Somaiya
TRUST

Title:LMS EVENT MANAGEMENT USING PYWEBIO

Problem Statement:

- In Engineering colleges as we know a lot of events take place.All these events can be helpful for students to bond as well as to boost their confidence and skill.But its quite possible that the information about these events don't reach the students or the targeted mass.In order to solve this problem we need a website that display the events along with its attributes and is also editable

Project description:The LMS event management system website is a website that is made using a python library known as PYWEBIO that allows you to build simple web applications without the knowledge of HTML and Javascript.

. On this website you can :

1.Display the events being carried out in our college

2.Add events 3.Delete events

.

SOMAIYA VIDYAVIHAR UNIVERSITY

K J Somaiya College of Engineering

Somaiya TRUST

How our code works?

- Firstly we created 2 classes Login and Event and then used getter setter method to get its details.
- We appended the details into a list named Eventlist and Loginlist respectively and then saved it in a csv file .
- This csv file is used to keep track of the events added and also the login credentials of the users.
- There are 2 separate csv files used …. 1 is Login.csv for login details and other is events.csv for event details.
- This csv is written read and saved using user built functions namely readEventlist,saveEventlist,readLoginlist,saveLoginlist.

Various other functions are used to add events,delete events,display events,validate the login credentials etc.

For the UI/UX part the pywebio consists of built in functions such as :

- **Put_row:Which places everything inside it in a row format**
- **Put_buttons:This is used to put buttons**

- **Put_scope:Scope is like a container which you can edit as per your convenience.**

- **We have also used css to do additional styling by adding a .style() on the output functions mentioned above**

Outputs:

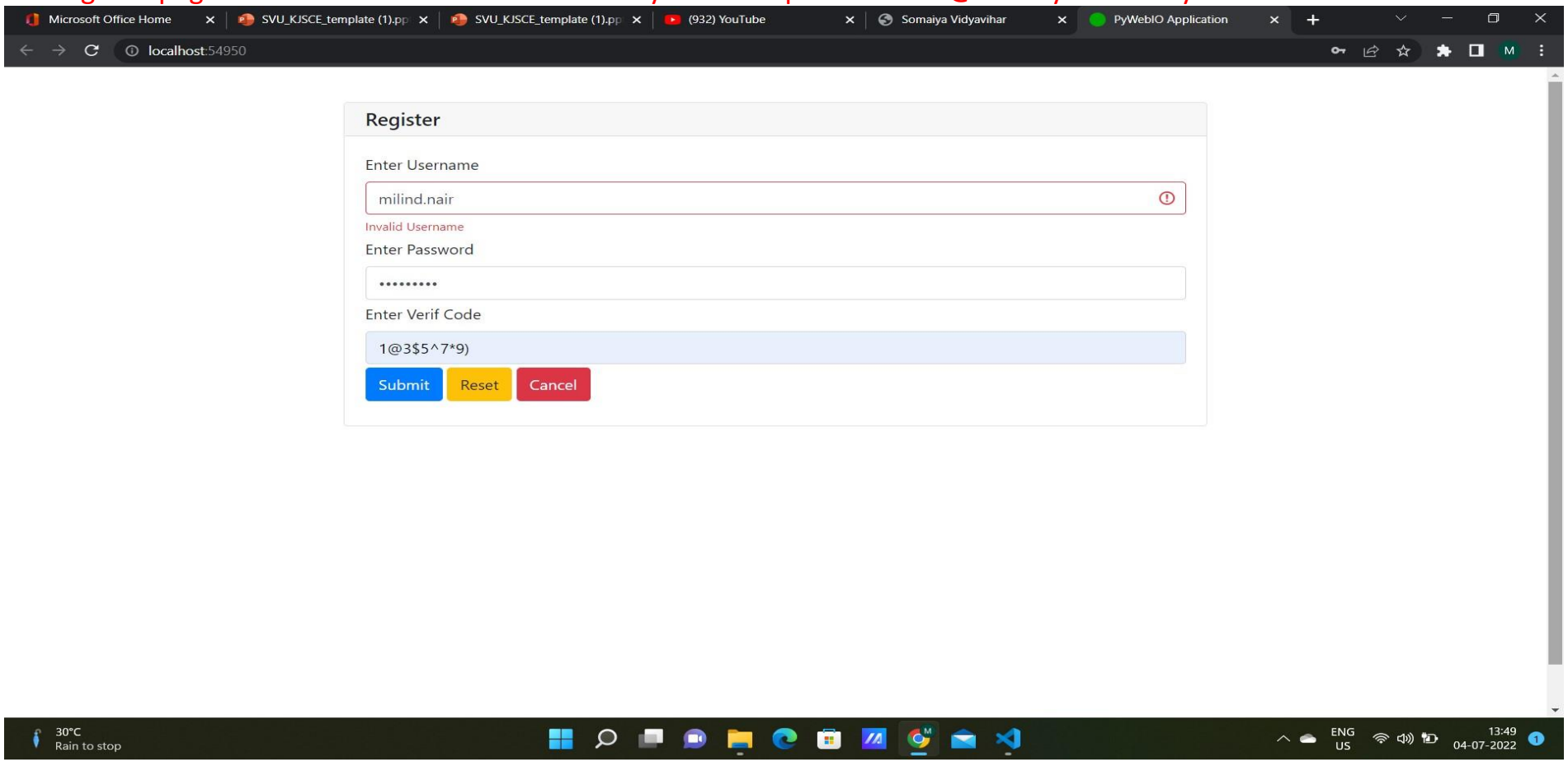
- 1.landing page

Login

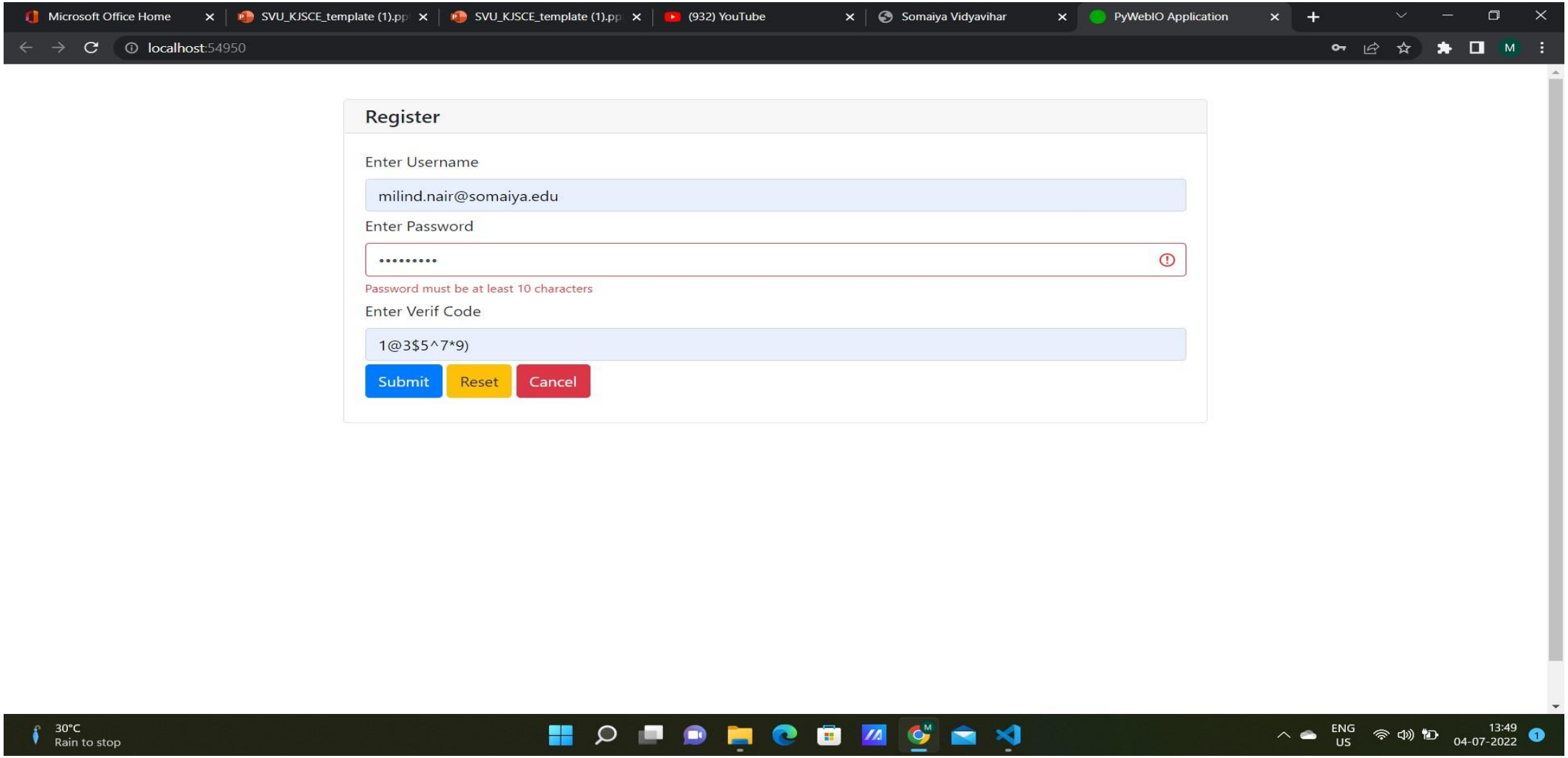
Register

Display Events

2.Register page In this it shows the error because you are required to use @somaiya.edu only



2.Register page In this it shows the error because you are required to have @ in your password



2.Register page

In this it shows the error because you are required to put the verification key given to you i.e 1@3$5^7*9)

Microsoft Office Home | SVU_KJSCE_template (1).pp | SVU_KJSCE_template (1).pp | (932) YouTube | Somaiya Vidyavihar | PyWebIO Application

localhost:54950

Register

Enter Username

milind.nair@somaiya.edu

Enter Password

••••••••••••••

Enter Verif Code

asffd

Invalid Code

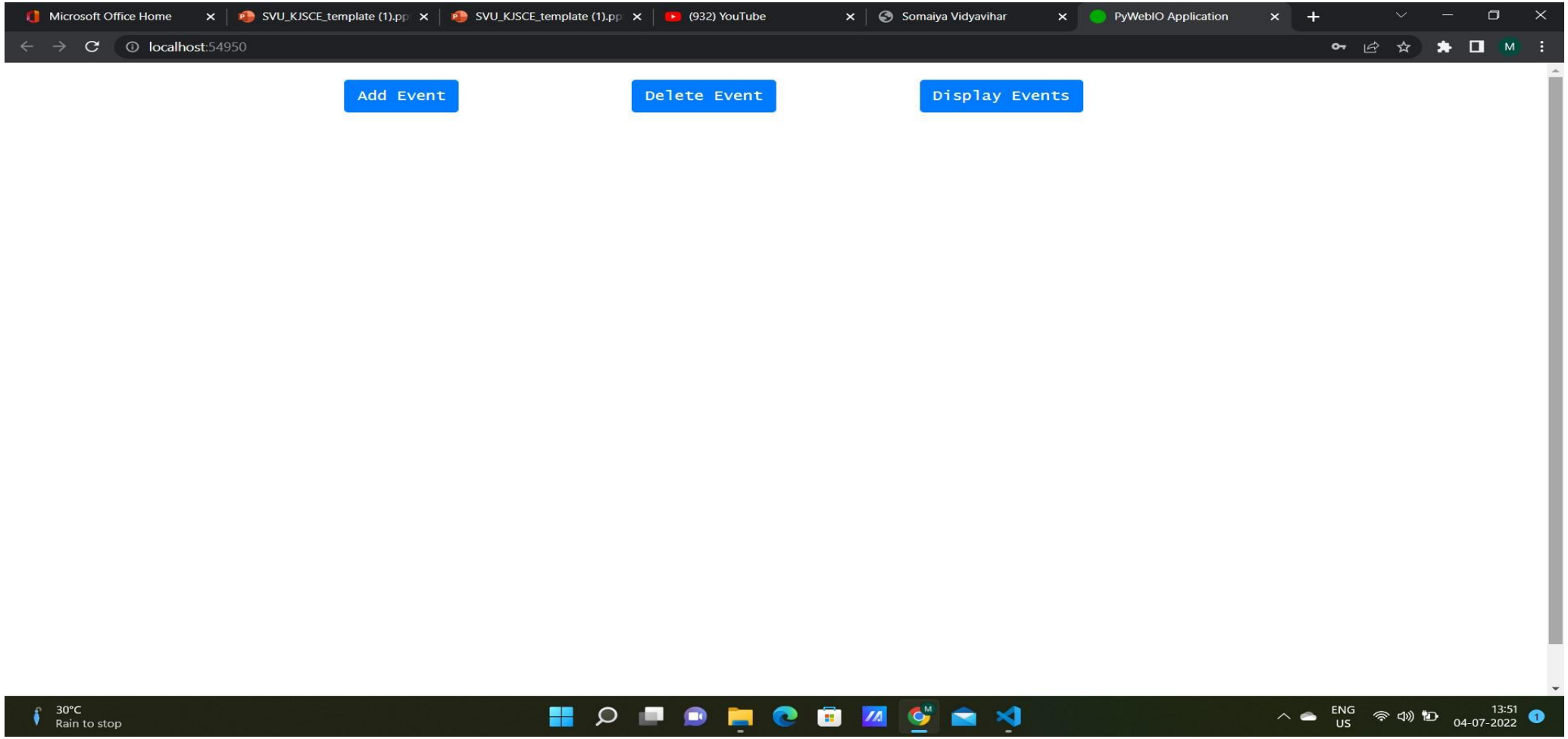
Submit Reset Cancel

30°C
Rain to stop

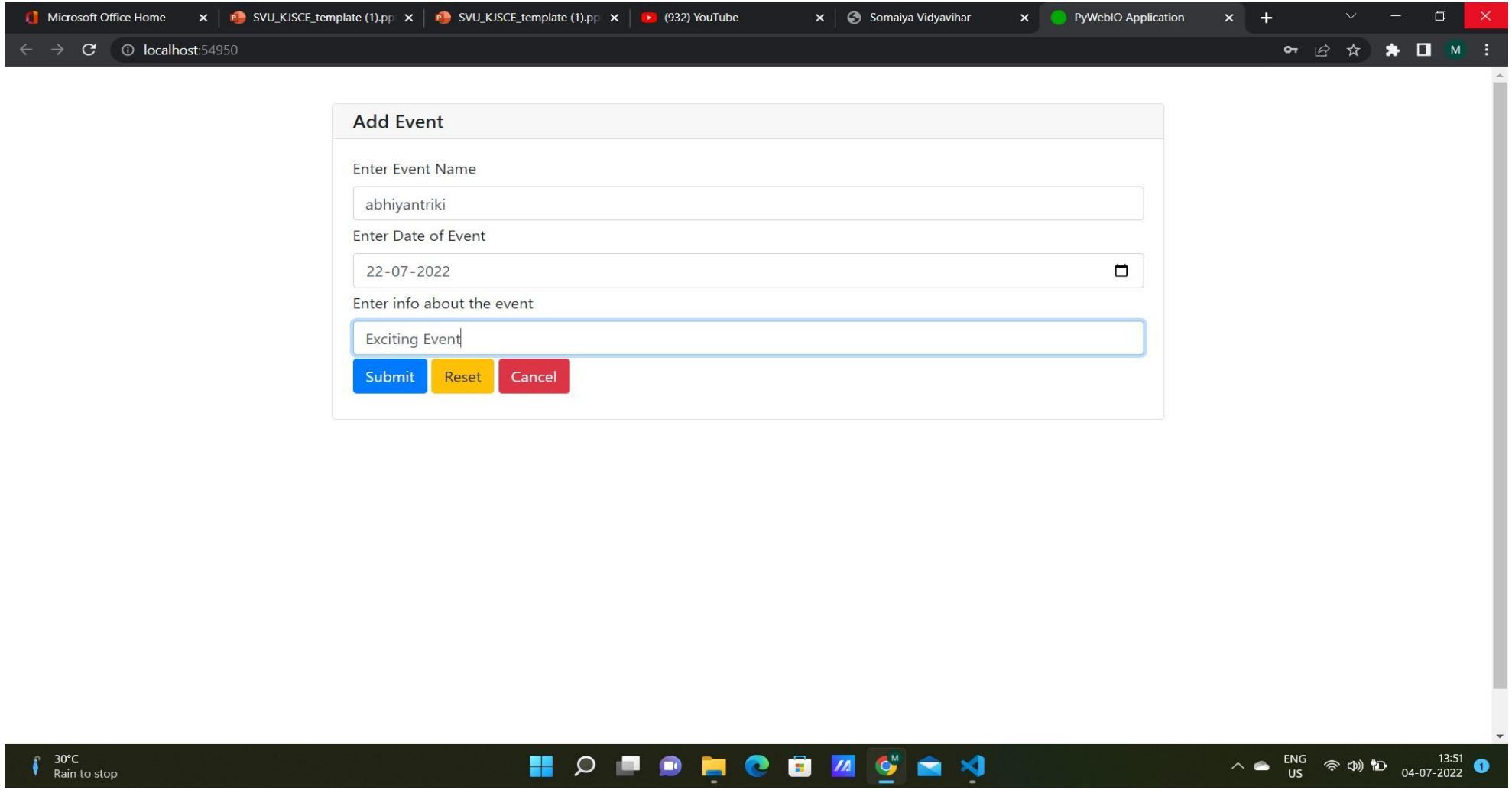
ENG
US

13:49
04-07-2022

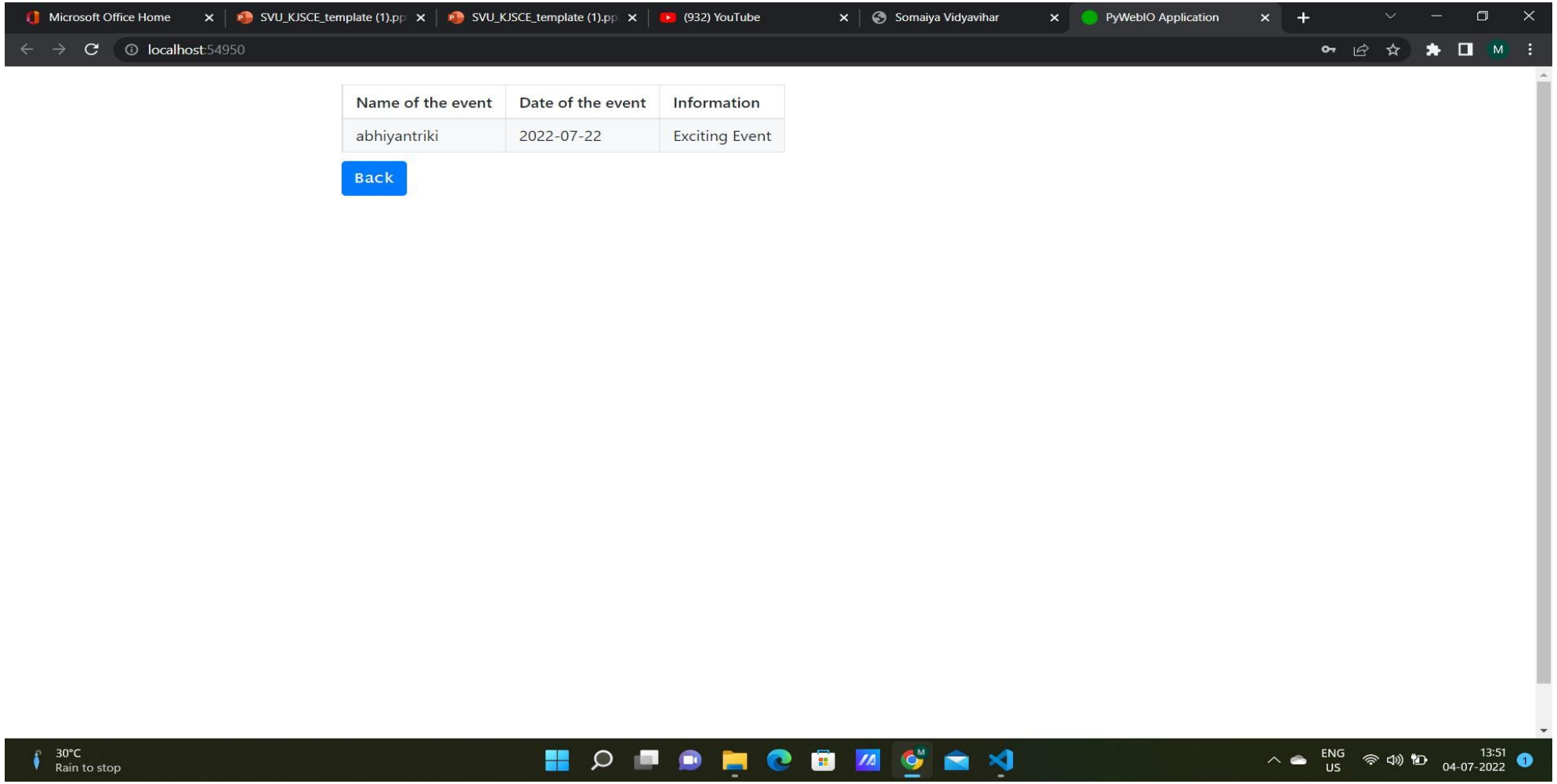
3.options Page where you can choose one of the following



4.Add event Page



5.Display event Page



6.Delete event Page...

Here the name should be specific or else it will throw a error

Microsoft Office Home | SVU_KJSCE_template (1).pp | SVU_KJSCE_template (1).pp | (932) YouTube | Somaiya Vidyavihar | PyWebIO Application

localhost:54950

Delete Event

Enter name of the event you want to delete

abhiyantriki

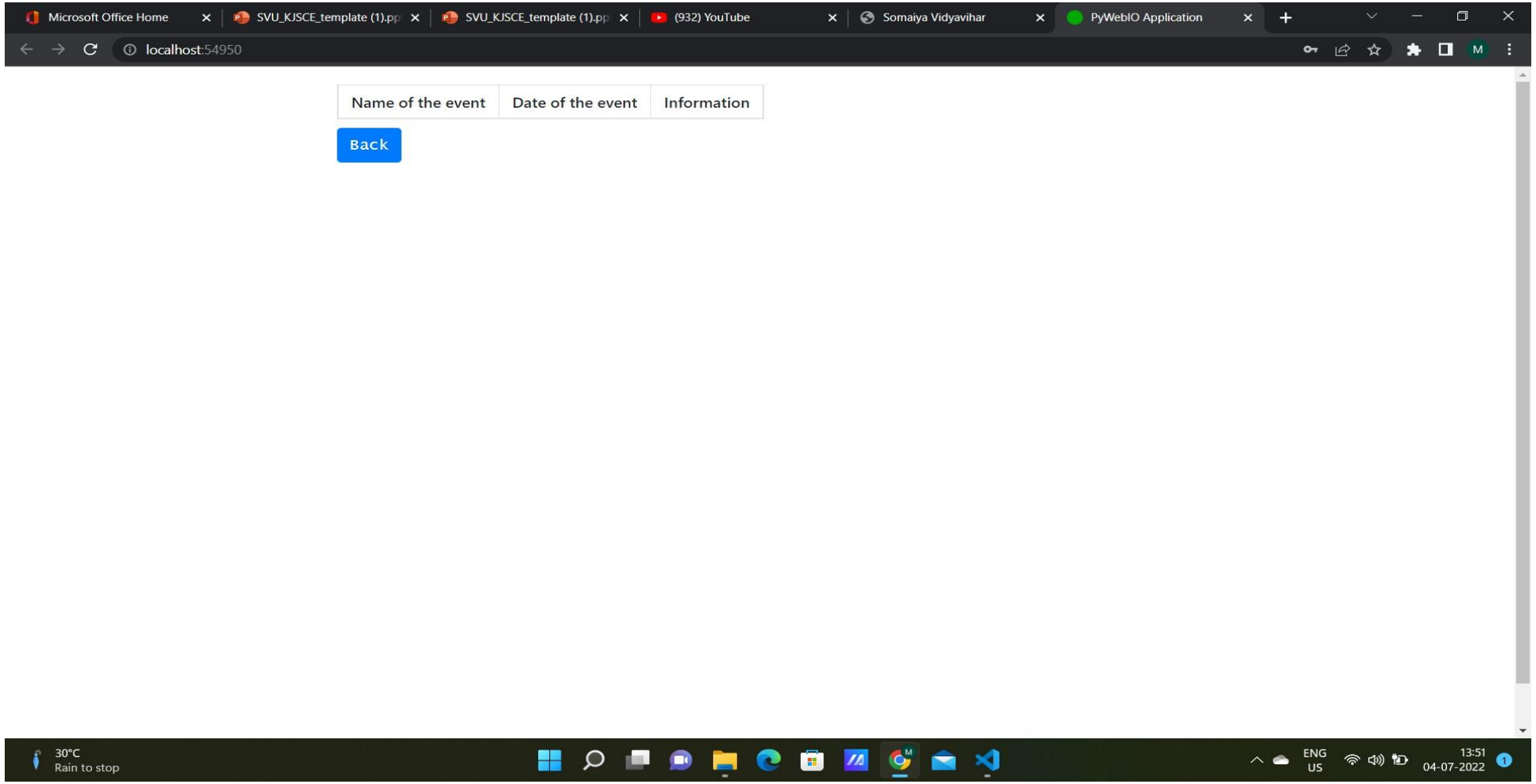
Submit Reset Cancel

30°C
Rain to stop

ENG
US

13:51
04-07-2022

7.Display event after deleting



Code:

```
# importing necessary libraries
import pywebio.output as pout
import pywebio.input as pimp
import csv
```

```
class Event:  # Event class

    def __init__(default, name, date, info):  # constructor
        default.name = name
        default.date = date
        default.info = info

    ''' # setter methods
    def setDate(default, date):
        default.date = date'''

    def setInfo(default, info):  # short one line info
        default.info = info

    # getter methods
    def getName(default):
        return default.name

    def getDate(default):
        return default.date

    def getInfo(default):  # short one line info
        return default.info

    def printDetails(default):  # print details of the event
        print(default.getName())
        print(default.getDate())
        print(default.getInfo())

class Login:
    def __init__(self, name, Pass):  # constructor
        self.name = name
        self.Pass = Pass
        print("Hello")

    ''' # setter methods
```

```
    def setDate(default, date):
        default.date = date'''

    # getter methods
    def getName2(self):
        return self.name

    def getPass(self):
        return self.Pass


    def printDetails(self):  # print details of the Login
        print(self.getName2())
        print(self.getPass())

Loginlist = []  #GLOBAL VARIABLE
Eventlist = []  # GLOBAL VARIABLE

def readEventlist():
    with open('events.csv', 'r') as csvfile:
        for i in csvfile:

            i = i.split(",")
            if(len(i)==3):
                Eventlist.append(Event(name=i[0], date=i[1], info=str(i[2])))
        csvfile.close()

def readLoginlist():
    with open('Login.csv', 'r') as csvfile:
        for i in csvfile:

            i = i.split(",")
            if(len(i)==2):
                Loginlist.append(Login(name=i[0], Pass=i[1]))
        csvfile.close()
```

```
def saveEventlist():
    '''
    Erases file as this is a very short list and not a huge database file.
    :return: none
    '''
    with open('events.csv', 'a') as csvfile:
        csvfile.truncate(0)
        writer = csv.writer(csvfile)
        for i in Eventlist:
            writer.writerow([i.getName(), i.getDate(), i.getInfo()])
        csvfile.close()

def saveLoginlist():
    '''
    Erases file as this is a very short list and not a huge database file.
    :return: none
    '''
    with open('Login.csv', 'a') as csvfile:
        csvfile.truncate(0)
        writer = csv.writer(csvfile)
        for i in Loginlist:
            writer.writerow([i.getName2(), i.getPass()])
        csvfile.close()

def addEvent(name, date, info):  # add event to list
    '''

    :param name: name of event
    :param date: date of event
    :param info: info about event
    :return: none
    '''

    Eventlist.append(Event(name=name, date=date, info=info))
    saveEventlist()  # save to database
```

```
def addUser(name, Pass):  # add event to list
    '''

    :param name: name of event
    :param date: date of event
    :param info: info about event
    :return: none
    '''

    Loginlist.append(Login(name=name, Pass=Pass))
    saveLoginlist()  # save to database

def deleteEvent(name):  # delete event from list
    '''

    :param name: name of event
    :return: none
    '''
    '''
    Here it is assumed that if the event is present in the list then it is deleted else pass silently
    '''
    for i in Eventlist:
        if i.getName() == name:
            Eventlist.remove(i)
    saveEventlist()  # save to database

def UserCheck(data2):
    print("in usercheck")
    name=data2['username']
    Pass=data2['password']
    j=0
    for i in Loginlist:
        print(i.getPass())
        print(i.getName2())

        if i.getName2() == name :
```

```
            j=1
            if (i.getPass())[0:-1] == Pass:
                Continue()
                j=2
        if(j==1 or j==2):
            break

    if(j==0):
        return('username','username not found')
    elif (j==1):
        return('password','Invalid password')




def editEvent(name, date, info):  # edit event info (Never used)
    '''

    :param name: name of event
    :param date: date of event
    :param info: info of event
    :return: none
    '''
    '''
    Editing will be carried out as deleting the event and adding a new one
    '''
   # Editing the event.
    for i in Eventlist:
        if i.getName() == name:
            i.setDate(date)
            i.setInfo(info)
    saveEventlist()  # save to database


def printEvents():  # print events Debug function
    '''
```

```
    :return: none
    '''
    print()
    for i in Eventlist:
        i.printDetails()
        print("-----------------------------------------------------")


def displayEvents():
    pout.clear("scopeV")
    # display events from list in scope1
    '''

    :return: None
    '''
    displaylist = [['Name of the event', 'Date of the event',
                    'Information']]  # lot to feed in put_table function contains a list of list.
    for i in Eventlist:
        displaylist.append([i.getName(), i.getDate(), i.getInfo()])
    pout.clear("scope1")  # clear prev display
    with pout.use_scope('scope1'):
        pout.put_table(displaylist)
        pout.put_button("Back", onclick=Continue).style('font-family: "Lucida Console", "Courier New", monospace;font-weight:bold;')




def notBlank(data):
    # check if event parameters are blank else returns False
    '''

    :param data: input data from pywebio
    :return: error message or False
    '''
    if (data['name'] == ""):
        return ('name', 'Name cannot be blank!')
```

```
    if (data['date'] == ""):
        return ('date', 'Date cannot be blank!')
    if (data['info'] == ""):
        return ('info', 'Info cannot be blank!')
    return False


def addEventWeb():
    pout.remove("scopeV")
    # add event from pywebio to list
    '''

    :return: none
    '''
    data = pimp.input_group("Add Event", [
        pimp.input('Enter Event Name',placeholder='Pls Enter Event Name', name='name' ,required=True),
        pimp.input('Enter Date of Event',
                name='date', type=pimp.DATE,required=True),
        pimp.input('Enter info about the event', name='info',required=True,placeholder='Pls Enter Event Details')],
        validate=AddEventValidate,cancelable=True
        # check if event exists or is blank to show error. error is handled by pywebio
    )
    if(data==None):
        Continue()
    else:
        addEvent(name=data['name'], date=data['date'], info=data['info'])
        Continue()


def AddEventValidate(data):
    # check if event exists or is blank to show error. error is handled by pywebio
    '''

    :param data: input data from pywebio
    :return: Tuple of (name, error)
    if the parameters are blank then it returns error for the parameter which is blank.
```

```
    Then it checks if the event exists or not and returns the error accordingly.
    '''
    if (notBlank(data) != False):  # if not blank return error message
        return notBlank(data)  # must return a tuple of (name, error)
    if (exists(data) == True):  # if already present then return error message
        return ('name', 'Event already exists!')
    pout.scroll_to('scopeV')


def DeleteEventValidate(data):
    # check if event exists or is blank to show error. error is handled by pywebio
    '''

    :param data: input data from pywebio
    :return: Tuple of (name, error)
    if the name is blank then it returns error
    Then it checks if the event exists or not and returns the error accordingly.
    '''
    if (data['name'] == ""):
        return ('name', 'Name cannot be blank!')
    if (exists(data) == False):  # if not present then return error message
        return ('name', 'Event doesnt exists!')


def exists(data):
    # check if event exists
    '''

    :param data: input data from pywebio
    :return: True if name already in Eventlist and false if name not found in Eventlist
    '''
    exists = False
    for i in Eventlist:
        if (i.getName() == data['name']):
            exists = True
```

```
    return exists


def deleteEventWeb():
    pout.remove("scopeV")
    '''

    :return: None

    Deleted events from the web using the name parameter.
    '''

    data = pimp.input_group("Delete Event", [
        pimp.input('Enter name of the event you want to delete', name='name')],
        validate=DeleteEventValidate,cancelable=True
        # check if event exists or is blank to show error. error is handled by pywebio
    )
    if(data==None):
        Continue()
    else:
        deleteEvent(name=data['name'])
    with pout.use_scope('scopeV',clear=True):
        pout.put_row([
            pout.put_button("Add Event", onclick=addEventWeb),  # a group of buttons
            pout.put_button("Delete Event", onclick=deleteEventWeb),  # a group of buttons
            pout.put_button("Display Events", onclick=displayEvents)]).style('font-family: "Lucida Console", "Courier New", monospace;font-weight:bold;')  # a group of buttons

def Login_func():

    pout.remove('scopeLogReg')


    data2 = pimp.input_group("Login", [
            pimp.input('Enter Username', name='username',required=True),
            pimp.input('Enter Password',
                name='password', type=pimp.PASSWORD,required=True)
```

```
        ],
        validate=UserCheck,
        cancelable=True

        )
    if((data2)==None):
        restart()


def Register():
    pout.remove('scopeLogReg')
    pout.scroll_to('scopeV')

    data3 = pimp.input_group("Register", [
        pimp.input('Enter Username', name='username',required=True),
        pimp.input('Enter Password',
                name='password', type=pimp.PASSWORD,required=True),
        pimp.input('Enter Verif Code',
                name='code',required=True),
        ],
        validate=is_valid,
        cancelable=True

        )
    if((data3)==None):
        restart()

def is_valid(data3):
    if(len(data3['username'].split(".."))==3):
        if((data3['username'].split("@"))[1]=="somaiya.edu"):
            pass
        else:
            return('username',"Invalid Username")
    else:
            return('username',"Invalid Username")
    if(len(data3['password'])>=10):
```

```
        if((data3['password']).count("@")>=1):
            Pass2=(data3['password']).replace("\n","")
            addUser(name=data3['username'], Pass=Pass2)
            Continue()
        else:
            return('password',"Password must have at least one \"@\" symbol ")
    else:
        return('password',"Password must be at least 10 characters")
    if(data3['code']=="1@3$5^7*9)"):
        pass
    else:
        return('code',"Invalid Code")

def restart():
    #pout.remove('scopeLogReg')
    pout.remove('scope1')
    with pout.use_scope('scopeLogReg'):
        pout.put_row([
            pout.put_button("Login", onclick=Login_func),  # a group of buttons
            pout.put_button("Register", onclick=Register),
            pout.put_button("Display Events",onclick=displayEvents2)]).style('font-family: "Lucida Console", "Courier New", monospace;font-weight:bold;')  # a group of buttons

def Continue():
    print("in continue")
    pout.remove('scope1')
    with pout.use_scope('scopeV'):
        pout.put_row([
            pout.put_button("Add Event", onclick=addEventWeb),  # a group of buttons
            pout.put_button("Delete Event", onclick=deleteEventWeb),  # a group of buttons
            pout.put_button("Display Events", onclick=displayEvents)]).style('font-family: "Lucida Console", "Courier New", monospace;font-weight:bold;') # a group of buttons

def displayEvents2():
    pout.remove('scopeLogReg')
    displaylist = [['Name of the event', 'Date of the event',
                    'Information']]  # lot to feed in put_table function contains a list of list.
    for i in Eventlist:
```

```
        displaylist.append([i.getName(), i.getDate(), i.getInfo()])
    with pout.use_scope('scope1'):
        pout.put_table(displaylist)
        pout.put_button("Back", onclick=restart).style('font-family: "Lucida Console", "Courier New", monospace;font-weight:bold;')

readEventlist()  # Get initial list of events
readLoginlist()
print(Loginlist)
with pout.use_scope('scopeLogReg'):

    pout.put_row([
        pout.put_button("Login", onclick=Login_func,color='success'),  # a group of buttons
        pout.put_button("Register", onclick=Register,color = 'warning'),
        pout.put_button("Display Events",onclick=displayEvents2)
        ],size='30%').style('font-family: "Lucida Console", "Courier New", monospace;font-weight:bold;font-size:80px;margin-color:solid green')  # a group of buttons
```