National Public School

Computer Science Project

SQL-Chat

A picture containing text, room, gambling house

Description automatically generated

Angad Bhalla

Class 12-A

2022-23

Roll No. 14

Acknowledgement

I, Angad Bhalla, would like to thank our school Principal and Computer science teachers for extending their support that enabled me to successfully complete my project on “SQL-Chat” in the school Computer science lab for the academic year 2022-23.

I would also like to extend my thanks to My Parents and group mates for helping me proof read this project.

Angad Bhalla

**Index**

|  |  |  |
| --- | --- | --- |
| **#** | **Topic** | **Page No.** |
| 1 | Overview of python | 1 |
| 2 | Project Synopsis | 1.5 |
| 3 | System requirements | 2 |
| 4 | Instruction/User Manual | 3 |
| 5 | Modules and functions | 5 |
| 6 | Program code | 6 |
| 7 | Program output | 27 |
| 8 | Drawbacks and limitations | 29 |
| 9 | Bibliography | 30 |

**Overview of python**

Python is a high level, interpreted dynamic programming language that’s gained popularity due to its zen-like syntax. It is the most popular programming language in the world due to its beginner friendly learning curve.

It is a very versatile programming language, Its most common practical applications include:

1. Data science (using NumPy, Matplotlib, tensorflow, pandas etc.)
2. Artificial intelligence/Machine Learning (using Theano)
3. Building web applications (using the Django framework)
4. Building server-side applications (using sockets)

Although with the recent advent of Py-Script and Pynecone, it is undeniable that we can expect to see python to be used more and more to build complex front-end web UI.

Python was initially created by Guido van Rossum working at Centrum Wiskunde & Informatica. At the time, he wanted to create a language which would allow him to communicate with an OS to work on a project called “Amoeba”. He intended for it to be a successor of ABC language.

Over time, python now has extended support for most if not all operating systems and some computers are even built with the explicit purpose to efficiently run python program, an example of this is the raspberry pi

Syntax-wise, python is extremely efficient, as it allows you to call multiple functions in a single line and define multiple functions together too, python syntax less intimidating to learn due to it requiring lesser boilerplate for simpler code when compared to languages like Java and html.

Python is a multi-paradigm programming language as it allow functional programming patterns, you can implement functional programming language patterns using “def” for defining complex multi line functions or “lambda” to define anonymous simple functions. It also allows OOP patterns with it support with classes, objects, inheritance etc.

Python also helps get work done quicker due to its easy to understand ecosystem of third party packages which can be accessed using the pip package manager. Python Public Library packages can simply be pip installed and imported into code on the fly, this makes using third party tools in python far simpler than most other languages.

**System Requirements**

**Minimum Hardware requirement:**

1. For Server/Database:

* 10mbps internet connection (for this project, 4gbps connection was used)
* Any 64 bit processor (for this project, an arm64 processor was used)
* At least 10GB of storage (minimum for any mysql server)
* At least 1gb of ram (for MySQL to run smoothly and manage connections)

1. For Client/GUI:

* A 500MHz processor (minimum for python to run smoothly)
* 128MB of ram (recommended 512MB ram for the application alone)
* 6MB of storage (to store userinfo, project files, and a complete python installation)

**Minimum Software requirements:**

1. For Server/Database:

* MySQL version 8.0.23 (or any version higher installed)
* An open port for MySQL connections (port: 3306 was used here)

1. For Client/GUI:

* Python 3 must be installed
* System must have PyPI packages “mysql-connector-python”, “pillow” and “requests” pre installed

**Important note:**

It is recommending that this code is executed using the dockerfile and virtual environment or the compiled exe that can be found in the git repository of this project, if these are used instead of the source code, most of the requirements can be ignored.

**User Manual/Instructions:**

**Logging in:**

On running the program for the first time, you will be greeted with the login page, where you can enter your login information, i.e username and password, if you do not already have an account, you can register within this page itself. Note that this page is only visible once when the code is ran for the first time, after this, all your login info is stored in the user.info file, and every time the code is ran, it will skip this page unless the login info is outdated or invalid or has been changed.

**Home Page:**

The homepage has many clickables that perform different functions, the gear on the top right is for managing your account/profile, the list of rooms shows rooms of two types: private and global rooms. Right clicking on the private rooms will open the room editor gui and left clicking on any room will open it. Note that you can only see those rooms which you actively have access to. The bottom right features a textbox, and 2 buttons, this can be used to search rooms and access unlisted rooms, i.e, public rooms that exist but don’t show up on the room list. The plus sign can also allow you to access the room creation gui.

**Room Creation:**

To create a room, you must enter an id for that room and also a privacy setting, which is global by default, this id must be unique, there are 3 privacy settings to choose from:

1. Global: Global rooms are visible to everyone and show up in everybody’s room list, anyone can enter these at any time to participate in them
2. Unlisted: Anyone can enter these rooms but they do not show up in anybody’s room list, to enter it, one must enter the room’s unique id on the bottom of the homepage
3. Private: These rooms are only accessible by a small group of people (max. 10), you can decide who has and doesn’t have access to these rooms in the room editor interface. These rooms are the only ones that are editable later on too. They show up as red on the screens of those who can access them

Once this data is filled, simply click on the “create room” button.

**Room Editing:**

This gui is only accessible for private rooms, and only by those users who have access to the private room itself, it allows you to remove and add people to a private room and also change the private room’s settings, once all desired changes have been added, simply press “save”, note that changing the privacy setting of a private room will make the room no longer editable

**Profile editor:**

This gui allows you to change aspects about your profile like your username and password, however for any changes to be implemented you must first enter your current password as a safety measure

**Chat GUI:**

This is a live chat interface that you can use to send and receive messages, messages sent by the same person will show up under their username and a timestamp for the messages is also often given.

**Error management:**

­­­­The application focuses on a simplified user experience and is quite robust, if an error were to occur, the user will be notified via the gui itself as shown below: Graphical user interface, text

Description automatically generated Text

Description automatically generated

**Modules and Functions**

Built in modules used:

1. Tkinter – used for GUI
2. Random – used for token generation for unique identification of users
3. Pickle – used to store user login information in a non human readable format
4. Time – used to send and display timestamps in the gui
5. Threading – used to stop mysql connections from interfering with tkinter’s mainloop

PyPI modules used:

1. MySQL connector – used to access remote database
2. Pillow – used to display images in the gui
3. Requests – used to obtain the ip address of the user using ipify api

**Program Code**

**Login.py-**

import mysql.connector,requests,time,pickle,random,functions,homepage

import tkinter as tk

from PIL import ImageTk, Image

filename='user.info'

def sessionLogin():

    global filename

    try:

        credentials=pickle.load(open(filename,"rb"))

        con=mysql.connector.connect(host="152.67.5.116", user="chat", password="sql123", database="info")

        cur=con.cursor()

        cur.execute("select \* from users where token='"+credentials[0]+"'")

        info=[x for x in cur][0]

        if (credentials[1]==info[1]) and (credentials[2]==info[2]):

            con.close()

            loggedIn(credentials[0],credentials[1],credentials[2])

            return True

        else:

            return False

    except:

        try:

            con.close()

            return False

        except:

            return False

def loginScreen():

    def clearCredentials():

        username.delete(0,tk.END)

        password.delete(0,tk.END)

    def PassError(message):

        global passmessage

        try:

            passmessage.set(message)

        except:

            passmessage=tk.StringVar()

            tk.Label(root, textvariable=passmessage, bg="#36393f", fg="RED", wraplength=password.winfo\_width()).grid(row=5, column=1)

            passmessage.set(message)

    def UNameError(message):

        global errormessage

        try:

            errormessage.set(message)

        except:

            errormessage=tk.StringVar()

            tk.Label(root, textvariable=errormessage, bg="#36393f", fg="RED", wraplength=username.winfo\_width()).grid(row=3, column=1)

            errormessage.set(message)

    def BasicChecks():

        global errormessage,passmessage

        output=True

        if username.get()=="":

            UNameError("Please enter a username")

            output=False

        elif len(username.get())>32:

            UNameError("Username can not be more than 32 charectors long")

            output=False

        elif len(username.get())<3:

            UNameError("Username must be at least 3 charectors long")

            output=False

        elif not functions.isAlphaNumeric(username.get()):

            UNameError("Username must be alphanumeric")

            output=False

        else:

            pass

        if password.get()=="":

            PassError("Please enter a password")

            output=False

        elif len(password.get())<8:

            PassError("Password must be at least 8 charectors long")

            output=False

        elif len(password.get())>64:

            PassError("Password cannot be more than 32 charectors")

            output=False

        return output

    def registerChecks():

        if not BasicChecks():

            return False

        else:

            global con,cur

            con=mysql.connector.connect(host="152.67.5.116", user="chat", password="sql123", database="info")

            cur=con.cursor()

            cur.execute("select \* from users")

            users=[x[1].lower() for x in cur]

            if username.get().lower() in users:

                UNameError("This username is taken")

                return False

            else:

                return True

    def register():

        global con,cur

        if not registerChecks():

            try:

                con.close()

            except:

                pass

        else:

            cur.execute("insert into users values ('"+"".join([random.choice(list("QWERTYUIOPASDFGHJKLZXCVBNM1234567890")) for x in range(32)])+"','"+username.get()+"','"+password.get()+"','"+requests.get('https://api.ipify.org').text+"',"+str(time.time())+","+str(time.time())+",0.0)")

            con.commit()

            con.close()

            tk.Label(root, text=str("registered as "+username.get()+"! now re-enter your credentials to sign-in"), bg="#36393f", fg="WHITE", wraplength=150).grid(row=7, column=0, columnspan=2)

            clearCredentials()

    def loginChecks():

        if not BasicChecks():

            return False

        else:

            global con,cur

            con=mysql.connector.connect(host="152.67.5.116", user="chat", password="sql123", database="info")

            cur=con.cursor()

            cur.execute("select \* from users")

            users=[x[1] for x in cur]

            users2=[x.lower() for x in users]

            if username.get().lower() not in users2:

                UNameError("This username is not registered")

                con.close()

                return False

            elif username.get() not in users:

                UNameEError("Usernames are case-sensitive")

                con.close()

                return False

            else:

                return True

    def login():

        if loginChecks():

            global con,cur

            cur.execute("select \* from users where username='"+username.get()+"'")

            info=[x for x in cur][0]

            if password.get()==info[2]:

                cur.execute("update users set lastjoin = "+str(time.time())+" where token='"+info[0]+"'")

                cur.execute("update users set ip = '"+requests.get('https://api.ipify.org').text+"' where token='"+info[0]+"'")

                con.commit()

                clearCredentials()

                root.destroy()

                loggedIn(info[0], info[1], info[2])

            else:

                PassError("Invalid Credentials, Username or password is wrong")

        else:

            pass

    root=tk.Tk()

    root.title("Login Or Register")

    root.geometry("500x500")

    root.iconbitmap("assets/icon.ico")

    root.config(bg="#36393f")

    ULabel=tk.StringVar()

    ULabel.set("Enter your username here - ")

    logo\_img = ImageTk.PhotoImage(Image.open("assets/icon.png"))

    tk.Label(root, image = logo\_img, bg="#36393f", borderwidth=10).grid(row=0, column=0, columnspan=2)

    text\_img= ImageTk.PhotoImage(Image.open("assets/logintext.png"))

    tk.Label(root, image = text\_img, bg="#36393f", borderwidth=20).grid(row=1, column=0, columnspan=2)

    tk.Label(root, textvariable=ULabel, bg="#36393f", fg="#d7d7d7", font="Terminal").grid(row=2, column=0)

    username=tk.Entry(root, bg="#40444b", fg="#d7d7d7", width=25)

    username.grid(row=2, column=1, sticky=(tk.W,tk.E))

    tk.Label(root, text="Enter your password here - ", bg="#36393f", fg="#d7d7d7", font="Terminal",borderwidth=10).grid(row=4, column=0)

    password=tk.Entry(root, bg="#40444b", fg="#d7d7d7", show="\*")

    password.grid(row=4, column=1, sticky=(tk.W,tk.E))

    tk.Button(root, text="Sign in", font="Calibri 12 bold", bg="#5865f2", fg="WHITE",command=login).grid(row=6,column=0,columnspan=1)

    tk.Button(root, text="Register", bg="#36393f", fg="#d1ddde", font="Calibri 12 bold",command=register).grid(row=6,column=0, sticky=(tk.E),columnspan=1)

    root.mainloop()

def loggedIn(token, username, password):

    pickle.dump([token,username,password],open(filename,"wb"))

    homepage.buildWindow(token)

if not sessionLogin():

    loginScreen()

**#Profile Editor.py**

import tkinter as tk

import mysql.connector, functions,homepage

def buildWindow(token):

    root=tk.Tk()

    grey="#36393f"

    root.title("Profile Editor")

    root.geometry("1000x300")

    root.iconbitmap("assets/icon.ico")

    root.config(bg=grey)

    tk.Label(root,bg=grey,fg='WHITE',text="Edit Your Profile",font="Calibri 32 bold").pack()

    name,pswd,auth,nmerror,passerror,autherror=tk.StringVar(),tk.StringVar(),tk.StringVar(),tk.StringVar(),tk.StringVar(),tk.StringVar()

    name.set(functions.getName(token))

    username=tk.Frame(root,bg=grey)

    username.pack()

    tk.Label(username,bg=grey,fg='WHITE',text="Edit your username - ").pack(side=tk.LEFT)

    tk.Entry(username,bg=grey,fg="WHITE",textvariable=name).pack(side=tk.LEFT)

    tk.Label(username,bg=grey,fg='RED',textvariable=nmerror).pack()

    def BasicChecks():

        output=True

        if name.get() == functions.getName(token) and pswd.get()=="":

            autherror.set("What do you even want to update?")

            output=False

        if name.get()=="":

            nmerror.set("Username cant be empty")

            output=False

        if len(name.get())>32:

            nmerror.set("Username too long (max 32 charectors)")

            output=False

        if len(name.get())<3:

            nmerror.set("Username too short! (min. 3 charectors)")

            output=False

        if not functions.isAlphaNumeric(name.get()):

            nmerror.set("Username must be alphanumeric!")

            output=False

        if len(pswd.get())<8:

            passerror.set("Password must be at least 8 long")

            output=False

        if len(pswd.get())>32:

            passerror.set("Password too long (max 32 charectors)")

            output=False

        if auth.get() in [" "\*x for x in range(32)]:

            autherror.set("Password cannot be blank")

            output=False

        return output

    def AuthCheck():

        if BasicChecks():

            con=mysql.connector.connect(host="152.67.5.116", user="chat", password="sql123", database="info")

            cur=con.cursor()

            cur.execute("select password from users where token='"+token+"'")

            password=[x for x in cur][0][0]

            cur.execute("select name from users")

            names=[x[0] for x in cur]

            con.close()

            if password==auth.get():

                if name.get() in names:

                    nmerror("username is taken")

                    return False

                else:

                    return True

            else:

                autherror.set("Password is incorrect.")

                return False

        else:

            return False

    def Change():

        if AuthCheck():

            if pswd.get()=="":

                con=mysql.connector.connect(host="152.67.5.116", user="chat", password="sql123", database="info")

                cur=con.cursor()

                cur.execute("select password from users where token='"+token+"'")

                password=[x for x in cur][0][0]

            else:

                password=pswd.get()

            username=name.get()

            cur.execute("update users set username='"+username+"', password='"+password+"' where token='"+token+"'")

            con.commit()

            con.close()

            root.destroy()

            homepage.buildWindow(token)

        else:

            pass

    password=tk.Frame(root,bg=grey)

    password.pack()

    tk.Label(password,bg=grey,text="Edit your password - ",fg="WHITE").pack(side=tk.LEFT)

    tk.Entry(password,bg=grey,fg="WHITE",textvariable=pswd,show="\*").pack(side=tk.LEFT)

    tk.Label(password,bg=grey,fg='RED',textvariable=passerror).pack()

    finalising=tk.Frame(root,bg=grey)

    finalising.pack()

    tk.Label(finalising,bg=grey,text="Type your original password (auth) - ",fg="WHITE").pack(side=tk.LEFT)

    tk.Entry(finalising,textvariable=auth,show="\*",fg='WHITE',bg=grey).pack(side=tk.LEFT)

    tk.Label(username,bg=grey,fg='RED',textvariable=autherror).pack()

    actions=tk.Frame(root,bg=grey)

    actions.pack()

    tk.Button(actions,text="Save",fg="WHITE",bg="#5865f2",command= lambda: Change()).pack(side=tk.LEFT)

    tk.Button(actions,text="Delete",fg="WHITE",bg="RED",command= lambda: print("Deleting is currently not enabled yet")).pack(side=tk.LEFT)

    def closing():

        root.destroy()

        homepage.buildWindow(token)

    root.protocol("WM\_DELETE\_WINDOW", lambda: closing())

    root.mainloop()

**room\_editor.py**

import tkinter as tk

import mysql.connector, functions,homepage

def buildWindow(token,room\_id):

    root=tk.Tk()

    grey="#36393f"

    root.title("Private Room Editor")

    root.geometry("1000x300")

    root.iconbitmap("assets/icon.ico")

    root.config(bg=grey)

    tk.Label(root,bg=grey,fg='WHITE',text='Private Room editor for: '+room\_id,font='Calibri 32').pack()

    con=mysql.connector.connect(host="152.67.5.116", user="chat", password="sql123", database="info")

    cur=con.cursor()

    cur.execute("select users from rooms where name='"+room\_id+"'")

    users=[x for x in cur][0][0].split()

    usernames=[]

    for i in range(0,7):

        try:

            usernames.append(functions.getName(users[i]))

        except:

            usernames.append("")

    user1,user2,user3,user4,user5,user6,user7,privacy,error1,error2,error3,error4,error5,error6,error7,username1,username2,username3,username4,username5,username6,username7=tk.Frame(root,bg=grey),tk.Frame(root,bg=grey),tk.Frame(root,bg=grey),tk.Frame(root,bg=grey),tk.Frame(root,bg=grey),tk.Frame(root,bg=grey),tk.Frame(root,bg=grey),tk.IntVar(),tk.StringVar(),tk.StringVar(),tk.StringVar(),tk.StringVar(),tk.StringVar(),tk.StringVar(),tk.StringVar(),tk.StringVar(),tk.StringVar(),tk.StringVar(),tk.StringVar(),tk.StringVar(),tk.StringVar(),tk.StringVar()

    user1.pack(),user2.pack(),user3.pack(),user4.pack(),user5.pack(),user6.pack(),user7.pack()

    tk.Label(user1,fg='WHITE',bg=grey,text="Enter Username of user 1 - ").pack(side=tk.LEFT),tk.Label(user2,fg='WHITE',bg=grey,text="Enter Username of user 2 - ").pack(side=tk.LEFT),tk.Label(user3,fg='WHITE',bg=grey,text="Enter Username of user 3 - ").pack(side=tk.LEFT),tk.Label(user4,fg='WHITE',bg=grey,text="Enter Username of user 4 - ").pack(side=tk.LEFT),tk.Label(user5,fg='WHITE',bg=grey,text="Enter Username of user 5 - ").pack(side=tk.LEFT),tk.Label(user6,fg='WHITE',bg=grey,text="Enter Username of user 6 - ").pack(side=tk.LEFT),tk.Label(user7,fg='WHITE',bg=grey,text="Enter Username of user 7 - ").pack(side=tk.LEFT)

    tk.Entry(user1,bg=grey,fg='WHITE',textvariable=username1).pack(side=tk.RIGHT),tk.Entry(user2,bg=grey,fg='WHITE',textvariable=username2).pack(side=tk.RIGHT),tk.Entry(user3,bg=grey,fg='WHITE',textvariable=username3).pack(side=tk.RIGHT),tk.Entry(user4,bg=grey,fg='WHITE',textvariable=username4).pack(side=tk.RIGHT),tk.Entry(user5,bg=grey,fg='WHITE',textvariable=username5).pack(side=tk.RIGHT),tk.Entry(user6,bg=grey,fg='WHITE',textvariable=username6).pack(side=tk.RIGHT),tk.Entry(user7,bg=grey,fg='WHITE',textvariable=username7).pack(side=tk.RIGHT),tk.Label(user1,bg=grey,fg='RED',textvariable=error1).pack(),tk.Label(user2,bg=grey,fg='RED',textvariable=error2).pack(),tk.Label(user3,bg=grey,fg='RED',textvariable=error3).pack(),tk.Label(user4,bg=grey,fg='RED',textvariable=error4).pack(),tk.Label(user5,bg=grey,fg='RED',textvariable=error5).pack(),tk.Label(user6,bg=grey,fg='RED',textvariable=error6).pack(),tk.Label(user7,bg=grey,fg='RED',textvariable=error7).pack()

    privacy.set(3)

    privacy\_row=tk.Frame(root,bg=grey)

    privacy\_row.pack()

    tk.Label(privacy\_row,bg=grey,fg="WHITE",font="Calibri 12 bold", text="Change Privacy Settings - ").pack(side=tk.LEFT)

    tk.Checkbutton(privacy\_row, text='Global Room',bg=grey,fg='BLUE',variable=privacy, onvalue=1, offvalue=3, command=lambda: print("public")).pack(side=tk.LEFT)

    tk.Checkbutton(privacy\_row, text='Unlisted Room',bg=grey,fg='PURPLE',variable=privacy, onvalue=2, offvalue=3, command=lambda: print("unlisted")).pack(side=tk.LEFT)

    bottom\_row=tk.Frame(root,bg=grey)

    bottom\_row.pack()

    tk.Button(bottom\_row,text="Save",fg='WHITE',bg="#5865f2",command=lambda: Submit()).pack(side=tk.LEFT)

    tk.Button(bottom\_row,text="Delete Room",fg='WHITE',bg="RED",command=lambda: delete()).pack(side=tk.RIGHT)

    username1.set(usernames[0]),username2.set(usernames[1]),username3.set(usernames[2]),username4.set(usernames[3]),username5.set(usernames[4]),username6.set(usernames[5]),username7.set(usernames[6])

    def delete():

        con=mysql.connector.connect(host="152.67.5.116", user="chat", password="sql123", database="info")

        cur=con.cursor()

        cur.execute("delete from rooms where name='"+room\_id+"'")

        con.commit()

        con.close()

        con=mysql.connector.connect(host="152.67.5.116", user="chat", password="sql123", database="rooms")

        cur=con.cursor()

        cur.execute("drop table "+room\_id)

        con.commit()

        con.close()

        root.destroy()

        homepage.buildWindow(token)

    def Submit():

        users=[username1.get(),username2.get(),username3.get(),username4.get(),username5.get(),username6.get(),username7.get()]

        tokens=[]

        spaces=[" "\*x for x in range(0,32)]

        if username1.get() in spaces:

            pass

        else:

            try:

                tokens.append(functions.getToken(username1.get()))

            except:

                error1.set("Invalid Username (not found)")

                return False

        if username2.get() in spaces:

            pass

        else:

            try:

                tokens.append(functions.getToken(username2.get()))

            except:

                error2.set("Invalid Username (not found)")

                return False

        if username3.get() in spaces:

            pass

        else:

            try:

                tokens.append(functions.getToken(username3.get()))

            except:

                error3.set("Invalid Username (not found)")

                return False

        if username4.get() in spaces:

            pass

        else:

            try:

                tokens.append(functions.getToken(username4.get()))

            except:

                error4.set("Invalid Username (not found)")

                return False

        if username5.get() in spaces:

            pass

        else:

            try:

                tokens.append(functions.getToken(username5.get()))

            except:

                error5.set("Invalid Username (not found)")

                return False

        if username6.get() in spaces:

            pass

        else:

            try:

                tokens.append(functions.getToken(username6.get()))

            except:

                error6.set("Invalid Username (not found)")

                return False

        if username7.get() in spaces:

            pass

        else:

            try:

                tokens.append(functions.getToken(username7.get()))

            except:

                error7.set("Invalid Username (not found)")

                return False

        tokens=" ".join(tokens)

        cur.execute("update rooms set privacy="+str(privacy.get())+", users ='"+tokens+"' where name='"+room\_id+"'")

        con.commit()

        root.destroy()

        homepage.buildWindow(token)

    def closing():

        root.destroy()

        homepage.buildWindow(token)

    root.protocol("WM\_DELETE\_WINDOW", lambda: closing())

    root.mainloop()

**homepage.py**

import tkinter as tk

import mysql.connector, chat, functions, createRoom, room\_editor, profileEditor

from PIL import ImageTk, Image

users={}

def buildWindow(token):

    root=tk.Tk()

    root.title("Home Page")

    root.geometry("710x600")

    root.iconbitmap("assets/icon.ico")

    root.columnconfigure(0,weight=1)

    root.rowconfigure(1,weight=1)

    root.config(bg="#36393f")

    def Connect(room\_id):

        if RoomNameCheck(room\_id):

            con=mysql.connector.connect(host="152.67.5.116", user="chat", password="sql123", database="info")

            cur=con.cursor()

            cur.execute("select privacy,users from rooms where name='"+room\_id+"'")

            output=[x for x in cur][0]

            if output[0] in [1,2]:

                root.destroy()

                chat.buildWindow(room\_id,token)

                buildWindow(token)

            else:

                if token in output[1].split():

                    root.destroy()

                    chat.buildWindow(room\_id,token)

                else:

                    RoomNameError("You do not have access to this private room.")

    tk.Label(root,text="Welcome, "+functions.getName(token)+"! Select a room to join - ", bg="#36393f",fg="WHITE", font="Calibri 25 bold").grid(row=0,column=0,sticky=(tk.W))

    def Settings():

        root.destroy()

        profileEditor.buildWindow(token)

    settings\_img=ImageTk.PhotoImage(Image.open("assets/settings.png"))

    settings=tk.Label(root,image=settings\_img,bg="#36393f", cursor='hand2')

    settings.grid(row=0,column=1,sticky=(tk.E))

    settings.bind('<Button-1>', lambda a: Settings())

    con=mysql.connector.connect(host="152.67.5.116", user="chat", password="sql123", database="info")

    cur=con.cursor()

    cur.execute("select \* from rooms where privacy=1")

    publicrooms=[x[0] for x in cur]

    cur.execute("select \* from rooms where privacy=3 and users like '%"+token+"%'")

    privaterooms=[x[0] for x in cur]

    con.close()

    usedSpaces=1

    rooms\_frame=tk.Frame(root,bg="#36393f")

    rooms\_frame.grid(row=usedSpaces, column=0, sticky=(tk.N,tk.W))

    def newRoomDisplay(name):

        nonlocal rooms\_frame,usedSpaces

        label=tk.Label(rooms\_frame,text=name,cursor='hand2',bg="#36393f",fg="#0067d6", font="Calibri 13 underline")

        label.grid(row=usedSpaces, column=0, columnspan=2, sticky=(tk.W,tk.N))

        label.bind('<Button-1>', lambda a:  Connect(name))

    def privRoomDisplay(name):

        nonlocal rooms\_frame,usedSpaces

        label=tk.Label(rooms\_frame,text=name,cursor='hand2',bg="#36393f",fg="RED", font="Calibri 13 underline")

        label.grid(row=usedSpaces, column=0, columnspan=2, sticky=(tk.W,tk.N))

        label.bind('<Button-1>', lambda a: Connect(name))

        label.bind('<Button-2>', lambda a: edit(name))

        label.bind('<Button-3>', lambda a: edit(name))

    for i in publicrooms:

        newRoomDisplay(i)

        usedSpaces+=1

    def edit(room\_id):

        root.destroy()

        room\_editor.buildWindow(token,room\_id)

    for i in privaterooms:

        privRoomDisplay(i)

        usedSpaces+=1

    def RoomNameError(error):

        try:

            global roomError

            roomError.set(error)

        except:

            roomError=tk.StringVar()

            roomError.set(error)

            tk.Label(root, textvariable=roomError, bg="#36393f", fg="RED",wraplength=room\_id.winfo\_width()).grid(row=usedSpaces,column=0, sticky=(tk.E))

    def RoomNameCheck(name):

        output=True

        if name=="" or name in " ".join([" " for x in range(32)]):

            output=False

            RoomNameError("please enter a room id")

        elif not functions.isAlphaNumeric(name):

            output=False

            RoomNameError("room names must be alphanumeric")

        elif len(name)>32:

            output=False

            RoomNameError("room names cannot be more than 32 charectors")

        elif len(name)<3:

            output=False

            RoomNameError("room names cannot be less than 3 charectors")

        elif not RoomExists(name):

            output=False

            RoomNameError("room does not exist, try creating instead?")

        return output

    room\_id=tk.Entry(root, bg="#36393f", fg="WHITE", font="Calibri 13")

    room\_id.grid(row=usedSpaces, column=0, sticky=(tk.E,tk.S))

    connect\_img=ImageTk.PhotoImage(Image.open("assets/connect.png"))

    ConnectTo=tk.Label(root,image=connect\_img,cursor='hand2',bg="#36393f")

    ConnectTo.bind('<Button-1>', lambda a: Connect(room\_id.get()))

    ConnectTo.grid(row=usedSpaces, column=1, sticky=(tk.W,tk.S))

    def builder():

        abc=room\_id.get()

        root.destroy()

        createRoom.buildWindow(token, abc)

    plus\_img=ImageTk.PhotoImage(Image.open("assets/plus.png"))

    addNew=tk.Label(root,image=plus\_img,cursor='hand2',bg="#36393f")

    addNew.bind('<Button-1>', lambda a: builder())

    addNew.grid(row=usedSpaces, column=1, sticky=(tk.E,tk.S))

    tk.Label(root, text="Or enter a custom room ID to connect to/create -",bg="#36393f",fg="WHITE", font="Calibri 13").grid(row=usedSpaces, column=0, sticky=(tk.W,tk.S))

    usedSpaces+=1

    def RoomExists(name):

        con=mysql.connector.connect(host="152.67.5.116", user="chat", password="sql123", database="rooms")

        cur=con.cursor()

        cur.execute("show tables")

        rooms=[x[0].lower() for x in cur]

        con.close()

        if name.lower() in rooms:

            return True

        else:

            return False

    root.mainloop()

**functions.py**

import mysql.connector

con=mysql.connector.connect(host="152.67.5.116", user="chat", password="sql123", database="info")

cur=con.cursor()

def getName(token):

    cur.execute("select username from users where token='"+token+"'")

    output=[x[0] for x in cur]

    return output[0]

def isAlphaNumeric(string):

    output=True

    for i in string.lower():

        if i in list("qwertyuiopasdfghjklzxcvbnm0123456789\_"):

            pass

        else:

            output=False

            break

    return output

def getToken(name):

    cur.execute("select token from users where username='"+name+"'")

    output=[x[0] for x in cur]

    return output[0]

**CreateRoom.py-**

import tkinter as tk

import functions,mysql.connector, homepage

def buildWindow(token,givenID=""):

    grey="#36393f"

    root=tk.Tk()

    root.title("Create A new room")

    root.geometry("500x400")

    root.iconbitmap("assets/icon.ico")

    root.config(bg=grey)

    tk.Label(root,text="Create a new room",bg=grey,fg='WHITE',font="Calibri 30",highlightthickness=30,highlightbackground=grey).pack()

    name=tk.Frame(root, bg=grey)

    name.pack()

    room\_id=tk.Entry(name, bg=grey, fg="WHITE")

    room\_id.pack(side=tk.RIGHT)

    room\_id.insert(0,givenID)

    tk.Label(name, text="Enter a room ID", bg=grey, fg="WHITE",highlightbackground=grey,highlightthickness=25).pack(side=tk.LEFT)

    privacy=tk.Frame(root, bg=grey)

    privacy.pack()

    var=tk.IntVar()

    var.set(1)

    public=tk.Checkbutton(privacy, text='Global Room',bg=grey,fg='BLUE',variable=var, onvalue=1, offvalue=0, command=lambda: print("public"))

    unlisted=tk.Checkbutton(privacy, text='Unlisted Room',bg=grey,fg='PURPLE',variable=var, onvalue=2, offvalue=0, command=lambda: print("unlisted"))

    private1=tk.Checkbutton(privacy, text='Private Room',bg=grey,fg='RED',variable=var,onvalue=3,offvalue=0,command=lambda: print("private"))

    tk.Label(privacy, text="Select a privacy level for your room - ",fg='WHITE',bg=grey).pack(side=tk.LEFT)

    public.pack(side=tk.LEFT)

    unlisted.pack(side=tk.LEFT)

    private1.pack(side=tk.LEFT)

    def BasicChecks():

        if room\_id.get()=="":

            errorMSG.set("Please Enter a valid room ID")

            return False

        elif len(room\_id.get())>32:

            errorMSG.set("Room id too long (max 32 charectors)")

            return False

        elif not functions.isAlphaNumeric(room\_id.get()):

            errorMSG.set("Room ID must be alphanumeric")

            return False

        elif len(room\_id.get())<3:

            errorMSG.set("Room ID too short (min 3 charectors)")

            return False

        elif var.get()==0:

            errorMSG.set("Please select a privacy setting.")

            return False

        else:

            return True

    def AdvancedChecks():

        if not BasicChecks():

            return False

        else:

            print("passed basic checks")

            con=mysql.connector.connect(host="152.67.5.116", user="chat", password="sql123", database="info")

            cur=con.cursor()

            cur.execute("select \* from rooms where name='"+room\_id.get()+"'")

            rooms=[x for x in cur]

            con.close()

            if not rooms==[]:

                errorMSG.set("Room name already exists!")

                return False

            else:

                return True

    def CreateRoom():

        if AdvancedChecks():

            print("passed all checks")

            con=mysql.connector.connect(host="152.67.5.116", user="chat", password="sql123", database="info")

            cur=con.cursor()

            if var.get()==3:

                users="'"+token+"'"

            else:

                users="NULL"

            cur.execute("insert into rooms values('"+room\_id.get()+"',"+str(var.get())+","+users+")")

            con.commit()

            con.close()

            con=mysql.connector.connect(host="152.67.5.116", user="chat", password="sql123", database="rooms")

            cur=con.cursor()

            cur.execute("create table "+room\_id.get()+" (token varchar(32),message mediumtext, time double);")

            con.commit()

            con.close()

            root.destroy()

            homepage.buildWindow(token)

    finish=tk.Frame(root,bg=grey)

    finish.pack(side=tk.BOTTOM)

    errorMSG=tk.StringVar()

    tk.Button(finish,text="Create Room",bg='#5865f2',fg='WHITE',font='Calibri 12 bold',command=lambda: CreateRoom()).pack()

    tk.Label(finish,textvariable=errorMSG,bg=grey,fg="RED").pack(side=tk.BOTTOM)

    root.mainloop()

**Chat.py –**

import tkinter as tk

import time,functions,threading,homepage1

try:

    import mysql.connector

except:

    import sys

    import subprocess

    subprocess.check\_call([sys.executable, '-m', 'pip', 'install','mysql-connector-python'])

    import mysql.connector

updateSpeed=500         #in milliseconds

spaces=[" "\*x for x in range(100)]

users={}

displayed=[(" "," "," ")]

usedSpaces=1

def renderMessage(a):

    a=a.replace(":smile:","☺")

    a=a.replace(":frown:","🙁")

    a=a.replace("%date%",str(time.ctime(time.time())))

    return a

def DisplayMessage(user, message, Time):

    global displayed,usedSpaces

    #print(message)

    displayed.append((user,renderMessage(message),Time))

    if(displayed[-1][0]==displayed[-2][0]):

        message = tk.Label(frame\_widgets, text=renderMessage(message),bg='#36393f', fg="#d1ddde", wraplength=window.winfo\_width(), justify='left', font='Calibri 14')

        message.grid(row=usedSpaces+1, column=0, columnspan=3, sticky=(tk.N,tk.W))

        message.columnconfigure(0,weight=1)

        message.rowconfigure(1,weight=1)

        usedSpaces+=1

    else:

        Name = tk.Label(frame\_widgets, text=user,bg='#36393f', fg='#ffffff', font='Calibri 16 bold')

        Name.grid(row=usedSpaces+1, column=0, columnspan=1, sticky=(tk.S,tk.W))

        Name.columnconfigure(0,weight=1)

        Name.rowconfigure(1,weight=1)

        usedSpaces+=1

        Time\_Widget = tk.Label(frame\_widgets, text=str(time.ctime(Time)),bg='#36393f', fg="#a3a6aa", font='Calibri 10')

        Time\_Widget.grid(row=usedSpaces, column=1, columnspan=3, sticky=(tk.E))

        Time\_Widget.columnconfigure(0,weight=1)

        Time\_Widget.rowconfigure(1,weight=1)

        message = tk.Label(frame\_widgets, text=renderMessage(message), wraplength=window.winfo\_width(),justify='left',bg='#36393f', fg="#d1ddde", font='Calibri 14')

        message.grid(row=usedSpaces+1, column=0, columnspan=3, sticky=(tk.N,tk.W))

        message.columnconfigure(0,weight=1)

        message.rowconfigure(1,weight=1)

        usedSpaces+=1

    frame\_widgets.update\_idletasks()

    canvas.config(scrollregion=canvas.bbox("all"))

def NewlyAdded():

    global LastChecked, room

    con=mysql.connector.connect(host="152.67.5.116", user="chat", password="sql123", database="rooms")

    cur=con.cursor()

    command="select \* from "+room+" where time>="+str(LastChecked)

    LastChecked=time.time()

    cur.execute(command)

    output=[]

    for x in cur:

        output.append(x)

    con.close()

    return output

def sendMessage(token):

    global spaces, room, textbox

    con=mysql.connector.connect(host="152.67.5.116", user="chat", password="sql123", database="rooms")

    cur=con.cursor()

    if textbox.get() in spaces:

        pass

    else:

        command="insert into "+room+" values ('"+token+"','"+str('"'.join(str(textbox.get()).split("'")))+"',"+str(time.time())+");"

        cur.execute(command)

        con.commit()

        con.close()

        textbox.delete(0, tk.END)

def update():

    try:

        global updateSpeed, empty

        def newMsgs():

            a=NewlyAdded()

            if a==[]:

                pass

            else:

                for i in a:

                    users[i[0]]=functions.getName(i[0])

                    DisplayMessage(users[i[0]],i[1],i[2])

        canvas.config(width=window.winfo\_width()-vsb.winfo\_width() ,height=window.winfo\_height()-100)

        threading.Thread(target=newMsgs).start()

        window.after(updateSpeed,update)

    except:

        pass

def buildWindow(ConnectTo, token):

    global textbox,empty,window,LastChecked,room,spaces,frame\_widgets,canvas,vsb,frame\_canvas

    LastChecked=0.0

    if ConnectTo in spaces:

        room="global"

    else:

        room=ConnectTo

    window=tk.Tk()

    window.title("ChatRoom")

    window.geometry("710x600")

    window.iconbitmap("assets/icon.ico")

    window.columnconfigure(0,weight=1)

    window.rowconfigure(1,weight=1)

    window.config(bg="#36393f")

    UNameLabel=tk.Label(window, text="Room ID: "+ConnectTo,bg='#36393f', fg= "#ffffff", font='System 20')

    UNameLabel.grid(row=0, column=0, sticky=(tk.N,tk.W))

    frame\_canvas=tk.Frame(window, width=window.winfo\_width(), bg="#36393f")

    frame\_canvas.grid(row=1, column=0, columnspan=3,sticky=(tk.W,tk.E,tk.S))

    canvas = tk.Canvas(frame\_canvas, bg="#36393f", highlightthickness=0)

    canvas.grid(row=0, column=0, sticky="news")

    vsb = tk.Scrollbar(frame\_canvas, orient="vertical", command=canvas.yview)

    vsb.grid(row=0, column=1, sticky='ns')

    canvas.configure(yscrollcommand=vsb.set)

    frame\_widgets = tk.Frame(canvas, bg="#36393f", borderwidth=0)

    canvas.create\_window((0, 0), window=frame\_widgets, anchor='nw')

    frame\_widgets.update\_idletasks()

    canvas.config(scrollregion=canvas.bbox("all"))

    send = tk.Button(window, text="Send", font="Sans 12 bold", bg="#5865f2",fg="WHITE",command=lambda e: sendMessage(token))

    send.grid(row=2, column=2, sticky=(tk.E))

    window.bind('<Return>',lambda e:sendMessage(token))

    textbox = tk.Entry(window, bg="#40444b", fg="#d7d7d7", font="Calibri 17")

    textbox.grid(row=2, column=0, columnspan=2, sticky=(tk.E,tk.W))

    textbox.columnconfigure(0,weight=1)

    def closing(token=token,window=window):

        window.destroy()

        homepage1.buildWindow(token)

    window.protocol("WM\_DELETE\_WINDOW", closing)

    update()

    window.mainloop()

**Program Output**

**Login Screen:**

**Graphical user interface, application, Teams

Description automatically generated**

**Home Page:**

**Graphical user interface, text

Description automatically generated**

**Room Creation GUI:**

**Graphical user interface, application

Description automatically generated**

**Profile Editor:**

**Graphical user interface, website

Description automatically generated**

**Chat Room Interface:**

**Graphical user interface, text

Description automatically generated**

**Room Editor:**

**Graphical user interface

Description automatically generated**

**Limitations:**

1. In the current release of the code, only one user can be logged in at a time per installation of the client code, this is because a formal user management system has not been setup in the current version, instead, all the data is stored in a pickled binary file named “user.info”, in order to log out of one’s account, one must delete this file

**Possible fix:** Implement a user management system within the gui

Graphical user interface, text, application, email

Description automatically generated

1. On decompiling the exe or going through the source code one can find the mysql user login information and mis-use of this may give a person unfiltered access to the app’s database.

**Possible Fix:** Using sockets to manage requests for sending and receiving data packages, and adding authentication information in all dta packages that are transferred

**Bibliography**

1. History of python - <https://www.artima.com/articles/the-making-of-python>
2. Python version documentation - <https://www.python.org/doc/versions/>
3. Python 3 info - <https://www.python.org/download/releases/3.0/>
4. Tkinter documentation - https://docs.python.org/3/library/tk.html
5. MySQL documentation - https://dev.mysql.com/doc/