people and getting to know how the connections are made possible using the cutting-edge technology Humanity has improved and will keep improving forever and after.

Register:

For every website or app we use, we must start by registering at the respective website/app to keep our data safe and to access our system with an assured mind that nobody else will be misusing your online persona.

Login:

Having a login system makes it easier for the User to use the system from anywhere with the same credentials, making life easier. A login system also helps improve security of the system so that there will no theft of identity leading to problems for the original User.

Chat Window:

The most important part of the Chat Messaging System. It is place where magic happens, where messages can be sent and received in between the users. For Admins, it a place where they can monitor the Backend of the System.

About The Project:

Our application uses simple tools for streamlining complex tasks such as Sending Messages, Logging In, etc. It also helps you understand what goes behind a chat system and how is managed, with the additional benefit of bringing people together. The Chat Messaging System is easy to understand and use.

The application uses MongoDB as the database which is also protected by a password to ensure your data is safe.

Chat Messaging System can be broadly divided into 2 functionalities: -

1. Home Page which allows to:
    a. Register – New Users can generally register themselves here from which they can avail the use of the Chat Window from anywhere.
    b. Login – This allows already registered Users to login to their accounts from any part of the world at any time, giving them access to the Chat Window with an assurance of their security being guaranteed.

c. Exit – Simple option for closing the System after it's satisfied usage.

2. Chat Window allows the user to
   a. To access the Inbox and the Input Bar, the main part of the system.
   b. The Chat Window also grants access to the Control Panel, a System for managing and maintaining the Chat System, which is accessible only to the Admin.

Structure of Chat Messaging System

DESIGN WORK

Modules/Libraries Used

1. subprocess - for installing all the modules
2. Tkinter - to construct basic graphical user interface (GUI) applications
3. Socket - for getting the local ip of the device sending msgs
4. sys - to exit the program if connection to database has failed
5. pymongo - to connect and manage data on MongoDB cloud
6. tabulate - to show the formatted data for the chat, registration and login window
7. datetime - for timestamps for the registration for the chat system and sending messages
8. time - for introducing delays in running and sending data to avoid system stress and lag.
9. threading - to run different processes simultaneously with the main program
10. certifi - to verify and validate SSL / TLS certificate for connecting to MongoDB

Source Code

```
#----------------X---------X--------------------X
                #[MODULES]#
#----------------X---------X--------------------X
import subprocess
done=True
while done:
    try:
        from tkinter import *
        import tkinter as tk
        import tkinter.messagebox
        import socket
        import sys
        import pymongo
        from tabulate import tabulate
        from datetime import datetime
        import time
        import threading
        done=False
        break
    except ModuleNotFoundError:
        modules_to_install = ['tk', 'pymongo','tabulate']
        for module in modules_to_install:
```

```
        subprocess.check_call(['pip', 'install', module])
#----------------X----------X--------------------X
            #[DATABASE]#
#----------------X----------X--------------------X
try:

client=pymongo.MongoClient("mongodb+srv://talkitive:class12proj@
talkitive.0rpcz4p.mongodb.net/")
    db=client['talkitive']
    db["Registration"].create_index("Username", unique=True)
    collection=db["Registration"]
except pymongo.errors.ConnectionFailure:
    tkinter.messagebox.showinfo("Connection Error!","ERROR
CONNECTING TO DATABASE")
    sys.exit(0)
window = tk.Tk()
#----------------X----------X--------------------X
            #[FUNCTIONS]#
#----------------X----------X--------------------X
def destroy():
    window.destroy()


def main():
    global window
    screen_width = window.winfo_screenwidth()
```

```
    screen_height = window.winfo_screenheight()

    window.geometry(f"{screen_width}x{screen_height}")
    window.title("Login and Signup system")
    bg=tk.PhotoImage(file="Images\\reg_login.png")

    l0 = Label(window,image=bg)
    l0.place(relwidth=1, relheight=1)

    label1 = Label(window, text="REGISTER OR
LOGIN!",bg='#00FFFF', font="times 30")
    label1.place(relx = 0.33, rely = 0.3)

    button1 = Button(window, text="Login", width=17,
height=2,bg='#FFFF00',font=("Times New Roman", 14),
command=login)
    button1.place(relx = 0.33, rely = 0.41)

    button2 = Button(window, text="Signup", width=17,
height=2,bg='#FFFF00',font=("Times New Roman", 14),
command=signup)
    button2.place(relx = 0.5, rely = 0.41)

    button3 = Button(window, text="Exit", width=17,
height=2,bg='#FF0000',font=("Times New Roman", 14),
command=destroy)
```

```
    button3.place(relx = 0.41, rely = 0.55)
    window.mainloop()
#----------------X-------X--------------------X
                #[LOGIN]#
#----------------X-------X--------------------X
def login():
    window.withdraw()
    global login_window
    login_window = tk.Toplevel(window)
    login_window.title("Login")
    screen_width = window.winfo_screenwidth()
    screen_height = window.winfo_screenheight()

    login_window.geometry(f"{screen_width}x{screen_height}")

    bg=tk.PhotoImage(file="Images\\login.png")

    l0 = Label(login_window,image=bg)
    l0.place(relwidth=1, relheight=1)

    username_text = StringVar()
    e1 = Entry(login_window,font=("Times New Roman",
20),width=23, textvariable=username_text)
    e1.place(relx=0.68,rely=0.284)
```

```
    password_text = StringVar()
    e2 = Entry(login_window,font=("Times New Roman",
20),width=23, textvariable=password_text, show='*')
    e2.place(relx=0.68,rely=0.47)
    def login_back():
        login_window.destroy()
        window.deiconify()
    def loginn():
        def ip_address():
            try:
                return socket.gethostbyname(socket.gethostname())
            except socket.error:
                return None
        username_text1=username_text.get()
        password_text1=password_text.get()

        mycollection = db["Registration"]
        fields = {"_id": 0, "Username":1, "Password":1}
        all_documents = mycollection.find({}, fields)
        data = [document for document in all_documents]
        cred=[]
        for i in data:
            cred.append(list(i.values()))
```

```
        for i in cred:
            if username_text1 in i and password_text1 in i:
                match=True
                if not
(db["Registration"].find_one({"Username":username_text1})["Block_
status"] and
db["Registration"].find_one({"Local_Ip":ip_address()})["Block_statu
s"]):
                    new_document = {
                    "Serial_no": db["Login"].find_one({}, sort=[("Serial_no",
pymongo.DESCENDING)])["Serial_no"]+1,
                    "Timestamp":datetime.now(),
                    "Local_Ip": ip_address(),
                    "Username":username_text1}
                    db["Login"].insert_one(new_document)


                    login_window.destroy()
                    tkinter.messagebox.showinfo("Logged in!","Login
Successful!")
                    if username_text1=="Admin":
                        adminchat(username_text1)# ADMIN chat screen
                        break
                    else:
                        chat(username_text1)# normal user chat screen
                        break
```

```
            else:
                tkinter.messagebox.showinfo("Banned!","You have been
blocked by the admin, please contact the administrator")
                login_window.destroy()
                break
        else:
            tkinter.messagebox.showinfo("WRONG
CREDENTIALS!!","Incorrect Username/Password!!")


    b = Button(login_window, text="Login",font=("Times New
Roman",20), width=19,bg="purple", command=loginn)
    b.place(relx=0.608,rely=0.73)


    b1 = Button(login_window, text="Back", width=20,bg="red",
command=login_back)
    b1.place(relx=0.66,rely=0.9)
    login_window.mainloop()
#----------------X--------------X--------------------X
                #[REGISTRATION]#
#----------------X--------------X--------------------X
def signup():
    window.withdraw()
    global signup_window
    signup_window = tk.Toplevel(window)
```

```
    screen_width = window.winfo_screenwidth()
    screen_height = window.winfo_screenheight()

    signup_window.geometry(f"{screen_width}x{screen_height}")
    signup_window.title("Sign Up")

    bg=tk.PhotoImage(file="Images\\register.png")

    l0 = Label(signup_window,image=bg)
    l0.place(relwidth=1, relheight=1)

    l4 = Label(signup_window, text="Do you accept useragreement? tick the checkbox to accept==>", font="times 15")
    l4.place(relx=0.4,rely=0.54)

    l5 = Label(signup_window,text="Password should be a minimum of 7 characters, contain special characters, digits, and at least 1 capital letter",bg="red", font="times 12")
    l5.place(relx=0.4,rely=0.49)

    name_text = StringVar()
    e1 = Entry(signup_window,font=("Times New Roman",20),width=15,textvariable=name_text)
    e1.place(relx=0.57,rely=0.15)
```

```
    username_text = StringVar()
    e2 = Entry(signup_window,font=("Times New
Roman",20),width=15, textvariable=username_text)
    e2.place(relx=0.57,rely=0.28)


    password_text = StringVar()
    e3 = Entry(signup_window,font=("Times New
Roman",20),width=15, textvariable=password_text, show='*')
    e3.place(relx=0.57,rely=0.39)


    agree_text = IntVar()
    e4 = Checkbutton(signup_window,text="
",variable=agree_text,font=("Times New Roman",15))
    e4.place(relx=0.8,rely=0.54)


    def signup_back():
        signup_window.destroy()
        window.deiconify()
#----------------X-------------------X---------------------X
                #[LICENSE-AGREEMENT]#
#----------------X-------------------X---------------------X
    def UserAgreement():
        tkinter.messagebox.showinfo("Useragreement","""
       By using this Python program, you agree to the following terms
and conditions. Please read them carefully before proceeding:
```

1. Responsible Use:

You agree to use this program responsibly and for lawful purposes only. You will not engage in any illegal, harmful, or malicious activities using this program.

2. Program Intended Use:

This program is designed to create safe environment. It should be used solely for its intended purpose by obeying the norms of society.

3. User\'s Responsibility:

Any message sent on this chat is the sole responsibility of the user. The owner of the program shall not be held liable for any message sent or recieved.

4. Compliance with Laws:

You agree to comply with all applicable laws, regulations, and legal requirements in your jurisdiction while using this program.

5. Indemnification:

You agree to indemnify and hold harmless the owner of the program from any claims, damages, or liabilities arising out of your use or misuse of the program.

6. Acceptance of Terms:

By using this program, you acknowledge that you have read, understood, and accepted these terms and conditions in their entirety.

```
    If you do not agree with any part of these terms and conditions, you
cannot not proceed with the use of this program.""")


    b2 = Button(signup_window, text="USER-AGREEMENT",
width=20,font=("Times New Roman",15),bg="lightgreen",
command=UserAgreement)
    b2.place(relx=0.5,rely=0.59)
    def signupp():
        def has_special_char_user(name_text):
            special_char = "!@#$%^&*()_+{}:\"<>?|/'[]~` "
            for char in name_text:
                if char in special_char:
                    return True
            return False
        def has_special_char(name_text):
            special_char = "!@#$%^&*()_+{}:\"<>?|/'[]~` "
            for char in name_text:
                if char in special_char:
                    return True
            return False


        def has_digits(name_text):
            for char in name_text:
                if char.isdigit():
```

```
            return True
        return False


    def valid_name(name_text):
            if not has_special_char(name_text) and not
has_digits(name_text) and name_text != "":
                return name_text
            else:
                tkinter.messagebox.showinfo("Invalid Name", "Name
should not contain special characters or digits")


    def user(username_text):
        special_char = "!@#$%^&*()+{}:\"<>?|/'[]~` "
        for char in username_text:
            if char in special_char:
                tkinter.messagebox.showinfo("Invalid Username",
"Username should not contain special characters")
                return None
        username_text = username_text.lower()
        column_name = "Username"


        if not has_special_char_user(username_text) and
len(username_text) > 4:
            if not
db["Registration"].find_one({"Username":username_text}):
                return username_text
```

```
            else:
                tkinter.messagebox.showinfo("Username Taken",
"Username is already taken. Please choose another one.")
                return None
        else:
            tkinter.messagebox.showinfo("Invalid Username",
"Username should be more than 4 characters and can only contain
digits, underscores and alphabets!")


    def Password(password_text):
        passtren = False
        for i in password_text:
            if i.isupper():
                passtren = True


        if has_special_char(password_text) and
has_digits(password_text) and passtren and len(password_text) >= 7:
            return password_text
        else:
            tkinter.messagebox.showinfo("Invalid Password",
"Password should be a minimum of 7 characters, contain special
characters, digits, and at least 1 capital letter")
            return None
    def ip_address():
        try:
            return socket.gethostbyname(socket.gethostname())
```

```
        except socket.error:
            return None

    validated_name = valid_name(name_text.get())
    validated_username = user(username_text.get())
    validated_password = Password(password_text.get())

    if validated_name and validated_username and
validated_password and agree_text.get():
        Error=True
        while Error:
            try:
                new_document = {
                "Serial_no": db["Registration"].find_one({},
sort=[("Serial_no", pymongo.DESCENDING)])["Serial_no"]+1,
                "Timestamp":datetime.now(),
                "Local_Ip": ip_address(),
                "Name": validated_name,
                "Username": validated_username,
                "Password": validated_password,
                "Block_status": False,
                "Useragreement": agree_text.get()
                }
                db["Registration"].insert_one(new_document)
                Error=False
```

```
            except pymongo.errors.DuplicateKeyError:
                Error = True
        signup_window.destroy()
        tkinter.messagebox.showinfo("Registered!","You have successfully signed up. now you can login")
    b1 = Button(signup_window, text="Sign-Up", width=15,font=("Times New Roman",20),bg="yellow", command=signupp)
    b1.place(relx=0.44,rely=0.75)


    b2= Button(signup_window, text="Back", width=15,font=("Times New Roman",20),bg="red", command=signup_back)
    b2.place(relx=0.65,rely=0.75)
    signup_window.mainloop()
#----------------X-------------X--------------------X
            #[CHAT-SCREEN]#
#----------------X-------------X--------------------X
def chat(username_text1):
    global chat_window
    chat_window = tk.Toplevel(window)
    screen_width = window.winfo_screenwidth()
    screen_height = window.winfo_screenheight()


    bg=tk.PhotoImage(file="Images\\chat.png")
```

```
    l0 = Label(chat_window,image=bg)
    l0.place(relwidth=1, relheight=1)

    chat_window.geometry(f"{screen_width}x{screen_height}")
    chat_window.title("Inbox")
    l1 = Label(chat_window, text="INBOX", font="times 20")
    l1.place(relx=0.5,rely=0.05)

    chat_text = tk.Text(chat_window, wrap=tk.WORD,
state=tk.DISABLED)
    chat_text.place(relx=0.3, rely=0.15)

    entry = tk.Entry(chat_window, font=("Times New Roman",20),
width=41)
    entry.place(relx=0.3,rely=0.705)
    def logout():
        chat_window.destroy()
        window.deiconify()
    def ip_address():
        try:
            return socket.gethostbyname(socket.gethostname())
        except socket.error:
            return None
    def send_message():
        message = entry.get()
```

```
        chat_text.config(state=tk.NORMAL)
        if message!="":
            new_document={
                "Serial_no": db["Chat"].find_one({}, sort=[("Serial_no",
pymongo.DESCENDING)])["Serial_no"]+1,
                "Timestamp":datetime.now(),
                "Local_Ip": ip_address(),
                "Sender":username_text1,
                "Message":message}
            db["Chat"].insert_one(new_document)
            time.sleep(1)
        chat_text.config(state=tk.DISABLED)
        entry.delete(0, tk.END)
    def recieve_message(sender,message):
        chat_text.config(state=tk.NORMAL)
        if message!="":
            chat_text.insert(tk.END, f"{sender}: {message}\n")
        chat_text.config(state=tk.DISABLED)
        entry.delete(0, tk.END)



    def check_msg():
        try:
            cursor = mycollection.find({}, fields).sort("Timestamp",
pymongo.DESCENDING).limit(5)
```

```
        msgs=[]
        for document in cursor:
            msgs.append((document["Sender"],document["Message"]))
        time.sleep(1)
        cursor1 = mycollection.find({}, fields).sort("Timestamp",
pymongo.DESCENDING).limit(5)
        msgsnew=[]
        for document in cursor1:

msgsnew.append((document["Sender"],document["Message"]))


        for i in msgs:
            if i in msgsnew:
                msgsnew.remove(i)
        if msgsnew!=[]:
            for i in msgsnew:
                recieve_message(i[0],i[1])
    except:
        pass


    finally:
        cursor.close()
  mycollection = db["Chat"]
  fields = {"_id": 1, "Sender": 1, "Message": 1}
```

```
    result = mycollection.find({}, fields).sort("Timestamp",
pymongo.DESCENDING).limit(5)
    list_result=list(result)

    for i in range(len(list_result)-1,-1,-1):
        recieve_message(list_result[i]['Sender'],list_result[i]['Message'])
    def chat_thread():
        while True:
            check_msg()

    mycollection = db["Chat"]



    send_button = tk.Button(chat_window, text="SEND",font=("Times
New Roman",15),bg="yellow", command=send_message)
    send_button.place(relx=0.748,rely=0.705)

    exit_button=Button(chat_window, text="LOGOUT",font=("New
Times Roman",20),bg="red", command=logout)
    exit_button.place(relx=0.5,rely=0.8)

    chat_update_thread = threading.Thread(target=chat_thread)
    chat_update_thread.daemon = True
    chat_update_thread.start()
```

```
    chat_window.mainloop()
#----------------X-----------------X--------------------X
                #[ADMINCHAT-SCREEN]#
#----------------X-----------------X--------------------X
def adminchat(username_text1):
    global chat_window
    chat_window = tk.Toplevel(window)
    screen_width = window.winfo_screenwidth()
    screen_height = window.winfo_screenheight()

    bg=tk.PhotoImage(file="Images\\chat.png")

    l0 = Label(chat_window,image=bg)
    l0.place(relwidth=1, relheight=1)

    chat_window.geometry(f"{screen_width}x{screen_height}")
    chat_window.title("Inbox")
    l1 = Label(chat_window, text="INBOX", font="times 20")
    l1.place(relx=0.5,rely=0.05)

    chat_text = tk.Text(chat_window, wrap=tk.WORD,
state=tk.DISABLED)
    chat_text.place(relx=0.3, rely=0.15)
```

```
    entry = tk.Entry(chat_window, font=("Times New Roman",20),
width=41)
    entry.place(relx=0.3,rely=0.705)
    def logout():
        chat_window.destroy()
        window.deiconify()
    def ip_address():
        try:
            return socket.gethostbyname(socket.gethostname())
        except socket.error:
            return None
    def send_message():
        message = entry.get()
        chat_text.config(state=tk.NORMAL)
        if message!="":
            new_document={
                "Serial_no": db["Chat"].find_one({}, sort=[("Serial_no",
pymongo.DESCENDING)])["Serial_no"]+1,
                "Timestamp":datetime.now(),
                "Local_Ip": ip_address(),
                "Sender":username_text1,
                "Message":message}
            db["Chat"].insert_one(new_document)
            time.sleep(1)
        chat_text.config(state=tk.DISABLED)
```

```
        entry.delete(0, tk.END)
    def recieve_message(sender,message):
        chat_text.config(state=tk.NORMAL)
        if message!="":
            chat_text.insert(tk.END, f"{sender}: {message}\n")
        chat_text.config(state=tk.DISABLED)
        entry.delete(0, tk.END)



    def check_msg():
        try:
            cursor = mycollection.find({}, fields).sort("Timestamp",
pymongo.DESCENDING).limit(5)
            msgs=[]
            for document in cursor:
                msgs.append((document["Sender"],document["Message"]))
            time.sleep(1)
            cursor1 = mycollection.find({}, fields).sort("Timestamp",
pymongo.DESCENDING).limit(5)
            msgsnew=[]
            for document in cursor1:

msgsnew.append((document["Sender"],document["Message"]))


            for i in msgs:
```

```
            if i in msgsnew:
                msgsnew.remove(i)
        if msgsnew!=[]:
            for i in msgsnew:
                recieve_message(i[0],i[1])
    except:
        pass

    finally:
        cursor.close()
mycollection = db["Chat"]
fields = {"_id": 1, "Sender": 1, "Message": 1}
result = mycollection.find({}, fields).sort("Timestamp",
pymongo.DESCENDING).limit(5)
list_result=list(result)

for i in range(len(list_result)-1,-1,-1):
    recieve_message(list_result[i]['Sender'],list_result[i]['Message'])
def chat_thread():
    while True:
        check_msg()

def control_panel():
    panel_window = tk.Toplevel(chat_window)
```

```
panel_window.title("CONTROL-PANEL")

screen_width = window.winfo_screenwidth()
screen_height = window.winfo_screenheight()
panel_window.geometry(f"{screen_width}x{screen_height}")

bg=tk.PhotoImage(file="Images\\control panel.png")

lbg = Label(panel_window,image=bg)
lbg.place(relwidth=1, relheight=1)

l0 = tk.Label(panel_window,text="CONTROL-
PANEL",font="times 25",bg="red")
l0.place(relx=0.4,rely=0.05)

l1 = tk.Label(panel_window,text="Collection
name",bg="lightgreen",font="times 15")
l1.place(relx=0.18,rely=0.15)

col_name = StringVar()
e1 = Entry(panel_window,font=("New Times
Roman",15),width=22, textvariable=col_name)
e1.place(relx=0.18,rely=0.2)
```

```
l2 =
tk.Label(panel_window,text="Username",bg="lightgreen",font="time
s 15")
l2.place(relx=0.4,rely=0.15)
block_unblock = StringVar()
e2 = Entry(panel_window,font=("New Times
Roman",15),width=20, textvariable=block_unblock)
e2.place(relx=0.4,rely=0.2)


l3 = tk.Label(panel_window,text="Collection
name",bg="lightgreen",font="times 15")
l3.place(relx=0.6,rely=0.15)
delall = StringVar()
e3 =  Entry(panel_window,font=("New Times Roman",15),
width=20,textvariable=delall)
e3.place(relx=0.6,rely=0.2)


l4 = tk.Label(panel_window,text="Collection
name",bg="lightgreen",font="times 15")
l4.place(relx=0.8,rely=0.15)
collec_name = StringVar()
e4 = Entry(panel_window,font=("New Times
Roman",15),width=20,textvariable=collec_name)
e4.place(relx=0.8,rely=0.2)
```

```
    l5 = tk.Label(panel_window,text="Serial
number",bg="lightgreen", font="times 15")
    l5.place(relx=0.8,rely=0.25)
    serial_no = IntVar()
    e5 =  Entry(panel_window,font=("New Times
Roman",15),width=20,textvariable=serial_no)
    e5.place(relx=0.8,rely=0.3)


    def allcol():
        labelcol.config(text="\n".join(db.list_collection_names()))


    labelcol =
tk.Label(panel_window,text="\n".join(db.list_collection_names()),bg
="lightgreen",font="times 15")
    labelcol.place(relx=0.05,rely=0.25)
    allcol_button=tk.Button(panel_window, text="All
Collections",font=("New Times Roman",15),bg="yellow",
command=allcol)
    allcol_button.place(relx=0.05,rely=0.15)


    def exitt():
        panel_window.destroy()


    exit_button=tk.Button(panel_window,
text="CLOSE",font=("New Times ROman",20),width=10,bg="red",
command=exitt)
```

```
    exit_button.place(relx=0.4,rely=0.6)


    def createcol():
        try:
            db.create_collection(col_name.get())
            e1.delete(0, tk.END)
        except:
            pass
    create_col=tk.Button(panel_window, text="Create
collection",font=("New Times Roman",11),bg="yellow",
command=createcol)
    create_col.place(relx=0.18,rely=0.25)


    def delcol():
        try:
            db[col_name.get()].drop()
            e1.delete(0, tk.END)
        except:
            pass


    del_col=tk.Button(panel_window, text="Delete
collection",font=("New Times Roman",11),bg="red",
command=delcol)
    del_col.place(relx=0.28,rely=0.25)
```

```
    def alldata():
        def chatwin():
            CHAT_window = tk.Toplevel(panel_window)
            CHAT_window.geometry("300x350")
            CHAT_window.title("CHAT")
            mycollection = db["Chat"]
            all_documents = mycollection.find()
            data = [document for document in all_documents]
            l1=tk.Label(CHAT_window,text=tabulate(data,
headers="keys", tablefmt="grid"),font="times 10",justify="left")
            l1.pack()

        def regwin():
            REG_window = tk.Toplevel(panel_window)
            REG_window.geometry("300x350")
            REG_window.title("REGISTRATION")
            mycollection = db["Registration"]
            all_documents = mycollection.find()
            data = [document for document in all_documents]
            l2=tk.Label(REG_window,text=tabulate(data,
headers="keys", tablefmt="grid"),font="times 10",justify="left")
            l2.pack()

        def logwin():
            LOGIN_window = tk.Toplevel(panel_window)
```

```
            LOGIN_window.geometry("300x350")
            LOGIN_window.title("LOGIN")
            mycollection = db["Login"]
            all_documents = mycollection.find()
            data = [document for document in all_documents]
            l3=tk.Label(LOGIN_window,text=tabulate(data,
headers="keys", tablefmt="grid"),font="times 10",justify="left")
            l3.pack()
        chatwin()
        regwin()
        logwin()
    all_data=tk.Button(panel_window, text="ALL
DATA",font=("New Times Roman",20),width=10,bg="yellow",
command=alldata)
    all_data.place(relx=0.4,rely=0.5)


    def block():

db["Registration"].update_one({"Username":block_unblock.get()},{"
$set":{"Block_status":True}})
        e2.delete(0, tk.END)
    block=tk.Button(panel_window, text="Block",font=("New Times
Roman",11),bg="red",width=10, command=block)
    block.place(relx=0.5,rely=0.25)


    def unblock():
```

```
db["Registration"].update_one({"Username":block_unblock.get()},{"
$set":{"Block_status":False}})
        e2.delete(0, tk.END)
    unblock=tk.Button(panel_window, text="Unblock",font=("New
Times Roman",11),bg="yellow",width=10, command=unblock)
    unblock.place(relx=0.4,rely=0.25)


    def dele_all():
        mycollection = db[delall.get()]
        filter = {"Serial_no": {"$gt": 0}}
        mycollection.delete_many(filter)
        e3.delete(0,tk.END)
    del_all=tk.Button(panel_window, text="Delete all",font=("New
Times Roman",11),bg="red",width=10, command=dele_all)
    del_all.place(relx=0.65,rely=0.25)


    def del_one():
        db[collec_name.get()].delete_one({"Serial_no":
serial_no.get()})
    del_one_button=tk.Button(panel_window,text="Delete
Record",font=("New Times
Roman",11),bg="red",command=del_one)
    del_one_button.place(relx=0.85,rely=0.35)


    panel_window.mainloop()
```

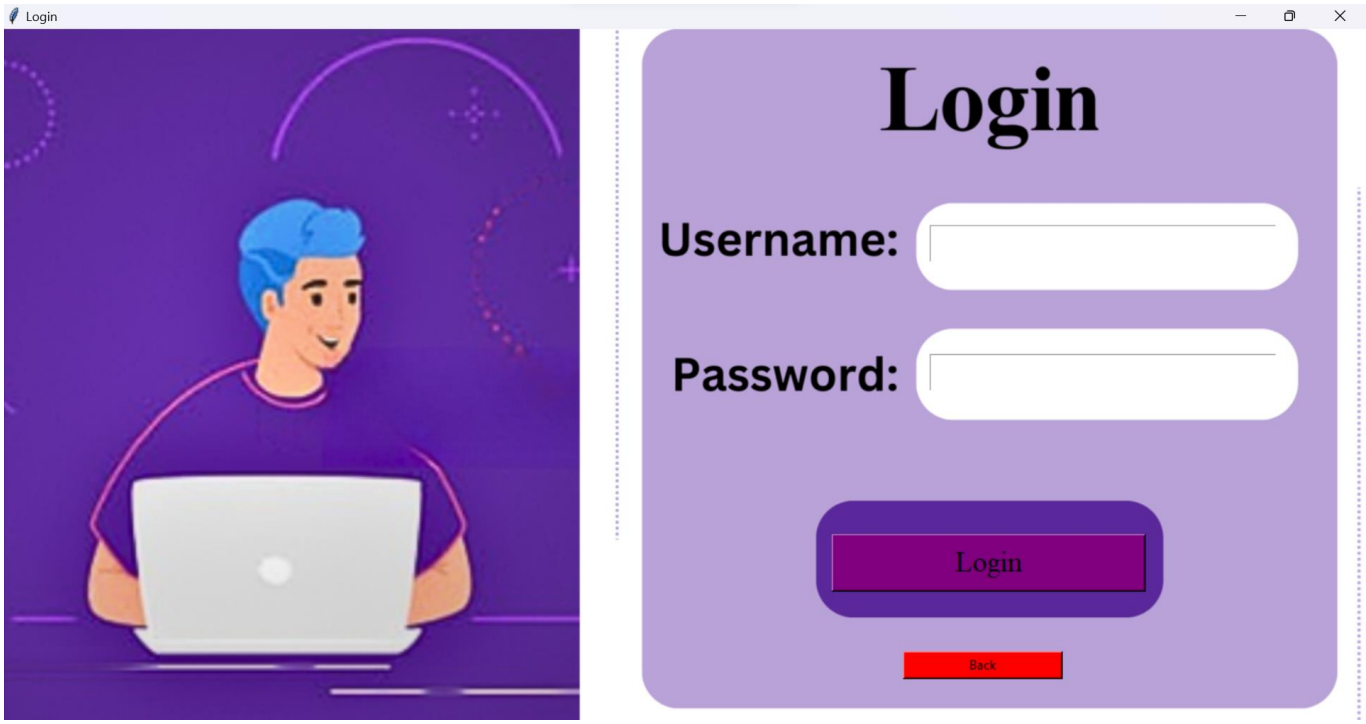
```
    mycollection = db["Chat"]

    control_panel_button = tk.Button(chat_window, text="Control-
Panel",font=("New Times
Roman",20),bg="green",command=control_panel)
    control_panel_button.place(relx=0.1,rely=0.2)

    send_button = tk.Button(chat_window, text="SEND",font=("Times
New Roman",15),bg="yellow", command=send_message)
    send_button.place(relx=0.748,rely=0.705)

    exit_button=Button(chat_window, text="LOGOUT",font=("New
Times Roman",20),bg="red", command=logout)
    exit_button.place(relx=0.5,rely=0.8)

    chat_update_thread = threading.Thread(target=chat_thread)
    chat_update_thread.daemon = True
    chat_update_thread.start()

    chat_window.mainloop()
#----------------X-------------X-----------------X
                #[MAIN _ BODY]#
#----------------X-------------X-----------------X
main()
```

OUTPUT

Homepage:

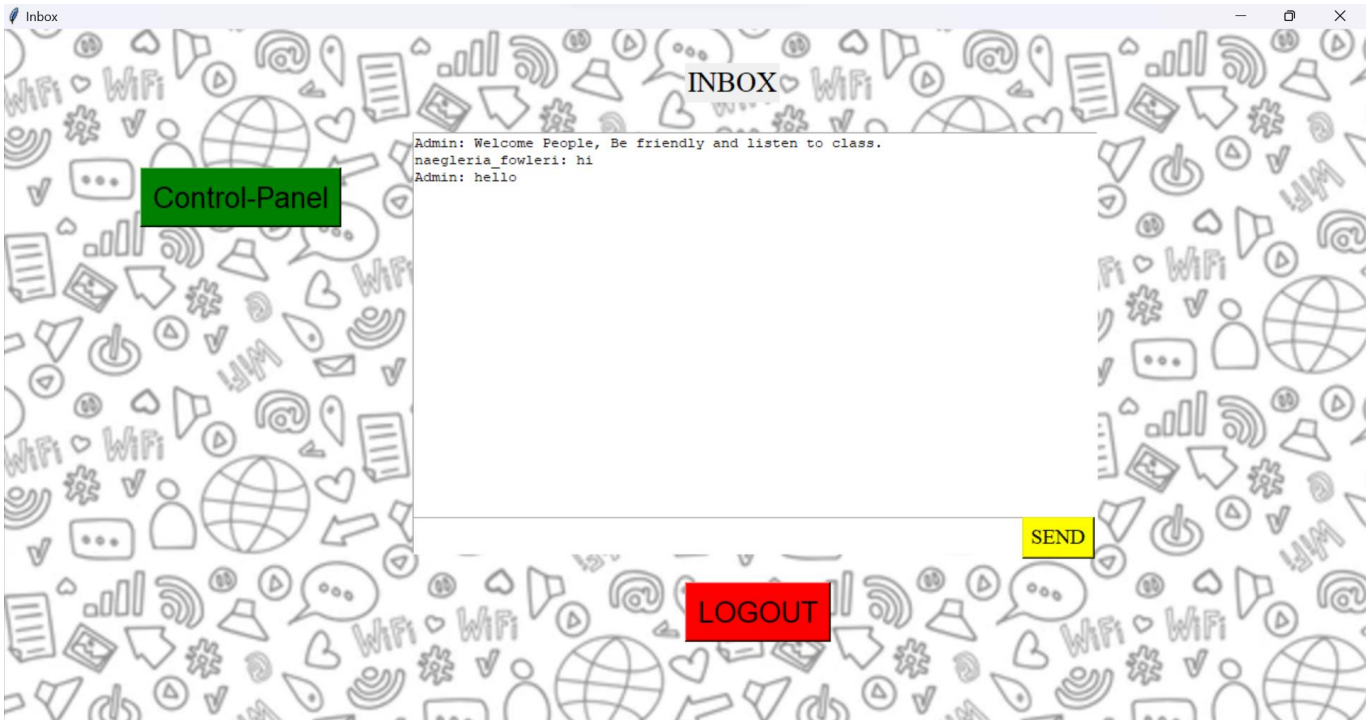
Signup Page:

Login Page:

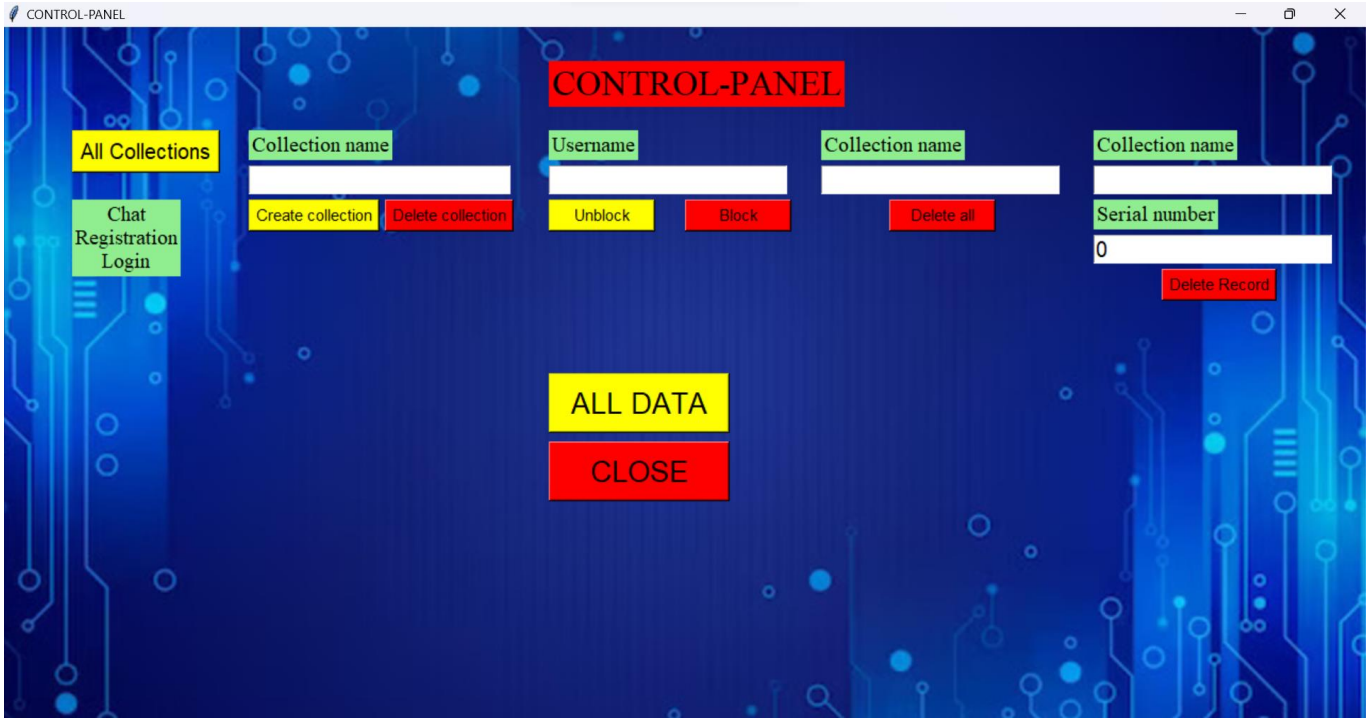


Normal User Inbox:

Admin Inbox:



Control Panel:



FURTHER DEVELOPMENT AREA

- Safety Measures:
  For keeping the chat system safer from any harmful posts, we can improve the system by keeping an algorithm which monitors the inbox for any post that violates the rules and sends a User associated a warning which will increase to a Ban if the act is continued.
  Encryption of data shared and additional protective measures to be applied in storage of User credentials in the cloud database.
- Censorship:
  Keeping up with the first point, we can introduce a Censorship algorithm included in the Input Bar which will help in stopping anyone from posting anything remotely harmful or hurtful.
- Edit Messages:
  We can introduce Users to a program where they are able to fix any of the typos they make, including that other users will know that the message is edited. So Users have a broader control over the data they share and have some time to edit the message they have sent.
- Online User System:
  Plans on introducing a window in the inbox which mentions the users present online.

BIBLIOGRAPHY

1. https://stackoverflow.com
2. https://www.geeksforgeeks.org/python-gui-tkinter/
3. https://pymongo.readthedocs.io/en/stable/
4. https://docs.python.org/3/library/tk.html