



LOVELY
PROFESSIONAL
UNIVERSITY

MYCLASS PLATFORM AUTOMATION

As a project for course

PYTHON PROGRAMMING(INT213)

Name: Yaswanth Bolisetty

Registration Number: 12009486

Name: Kumar Ashish

Registration Number:12010912

Program: B. Tech Computer Science
Engineering (CSE)

Semester: Third

Name of the university: Lovely Professional
University

Date of Submission:20th November 2021

MYCLASS PLATFORM AUTOMATION

ABSTRACT:

Students like us need to attend and listen to classes regularly because this allows us to learn new skills, improve them every day and build up our foundation for learning new ones. Even though missing classes is not good, there are times when we must take time away from classes due to a busy schedule or an emergency.

In that case, this myclass platform automation project automatically attends your class for you and ensures that you do not lose your attendance even in emergencies.

ACKNOWLEDGEMENT:

It has been a pleasure working on this project under the guidance of Prof. Sagar Pande.

I must also thank my parents and friends for their immense support and help during this project. Without their help, completing this project would have been very difficult.

Contents of Report

S.No.	Title	Page No.
1.	Abstract	2
2.	Introduction: 1. Context 2. Motivations 3. Idea	4
3.	Team Members: 1. Team Leader 2. Team members 3. Contributions	5
4.	Libraries Used and why they are used	6-7
5.	Screenshots: 1. Installation process 2. Execution of code 3. Automation in process	8-17
6.	Explanation of Code	17-24
7.	References	26

Introduction

Context:

Under the supervision of Sagar Pande, I have done this project as part of my Computer Science Engineering (CSE) course at Lovely Professional University. I have two months to complete the requirements to pass the module.

Motivations:

Because we were interested in automation, we chose a project that included automation. Then we began looking for project ideas involving automation, but we couldn't decide. Then one day, I received a note from one of my juniors, who was concerned about his low attendance. He was unable to attend his classes because he was relocating to a different state. This gave me an idea: if I automate these online classes, they will be useful in some time-sensitive scenarios.

Idea:

We wanted to create a unique and exciting project that included Python automation.

The idea was simple: once the code is executed, the myclass platform will automatically attend the classes. The code is provided with timetable. It checks the system time, joins the session according to the timetable provided, and leaves when the teacher ends the session.

Team Members:

Team Leader:

Yaswanth Bolisetty

Contributions:

- 1.Selenium, tkinter, pip
- 2.WebDriver related classes
- 3.Login function
- 4.Discordwebhook(dhooks)
- 5.Report

Team Member:

Kumar Ashish:

Contributions:

- 1.Selenium, tkinter
- 2.Date and Time related classes
3. Apscheduler
- 4.Join Function
- 5.Report

Libraries Used and why they are used:

Selenium:

Selenium is an open-source web-based automation tool. Python language is used with Selenium for testing. It has far less verbose and easy to use than any other programming language. ... Selenium can send the standard Python commands to different browsers, despite variation in their browser's design.

Time:

The Python time module provides many ways of representing time in code, such as objects, numbers, and strings. It also provides functionality other than representing time, like waiting during code execution and measuring the efficiency of your code.

Calendar:

Python defines an inbuilt module calendar that handles operations related to the calendar.

The calendar module allows output calendars like the program and provides additional useful functions related to the calendar. Functions and classes defined in the Calendar module use an idealized calendar, the current Gregorian calendar extended in in both directions. By default, these calendars have Monday as the first day of the week, and Sunday as the last (the European convention).

Datetime:

In Python, date and time are not a data type of their own, but a module named datetime can be imported to work with the date as well as time. Python Datetime module comes built into Python, so there is no need to install it externally.

Tkinter:

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

APScheduler:

Advanced Python Scheduler (APScheduler) is a Python library that lets you schedule your Python code to be executed later, either just once or periodically. ... That said, APScheduler does provide some building blocks for you to build a scheduler service or to run a dedicated scheduler process.

Discord:

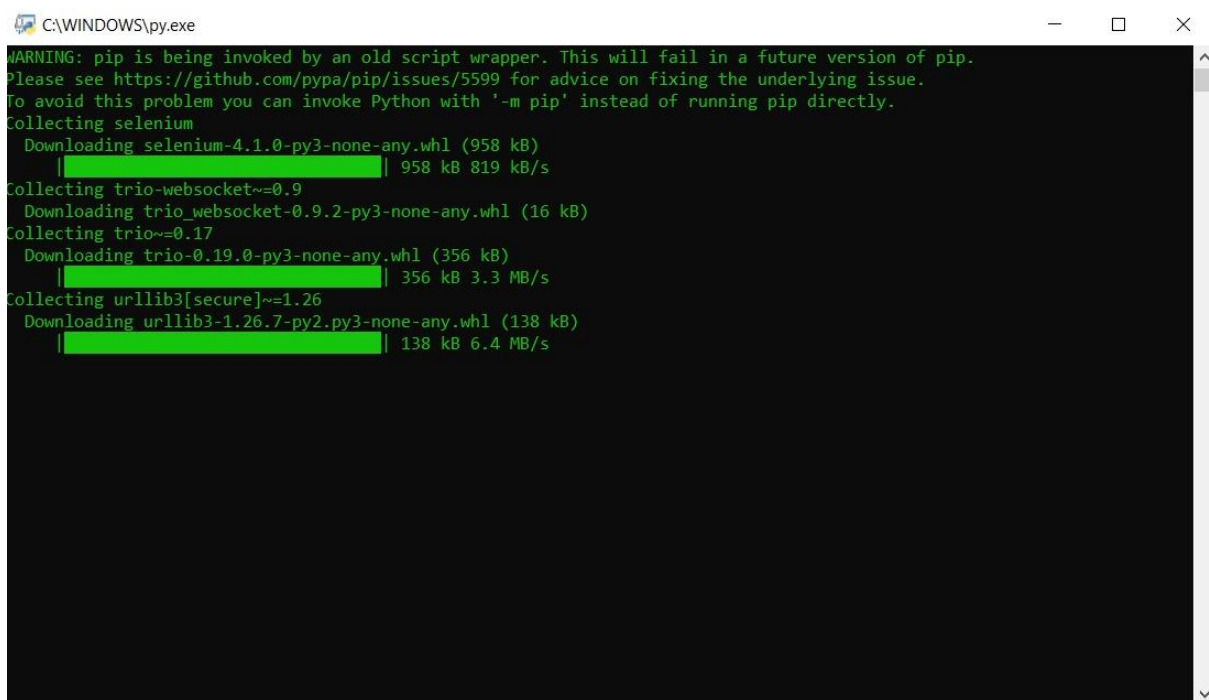
Discord module is made to make bots very simple. You can send messages, create channels and everything else you need. If you need something we doesn't have, create an issue, and let us know.

Screenshots:

Installation Process:

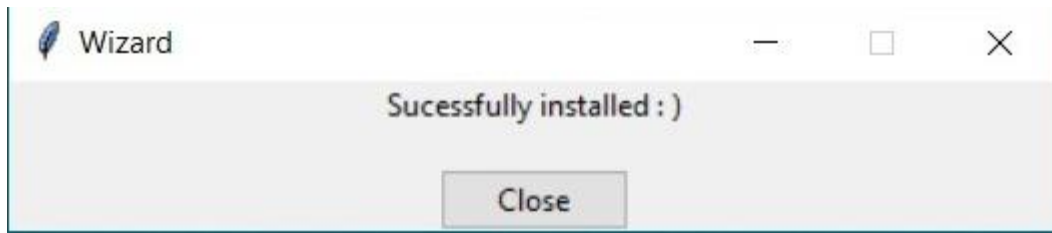
In this phase we install the requirements:

1. For installing modules, open the check.py file then it automatically downloads all the necessary packages will be installed



```
C:\WINDOWS\py.exe
WARNING: pip is being invoked by an old script wrapper. This will fail in a future version of pip.
Please see https://github.com/pypa/pip/issues/5599 for advice on fixing the underlying issue.
To avoid this problem you can invoke Python with '-m pip' instead of running pip directly.
Collecting selenium
  Downloading selenium-4.1.0-py3-none-any.whl (958 kB)
    | 958 kB 819 kB/s
Collecting trio-websocket~=0.9
  Downloading trio_websocket-0.9.2-py3-none-any.whl (16 kB)
Collecting trio~=0.17
  Downloading trio-0.19.0-py3-none-any.whl (356 kB)
    | 356 kB 3.3 MB/s
Collecting urllib3[secure]~=1.26
  Downloading urllib3-1.26.7-py2.py3-none-any.whl (138 kB)
    | 138 kB 6.4 MB/s
```

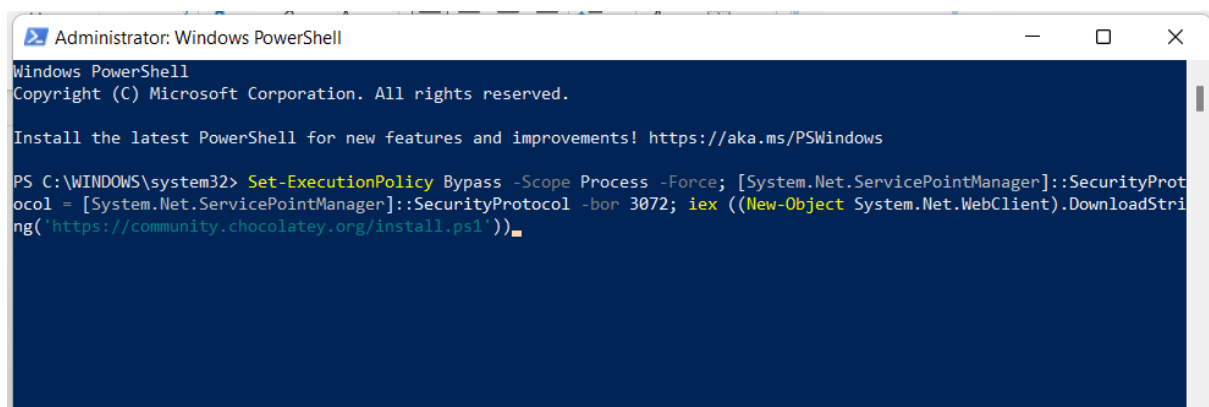
After it is done it will show a pop up like this this means u don't have any errors in installing:



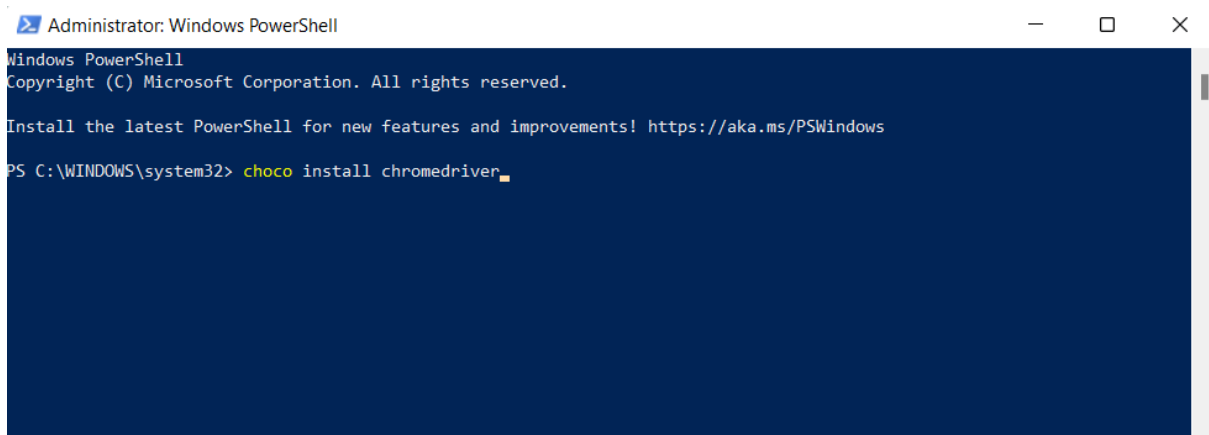
2.Installing chrome web driver:

Open your Windows Power Shell as Admin and paste this command {{ Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.ServicePointManager]::SecurityProtocol = [System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1')) }}

and press enter...



then chocolatey will be installed in your system now type “choco install chromedriver” in the same Windows Power Shell window which is running as admin

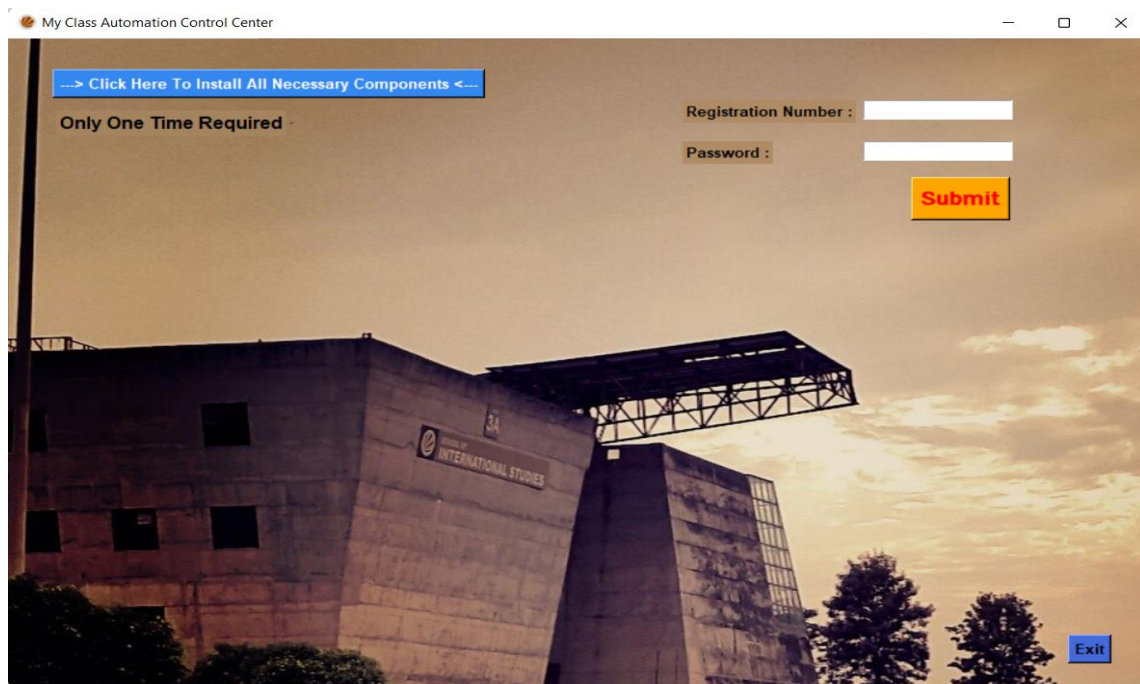


Then chrome webdriver will be installed

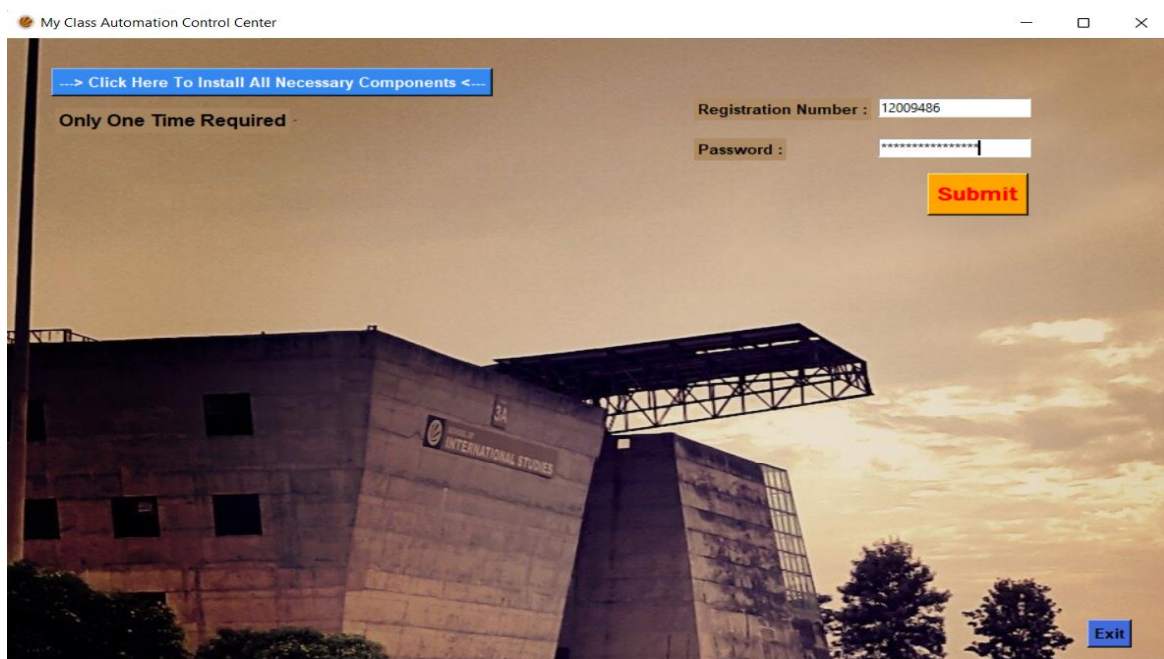
Now you are ready to use the application 😊

Code Execution :

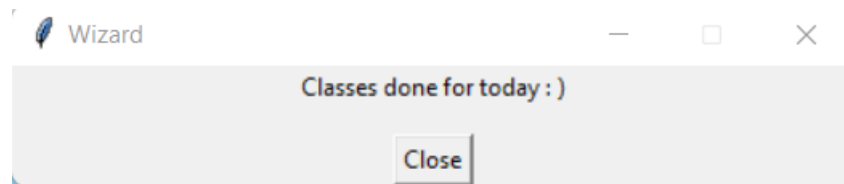
1. When u run the code this GUI window opens:



2. Enter your credentials in respective fields and click submit button:



3. If there are no classes then this window appears



If there are classes, then automation process starts:

Automation Process:

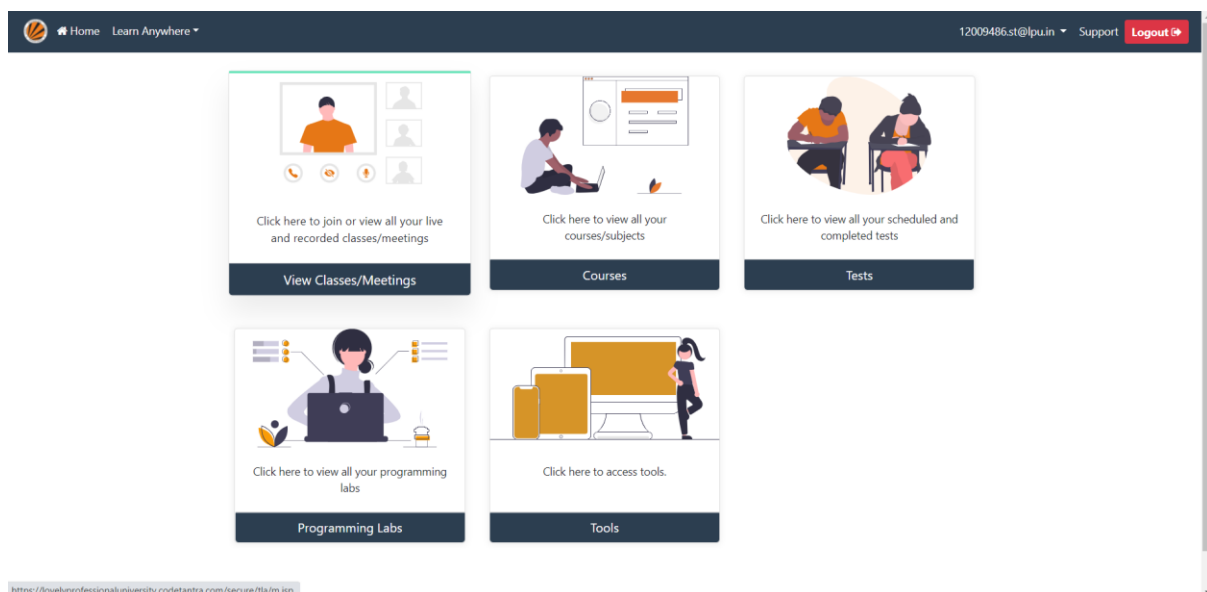
1. Opens the My Class website:



2. Enters the login credentials and clicks login:



3. Clicks View Classes Card-Container:



4. Selects the First class to attend:

The screenshot shows a web application interface for a learning management system. At the top, there is a navigation bar with a logo, 'Home', 'Learn Anywhere', and user information '12009486.st@lpu.in' with 'Support' and 'Logout' links. Below the navigation bar is a legend: Upcoming (blue square), Delayed (red square), Ongoing (green square), Completed (grey square), and Expired (grey square with diagonal lines). Below the legend is a calendar for November 2021, showing dates from 1 to 11. To the right of the calendar is a timetable for Monday, November 22, 2021 (India Standard Time). The timetable shows a list of classes with their start and end times. The first class is 'CSE205-Practical by : 14591:Virrat Devaser (K20BGCSE205)' from 9:00 to 10:00. Below the timetable is a 'Filters' section with 'Status' filters: Ongoing, Completed, and Upcoming. The 'Upcoming' filter is selected. Below the filters is an 'Apply' button.

Legend: Upcoming Delayed Ongoing Completed Expired
Click on the class/meeting name in the below timetable

November 22, 2021 (India Standard Time)

Monday

Time	Class/Meeting
9am - 10:00	UJTH421-Lecture by : 14846:Dr. Seema Mishra (K20BGMTH401)
10:00 - 11:00	INT306-Practical by : 13436:Cherry Khosla (K20BGINT306)
12:00 - 1:00	PEL131-Lecture by : 24881:Varinder Kaur (K20BGPEL131)
1:00 - 2:00	CSE211-Lecture by : 26156:Dr. Jayavadivel R (K20BGCSE211)
3:00 - 5:00	INT213-Lecture by : 23754:Sagar Pande (K20BGINT213)

Filters

Status

Ongoing Completed Upcoming

Apply

5. Clicks the Join Button:

The screenshot shows a web application interface for a learning management system. At the top, there is a navigation bar with a logo, 'Home', 'Learn Anywhere', and user information '12009486.st@lpu.in' with 'Support' and 'Logout' links. Below the navigation bar is a header for the class 'CSE205-Practical by : 14591:Virrat Devaser (K20BGCSE205) as seen by Yaswanth Bolisetty (12009486.st@lpu.in)'. Below the header is a 'Join' button. Below the 'Join' button is a section with 'Class Timings' (20 Nov 12:00 - 20 Nov 14:00), 'Groups' (K20BGCSE205), and 'Status' (Not started yet). To the right of this section is a 'Moderators' section with 'Recording' (N/A) and 'Class Attendance' (N/A). Below the 'Join' button is a 'Summary' section with a 'Agenda' link. Below the 'Summary' section is a message: 'Your attendance analytics are unavailable as you did not join this meeting.'

CSE205-Practical by : 14591:Virrat Devaser (K20BGCSE205) as seen by Yaswanth Bolisetty (12009486.st@lpu.in)

Join

Class Timings 20 Nov 12:00 - 20 Nov 14:00

Groups K20BGCSE205

Status Not started yet

Moderators Virrat Devaser

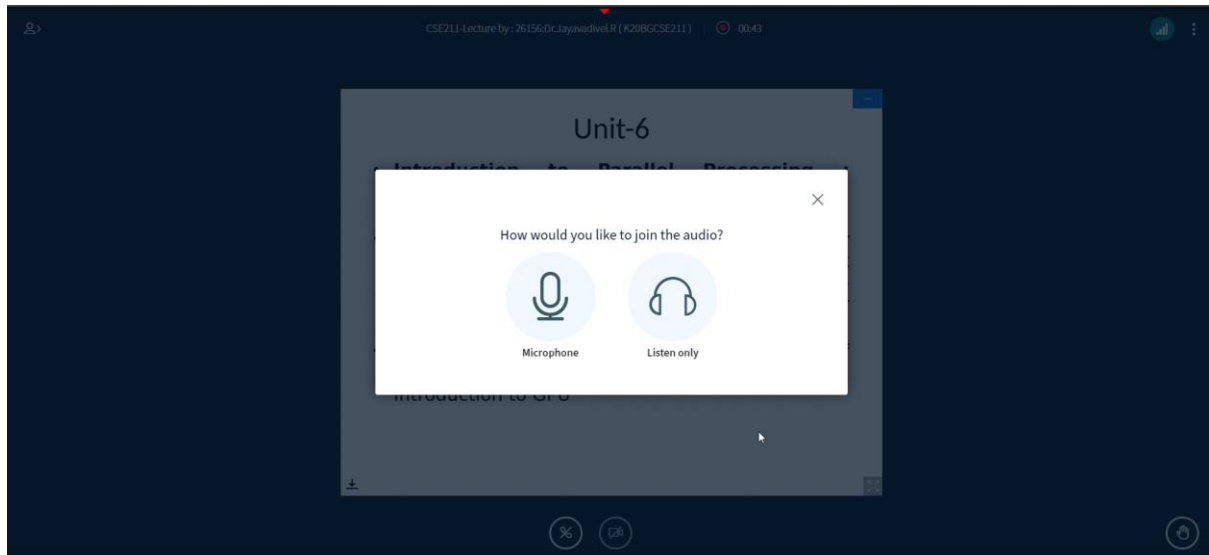
Recording N/A

Class Attendance N/A

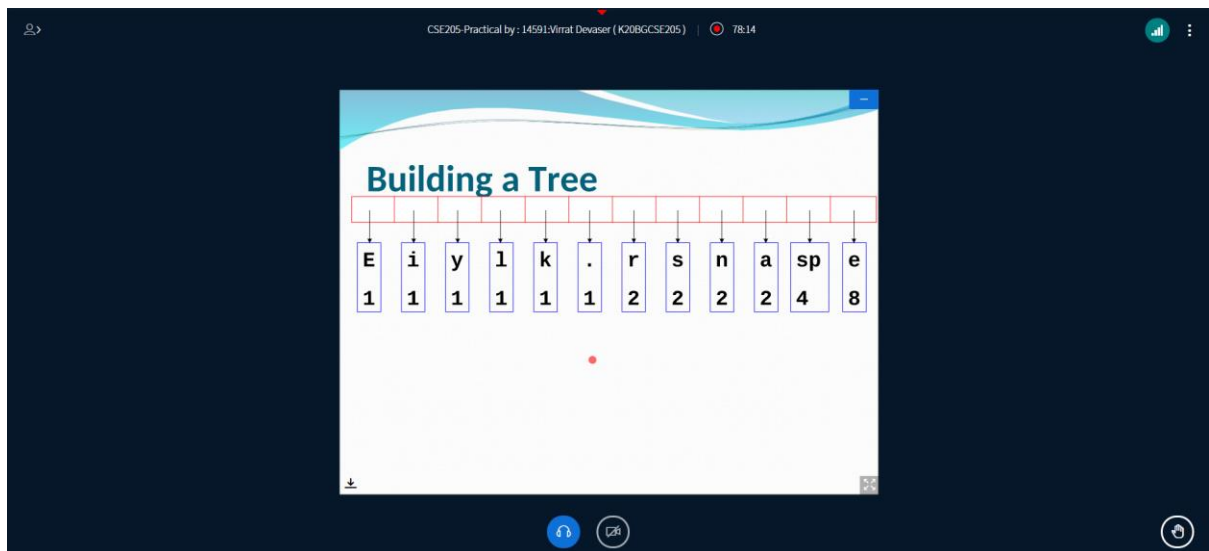
Summary Agenda

Your attendance analytics are unavailable as you did not join this meeting.

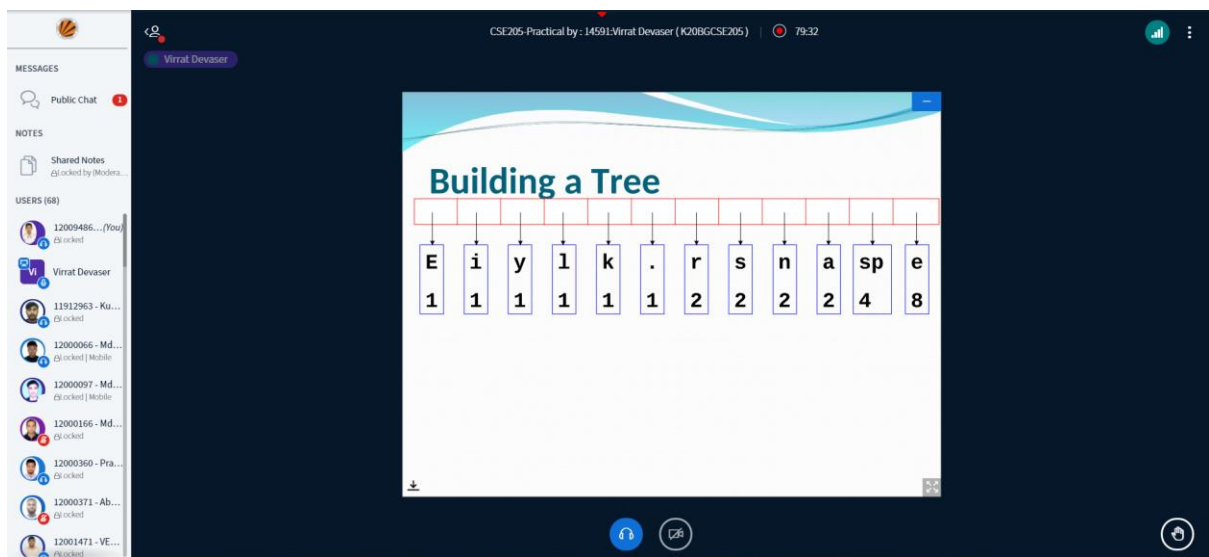
6. Clicks the Listen only Button:



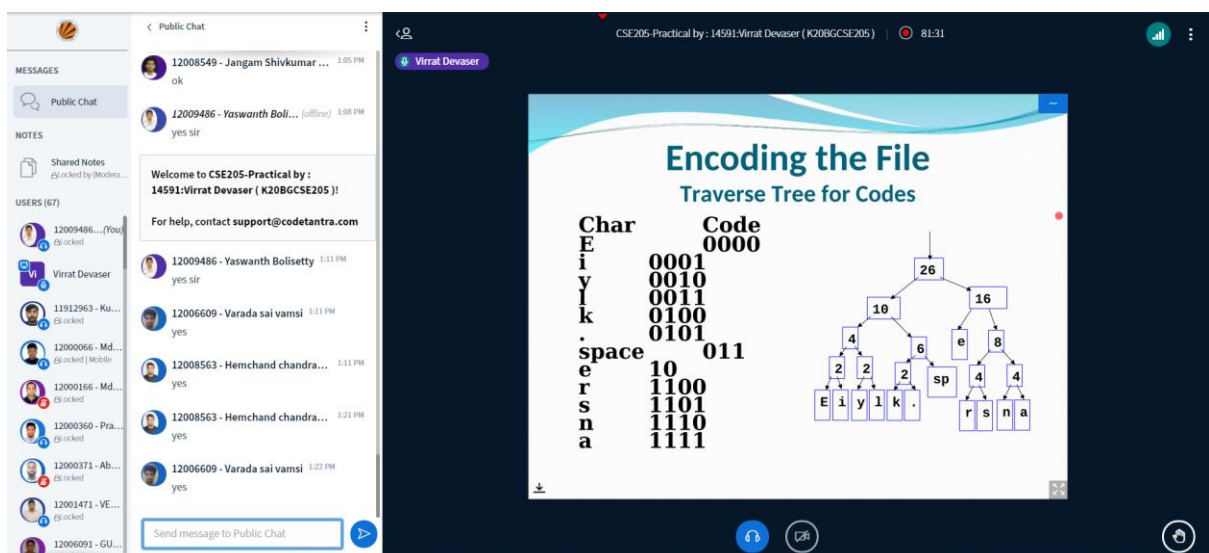
7.Once u click this public chat icon which is on the top left corner:



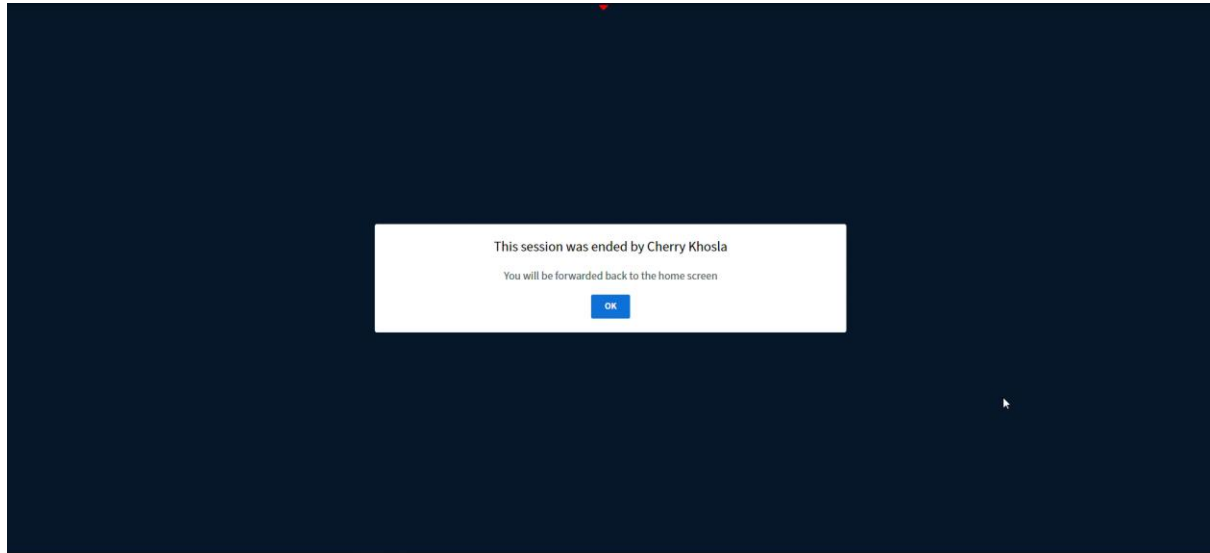
8. This opens page opens, then it automatically clicks public chat container :



9. Then this page appears, it will send “Good morning” or “Good afternoon” message according to the time:



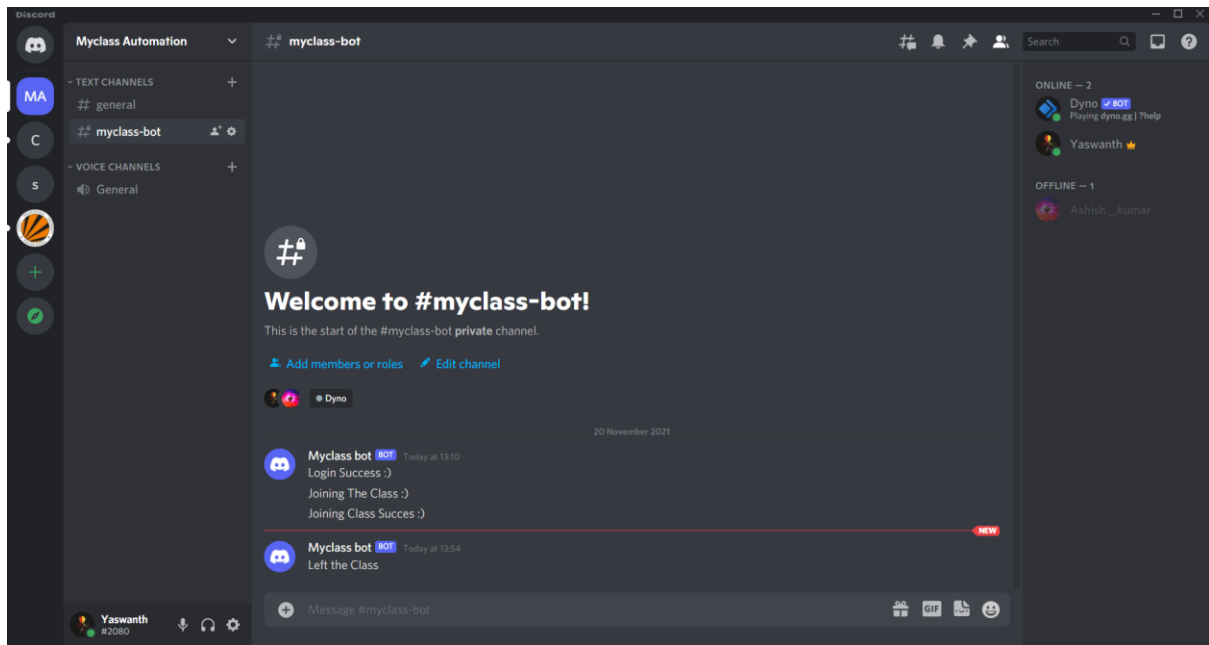
10. Once the teacher ends the meeting, this page appears then it closes the chrome window:



After this process it will wait for next class to start once it starts the same process is repeated but this time it selects the second class as its time for second class and next time third class and this process continues up to the last class present in the timetable.

Discord Notifications:

It notifies the actions done by it:



Explanation of Code:

1. Importing of the modules and libraries used:

```
from tkinter import *
from PIL import ImageTk, Image
import tkinter.font as font
import time
import pip
import calendar
from datetime import datetime
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from apscheduler.schedulers.blocking import BlockingScheduler
from dhooks import Webhook
```

2. Webhook and datetime:

```
webhook =  
Webhook("https://discord.com/api/webhooks/910720477427798026/3KAIFgmjpKP5tWp9Q  
ZyOSNF2J50QQu0-JpIicbvatiRevLqXQQ5wu4-y3si6l4Z_I71u")  
#webhook link of a discord channel  
datetime_now=datetime.now()  
#gets the date and time of current day
```

3.Tkinter part:

```
test_root = Tk()  
test_root.iconbitmap("lpu_logo.ico")  
test_root.geometry("948x632")  
test_root.minsize(948,632)  
test_root.maxsize(948,632)  
test_root.title("My Class Automation Control Center")  
img = ImageTk.PhotoImage(Image.open("background.png"))  
panel = Label(test_root, image=img)  
panel.place(x=-5, y=-5)  
  
myFont = font.Font(family='Helvetica', size=10, weight='bold')  
btn = Button(test_root, text="Exit", command=test_root.destroy,  
font=myFont,bg="#456ada")  
btn.place(x=880, y=580)
```

4. “Click Here To Install All Necessary Components” BUTTON :

```
4.  
5. def check_system():  
6.  
7.     def installing():  
8.         package1='selenium'  
9.         package2='apscheduler'  
10.        package3='dhooks'  
11.        pip.main(['install',package1])  
12.        pip.main(['install',package2])
```

```

13.     pip.main(['install',package3])
14.     def check():
15.         try:
16.             import selenium
17.             import apscheduler
18.             import dhooks
19.             final_message=Label(text="Sucessfully Installed : ")
20.             final_message.pack()
21.
22.         except ModuleNotFoundError:
23.             final_message=Label(text="Installation Failed : ")
24.             final_message.pack()
25.     installing()
26.     check()
27.
28. btn_2= Button(test_root, text="---> Click Here To Install All Necessary
    Components <---",
29.               command=check_system,font=myFont,fg="white", bg="#3488EF")
30. btn_2.place(x=37, y=30)
31. check_instruction_1=Label(text="Only One Time Required",font=('roboto',
    12, 'bold'),bg="#a58460").place(x=40, y=70)
32.

```

5. Username, password, submit buttons:

```

var1= StringVar()
var2=StringVar()

x, y="", ""
def get_value():
    global x, y
    x= var1.get()
    y= var2.get()
    test_root.destroy()
    automation()

user_name = Label(text="Registration Number :",font=('Avenir Next Condensed',
10, 'bold'),bg="#af8b66").place(x=560,y=60)
user_password = Label(text="Password :",font=('Avenir Next Condensed', 10,
'bold'),bg="#af8b66").place(x=560,y=100)
user_name_input_area = Entry(test_root,textvariable=var1).place(x=710,y=60)
user_password_entry_area =
Entry(test_root,textvariable=var2,show="*").place(x=710,y=100)
submit_button = Button(text="Submit",font=('Franklin Gothic Medium', 14,
'bold'),fg="red", bg="orange",command= get_value).place(x=750,y=135)

```

6.Login Function:

```
def automation():
    username_input = x
    password_input = y

    def login(Id):
        web = webdriver.Chrome() # chrome webdriver
        web.get("https://myclass.lpu.in") # opens myclass website
        time.sleep(1) # waits 1 sec
        web.maximize_window() # maximizes the chrome window
        username = web.find_element(By.XPATH,
        "/html/body/div[2]/div/form/div[6]/input[1]") # Locates username field
        username.send_keys(username_input) # enters registration number in
        username field
        time.sleep(1) # waits 1 sec
        password = web.find_element(By.XPATH,
        "/html/body/div[2]/div/form/div[6]/input[2]") # Locates password field
        password.send_keys(password_input) # enters password in password
        field
        time.sleep(1) # waits 1 sec
        login = web.find_element(By.XPATH,
        "/html/body/div[2]/div/form/div[7]/button") # Locates Login button
        login.click() # clicks Login button
        webhook.send("Login Success :)") # sends "Login Success :)"
        notification to discord channel
        time.sleep(1) # waits 1 sec
        View_classes = web.find_element(By.XPATH,
        "/html/body/div[9]/div/div[1]/div/div/div[1]/div/div[2]/a")
        # locates view_classes container
        View_classes.click() # Clicks View Classees Button
        time.sleep(1) # waits 1 sec
        class_elements =
        web.find_element(By.XPATH, "/html/body/div[1]/div[6]/div[3]/div[2]/div/div[2]/div/div/table/tbody/tr/td/div/div/div[3]/table/tbody/tr/td[2]/div/div[2]/a[{}]").format(Id))
        # Locates the class container respectively according to time
        class_elements.click() # clicks that class container
        time.sleep(1) # waits 1 sec
        try:
            join_button = WebDriverWait(web,
            150).until(EC.presence_of_element_located((By.XPATH,
            '//*[@id="meetingSummary"]/div/div/a/i')))
            # waits for 150 sec until the join button appears
            join_button.click() # clicks the join button once it appears
```

```

        webhook.send("Joining The Class :)") # sends "Joining The Class
:) " notification to discord channel
    except:
        webhook.send("Joining class failed :(") # if the join button
does't appear in 150sec then
        # sends "Joining class failed :(" notification to discord channel
    try:
        time.sleep(5) # waits 5 sec
        web.switch_to.frame(web.find_element(By.ID, "frame")) # finds the
frame by id and switch to it
        listen_only_button = WebDriverWait(web,
150).until(EC.presence_of_element_located((By.XPATH,
'/html/body/div[2]/div/div/div[1]/div/div/span/button[2]/span[1]/i')))
        # waits for 150 sec until the listen_only button appears
        listen_only_button.click() # clicks listen_only button once it
appears
        webhook.send("Joining Class Succes :)") # sends "Joining Class
Succes :) " notification to discord channel
    except:
        webhook.send("Not able to Join class :(") # if the listen_only
button does't appear in 150sec then
        # sends "Not able to Join class :(" notification to discord
channel
        # chatbox-----
        -----

        time_now = datetime.now.strftime("%H:%M:%S") # gets current time in
hours,minutes,seconds format
        if time_now >= "12:00:00": # checks if the current time is greater or
equal to 12:00:00
            chat = ("Good Afternoon") # if true then "Good Afternoon" value
is given to chat variable
        else:
            chat = ("Good Morning") # if false then "Good Morning" value is
given to chat variable
        try:
            public_chat = WebDriverWait(web,
60).until(EC.presence_of_element_located((By.ID, 'chat-toggle-button')))
            # waits for 30 sec until the public-chat container appears
            public_chat.click() # clicks public-chat container once it
appears
        except: # if the public-chat container does't appear in 60sec then
            pass # nothing happens
        try:
            greeting = WebDriverWait(web,
60).until(EC.presence_of_element_located((By.ID, 'message-input')))
            # waits for 30 sec until the chat-box appears
            greeting.send_keys(chat) # enters value present in chat variable
in chat-box
            greeting.send_keys(Keys.RETURN) # presses enter key

```

```

except: # if the chat-box doesn't appear in 60sec then
    pass # nothing happens
# chatbox-----
-----

try:
    class_ended_button = WebDriverWait(web,
7200).until(EC.presence_of_element_located((By.XPATH,
'//*[@id="app"]/div/div/div/div/button/span'))
    # waits for 7200 sec until the class-ended by teacher page appears
    class_ended_button.click() # clicks ok
    web.close() # closes the chrome window
    webhook.send("Left the Class") # sends "Left the Class"
notification to discord channel
except: # if the class-ended by teacher page doesn't appear in
7200sec then
    pass # nothing happens
    time.sleep(5) # waits 5 sec

```

7. Checks if time is greater than 5:00 pm then it sends message

```

total_time_now = datetime_now.strftime("%H") # gets the hour value from
current time
if total_time_now > "17": # checks if the current time hour value is
greater 17
    message_root = Tk()
    message_root.geometry("420x60")
    message_root.resizable(width=False, height=False)
    message_root.title("Wizard")
    exit_button = Button(message_root, text="Close",
command=message_root.destroy)
    exit_button.pack(side="bottom")
    final_message = Label(text="Classes done for today :)")
    final_message.pack()
    message_root.mainloop()

    print("Classes done for today :)")
    webhook.send("Classes done for today :)")
    quit() # if it is true then exits the program

```

8.Join Function:

```
def join(WEEK):
    sched = BlockingScheduler() # BlockingScheduler
    Year, Month, Day = datetime_now.year, datetime_now.month,
datetime_now.day
    # Year variable is assigned with current year,
    # Month variable is assigned with current Month,
    # Day variable is assigned with current Day.
    for i, Id in zip(WEEK.values(), WEEK.keys()):
        # i variable is assigned with values of dictionary which matches
with current weekday
        # Id variable is assigned with keys of dictionary which matches
with current weekday
        print(i, Id) # prints i and Id
        x = i.split(":") # splits dictionary values
        sched.add_job(login, run_date=datetime(Year, Month, Day,
int(x[0]), int(x[1]), int(x[2])), args=Id)
        # does this job at specific time mentioned in the dictionary
        sched.start() # starts the BlockingScheduler
```

9.Timetable entered as dictionary

```
# Time Table entered as dictionary-----
-----
MON  =
{"1":"9:00:00","2":"10:00:00","3":"12:00:00","4":"13:00:00","5":"15:00:00"}
TUE  =
{"1":"9:00:00","2":"12:00:00","3":"13:00:00","4":"15:00:00","5":"16:00:00"}
WED  = {}
THUR =
{"1":"9:00:00","2":"12:00:00","3":"13:00:00","4":"15:00:00","5":"16:00:00"}
FRI  = {"1":"9:00:00","2":"12:00:00","3":"13:00:00","4":"15:00:00"}
SAT  = {"1":"12:00:00","2":"15:00:00"}
# Time Table entered as dictionary-----
-----
```

10. Checks the current weekday and dictionary is assigned respectively:

```
week_day = calendar.day_name[datetime.now.weekday()] # gets the weekday name
# Assigns the corresponding dictionary according to current weekday
if week_day == "Monday":
    join(MON)
elif week_day == "Tuesday":
    join(TUE)
elif week_day == "Wednesday":
    print("Enjoy Your Wednesday No classes Today")
    webhook.send("Enjoy Your Wednesday No classes Today")
elif week_day == "Thursday":
    join(THUR)
elif week_day == "Friday":
    join(FRI)
elif week_day == "Saturday":
    join(SAT)
else:
    print("Enjoy Your Sunday No classes Today")
    webhook.send("Enjoy Your Sunday No classes Today")
test_root.mainloop()
```


Conclusion

My class automation is a python program that joins the class according to the given schedule at right time and leaves the class accordingly. This starts by taking registration number as username and password from GUI application and upon clicking submit button it registers the following input and proceeds to open chrome browser. Using the chrome web driver (which is imported from selenium)it enters the value of credentials in the right field and clicks the login button, after successful login it automatically navigates to the meeting selecting page where this program judges the meeting it should join by checking the current time, after that, it proceeds to join the class. All these function logs are being generated and sent to the discord bot as a message, which further confirms that the joining of the meeting was successful. When the teacher ends the meeting then the window automatically closes and checks if there's an upcoming class, if there is it repeats the same, if not It just waits till the next meeting starts, or in case when it's the last class of the day, it pops up a dialogue box consisting of a message which reads "No upcoming classes", and at this stage, the program ends.

References:

<https://selenium-python.readthedocs.io/>

<https://www.youtube.com/>

<https://pypi.org/>

<https://www.geeksforgeeks.org/>

<https://docs.python.org/3/library/>

<https://stackoverflow.com/>