Project Report On

"PHYSICS NUMERICAL SOLVER"

Submitted by

DURJAN SINGH A	**XII COMMERCE**
DHARMESH M	XII-SCIENCE
EBEN GORKY S J	XII-SCIENCE

Guided by

AMEENOOR RASEED A R, M.C.A., B.Ed.,

VENI M, M.E., B.Ed.,

GEETHAANJALI ALL INDIA SENIOR SECONDARY SCHOOL,

THINDAL, ERODE - 638 012.

CERTIFICATE

This is to certify that **DURJAN SINGH A (Reg. No.)** of class XII-COMMERCE, **Geethaanjali All India Senior Secondary School, Thindal, Erode**, has successfully completed his Project Report in Computer Science on the topic of **"PHYSICS NUMERICAL SOLVER"** for the partial fulfillment of **AISSCE** as prescribed by **CBSE** in the year **2023-2024.**

Date:

Registration number:

Signature of the Guide **Signature of the Principal**

Signature of the Internal Examiner

Signature of the External Examiner

ACKNOWLEDGEMENT

I am utterly humbled and overwhelmed with gratitude for everyone who has assisted us in transforming these concepts into something concrete that is well above the simple level.

I want to express my gratitude to my principal, **Mrs. Subbulakshmi V, M.A., M. Ed., M.Phil., Ph.D.,** who gave me the chance to finish my project by giving me library references, a computer lab for internet research, moral support, etc.,

I want to thank my teachers in computer science, **Mr. Ameenoor Raseed A R, M.C.A., B.Ed.,** and **Miss. Veni M, M.E., B.Ed.,** for their important guidance in identifying potential problems, motivation, and advice for this project.

I also want to express my gratitude to my friends, who frequently supported me and offered assistance when I needed it during the project development.

“PHYSICS NUMERICAL SOLVER”

TABLE OF CONTENTS

ABSTRACT

ABSTRACT

The document contains the software project named "**PHYSICS NUMERICAL SOLVER".** This project is mainly focussed on helping out the student who is struggling to solve the physics numericals and graphs. This is designed in such a way that; the user can easily search the required formula to solve the given numerical.The user also gets the respective formula to solve the problem along with the final answer,So that the user can use it for solving the problems of such types in the future,this can solve multiple types of numerical questions related to a single concept so this will reduce the effort made by the user to search a numerical by just asking them to enter what he needs to find and enter the given values from the question,and the user can add or remove any formula if requiredwhich makes it as an advantage for the software over some other softwares.

"PHYSICS NUMERICAL SOLVER"

INTRODUCTION

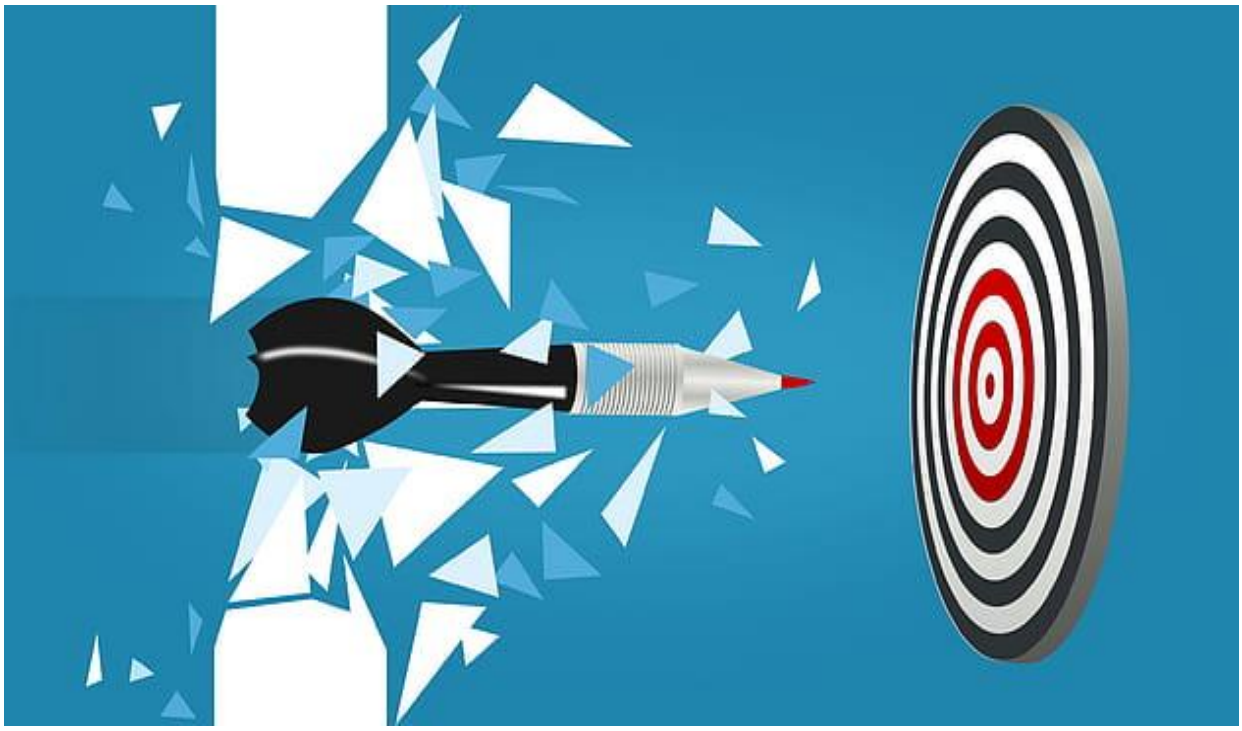
INTRODUCTION

The name of the project is **"PHYSICS NUMERICAL SOLVER"**. The main objective of the project is to make an easy way for the student to solve or check a physics numerical or a graph. We have designed the project in such a way that everyone can easily acess and give the required input from the question and find the solution along with an appropriate formula for solving the same kind of question in the future.

This project will help the users to find solution even if they are out of network as the project is working with help of python, which is a big advantage for the users who are suffering to find the solution when the they are out network which makes them use the software seamlessly.

This software helps the student to free up themselves from the stress of solving the difficult physics numericals and graphs, which also helps in bringing an interest in solving them as the solutions are now made easily available along with the appropriate formula with this project made by us. Now let us see a detailed explanation of the project made by us along with the runtime screenshots and backround and foreground codes used.

OBJECTIVE



OBJECTIVE

The main objective of the project is to create interest in solving physics numerical by helping out the user to find a solution for a difficult numerical by just simply searching the concept from which the question is from. As the the problem of finding a solution is now made simple, the user will get a positive notion on solving physics numericals and graphs.

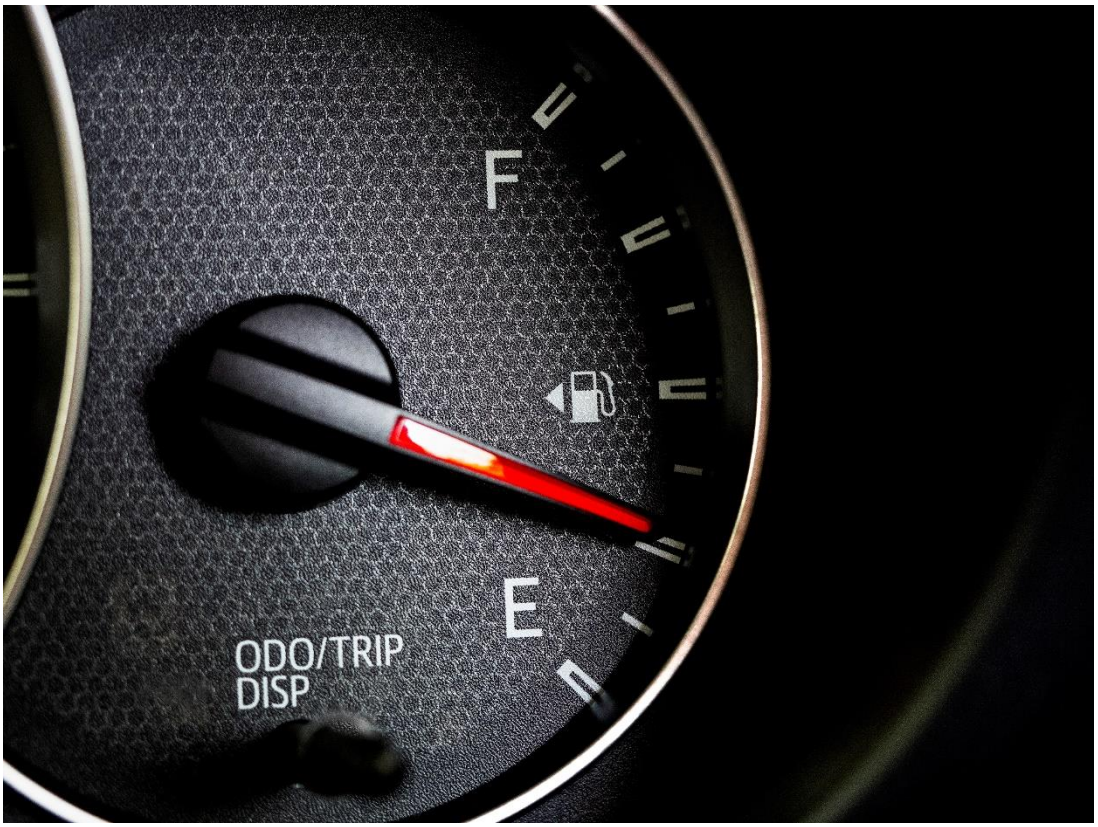
When the user starts the application, he will get a window showing the concepts, the user also gets an option to search for the required concept. And also, the user will get the appropriate formula to solve any question related to the concept in the future.

If the user wants to add or remove any concept if and if not required an option for that is also given. Since, the project is made after more trials there will be very less chance for the program to give the user a wrong solution. Hence the application is more accurate for a twelth grader to refer for a solution in the tough concepts of physics.

LIMITATIONS

OF

EXISTING SYSTEM



LIMITATIONS OF EXISTING SYSTEM

The existing system of finding a solution for the physics numerical is way harder than solving by our own as we are required to type the question fully,even though there are many options for searching a question using the image with the help of image search and lens etc,..it is not working in all the occasions as we may have question paper which is not clear or we may be working with a PC and the webcam couldn't have supported.This project can be operated withouted any limitations that was mentioned above.

IMPORTANCE

IMPORTANCE

ENHANCEMENT

The main objective of the PHYSICS NUMERICAL SOLVER is to develop the interest in physics by redusing the problem of finding a solution for a numerical or a graph.

ACCURACY

In the existing system of finding a solution for a particular problem is having even more limitations by giving the user with approximate values. But our system will enhance it by giving full answer without any approximation.

USER-FRIENDLY

There are many things which makes our system user friendly, some of them are.

- It is easy for the user to search a specific formula from n number of existing formulae.
- It is very convient for the user to add a formula, if the user wishes to.
- The graphical user interface provides a best interface between user and machine.
- The user will be asked to enter the value with the specific variable used in the formula.
- The user will be getting a proper message if they enter a improper value.

INPUTDATA

INPUT DATA

The main objective of making input data is to make the user and his friends to enter the information error free. The user has to follow the correct steps:

- Giving the value of a variable without any special characters.
- Must not compute the formula value without giving a data.

TYPES OF OUTPUT

TYPES OF OUTPUT

In this project the system has two different outputs, the calculated value of the formula and graphical respentiction of the function. The output varies with formula selected by the user and values entered by them. Thus, the program is made for easily understanding the formula and its application using python and tkinter to give best experience to the user. The graph drawn by the formula gives object doing an event evolve over time or other peramater. The information such as error and other important info are made to be pop up to notify the user to give appropriate answer that the user needs.

IMPLEMENTATION

IMPLEMENTATION

The implementation process is a long process. We first install and check whether the application has the perfect client requirements. Then, we run every formula of the application and check it has 0% error or glitch which would be a headache for the users. We make this coding very simple so that it only uses a less working space and supports every system which is installed with matplotlib, tkinter and math modules. After hours of checking and confirming, we can now definitely assure you that it is 100% error free and could give you a good time.

FURTHER LOOKING FORWARD

FURTHER LOOKING FORWARD

In our project, users have the option to add a required formula for their use. Also it is virus free and no one can configure the application's safety measures. The main characteristics that we should look forward about a educational application are security, extensibility and maintainability. These features are included in this project.The application is made in such a way that it works seamlessly and solves many numerical problems within in a short period of time with the help of search function for searching a concept.And this helps the user to further solve the same kind of problems in the future by providing the appropriate formula related to the particular concept.

SOURCE CODE


.show
.dropdown
.dropdown-toggle::before
tent-catalog{padding:0}
.clear-filters-toggle:hover
row}.tbb-subcats-ul{display:flex;
lex:1;justify-content:start}
or:#a43bf4}.tb-t-r-buttons-a
transition:all 220ms ease;
#acacac;max-width:
width:36px;transition:
play:none}

SOURCE CODE

GUI code

```
from tkinter import *
from tkinter import messagebox
import oxilanari_functions as oxi
import functions_project as f_p
import graph_function as g_p

oxi_info = oxi.get_function_name()
oxi_gra_info = oxi.get_graph_function_name()

global win
win = Tk()
win.configure(bg = 'light green')
win.title('CS project')
win.geometry('600x600')
win.resizable(0,0)

fram_intro = None
fram_for = None
fram_gra_con = None
fram_add_f_g = None
fram_run_fun = None
fram_graph = None
ans = 0
el = None
l_ans = None
n_list_v = list(oxi_info)
d = 0

def run_fun(name):
```

```
    global n_list_v
    global ans
    global el
    global l_ans
    n_list_v = []
    def find_ans():
        global ans
        global el
        global l_ans
        if el != None:
            el.destroy()
        x = None
        e = '('
        for i in range(len(v[1])):
            e += ev[i].get() + ','
        e = e[:-1:]
        e += ')'
        for i in oxi_info:
            if i[0] == name:
                x = i[1]
                break
        ans = eval('f_p.'+x[:x.index('(')]+e)[1]
        if l_ans != None:
            l_ans.destroy()
        l_ans = Label(fram_run_fun,text = f"Answer is {v[0][:v[0].index('=')+1]}{ans}",font = ('Times New
Roman',20))
        l_ans.grid(row = u+2,column = 0)
    def check_run_fun():
        a = True
        z = 0
        global el
        global l_ans
```

```
        for i in ev:
            if len(i.get()) == 0:
                messagebox.showerror('sujection to error','enter all values')
                a = False
                break
            else:
                try:
                    float(i.get())
                except ValueError:
                    messagebox.showerror('sujection to error','enter all values with no alphabets or spectal
charactor')
                    a = False
                    break

        if a:
            find_ans()

    global fram_for
    global fram_run_fun
    fram_for.destroy()
    fram_run_fun = Frame(win)
    fram_run_fun.pack()

    for i in oxi_info:
        if i[0] == name:
            v = oxi.run_and_return_function_formula_var(i[1])
            break
    g = Label(fram_run_fun,text = 'Formula '+v[0],font = ('Times New Roman',20))
    g.grid(row = 0,column= 0)
    Label(fram_run_fun,text = '',font = ('Times New Roman',20),pady = 10).grid(row = 1,column= 0)
    ev = []
    s = 0
```

```
    u = 3
    for i in range(len(v[1])):
        ev.append(Entry(fram_run_fun,font = ('Times New Roman',20)))


    for i in ev:
        Label(fram_run_fun,text = f"enter the value of variable {v[1][s]}",font = ('Times New
Roman',20)).grid(row = u-1,column = 0)
        i.grid(row = u,column = 0)
        Label(fram_run_fun,text = '',font = ('Times New Roman',20),pady = 5).grid(row = u+1,column= 0)
        u += 3
        s += 1
    k = Button(fram_run_fun,text = 'calculate',command = check_run_fun,font = ('Times New Roman',20))
    k.grid(row = u+1,column= 0)
    Label(fram_run_fun,text = '',pady = 10).grid(row = u+2,column = 0)
    h = Button(fram_run_fun,text = 'Back',command = view_for,font = ('Times New Roman',20))
    h.grid(row = u+4,column= 0)


def intro():
    global fram_intro
    global fram_for
    global fram_gra_con
    global fram_
    global fram_run_fun

    if fram_for != None:
        fram_for.destroy()
    if fram_gra_con != None:
        fram_gra_con.destroy()
    if fram_add_f_g != None:
        fram_add_f_g.destroy()


    fram_intro = Frame(win)
```

```
    fram_intro.pack()

    Label(fram_intro,text = 'physics numerical solver'.upper(),font = ('Times New Roman',20,'bold'),fg =
'#F68C22').grid(row = 0,column = 1,pady = 100,padx = 30)

    Label(fram_intro,text = '',pady = 35).grid(row = 1,column = 1)

    but_for = Button(fram_intro,text = 'view formula to solve',command = view_for,font=('Times New
Roman',17),bg = 'light blue')

    but_for.grid(row = 2,column = 1,padx = 40,pady = 10)

    but_gr_ap = Button(fram_intro,text = 'graph and real life applications',command =
view_gra_con,font=('Times New Roman',17),bg = 'light blue')

    but_gr_ap.grid(row = 3,column = 1,padx = 40,pady = 10)

    but_hello = Button(fram_intro,text = 'Add formula',command = add_fun_gra,font=('Times New
Roman',17),bg = 'light blue')

    but_hello.grid(row = 4,column = 1,padx = 40,pady = 10)

    Label(fram_intro,text='').grid(row = 5,column = 1,pady=50)


def view_for():

    global n_list_v

    def abc():

        if list_box.get(ANCHOR) != '':

            run_fun(list_box.get(ANCHOR))

        else:

            messagebox.showerror('suggestion to error','select any formula to work')


    def sca():

        global n_list_v

        n_list_v = []

        nm = 0

        for i in oxi_info:

            list_box.delete(0,END)

            if len(s_e.get()) != 0:

                if s_e.get() in i[0]:

                    n_list_v.append((i[0]))

            else:
```

```
            n_list_v.append((i[0]))
        for i in n_list_v:
            list_box.insert(END,i)


    global v
    global fram_for
    global fram_run_fun
    v = ''
    fram_intro.destroy()
    if fram_run_fun != None:
        fram_run_fun.destroy()
    fram_for = Frame(win)
    fram_for.pack()
    Label(fram_for,text = 'View Formula',pady = 0,font = ('Times New Roman',20)).grid(row = 0,column =
1,pady = 10)
    Label(fram_for,text = 'Search Formula ',font = ('Times New Roman',15)).grid(row = 1,column = 0)
    s_e = Entry(fram_for,width=30,font = ('Times New Roman',13))
    s_e.grid(row = 1,column = 1)
    Button(fram_for,text = 'Search',command = sca,font = ('Times New Roman',13)).grid(row = 1,column
= 3)
    list_box = Listbox(fram_for,width = 80,height = 25,font = ('Times New Roman',10))
    list_box.grid(row = 2,column = 0,columnspan = 4)
    if len(n_list_v) == 0:
        n_list_v = oxi_info
    for i in n_list_v:
        list_box.insert(END,i[0])
    Button(fram_for,text = 'Select',command = abc,font = ('Times New Roman',13),padx = 20).grid(row =
3,column = 0)
    Button(fram_for,text = 'Back',command = intro,font = ('Times New Roman',13),padx = 30).grid(row =
4,column = 3)
    Label(fram_for,text = '',pady = 55).grid(row = 5,column = 0)


def graph(f):
```

```
    global d,list_box_gra,fram_graph
    def abcdefghij():
        global d
        s = '('
        for i in range(len(vg[0])):
            s += str(el[i].get()) + ','
        s = s[:-1]
        s += ')'
        d = d.replace(d[d.index('('):],s)
        eval('g_p.'+d)


    def check_run_graph():
        a = True
        for i in el:
            if len(i.get()) == 0:
                messagebox.showerror('suggestion to error','enter all values')
                a = False
                break
            else:
                try:
                    float(i.get())
                except ValueError:
                    messagebox.showerror('suggestion to error','enter all values without any alphabet or
special character')
                    a = False
                    break


        if a:
            abcdefghij()


    fram_gra_con.destroy()
    fram_graph = Frame(win)
```

```
    fram_graph.pack()
    vg = list()
    el = list()
    r = 1
    d = 0
    for i in oxi_gra_info:
        if i[0] == f:
            d = i[1]
            vg.append(i[1][i[1].index('(')+1:i[1].index(')')].split(','))
            vg.append(i)
    for i in range(len(vg[0])):
        Label(fram_graph,text = "",pady=10).grid(row = r-1,column = 0)
        Label(fram_graph,text = f"enter the value of {vg[0][i]}",font=('Times New
Roman',20),padx=100).grid(row = r,column = 0)
        a = Entry(fram_graph,font=('Times New Roman',20))
        a.grid(row = r+1,column = 0)
        el.append(a)
        r += 3
    Label(fram_graph,text = "",pady=20).grid(row = r-1,column = 0)
    Button(fram_graph,text = 'Draw Graph',command = check_run_graph,font=('Times New
Roman',15),padx = 30).grid(row = r,column = 0)
    Label(fram_graph,text = "",pady=20).grid(row = r+1,column = 0)
    Button(fram_graph,text = 'Back',command = view_gra_con,font=('Times New Roman',15),padx =
30).grid(row = r+2,column = 0)
    Label(fram_graph,text = "",pady=50).grid(row = r+3,column = 0)


def view_gra_con():
    def check_select_gra():
        if len(list_box_gra.get(ANCHOR)) != 0:
            graph(list_box_gra.get(ANCHOR))
        else:
            messagebox.showerror('sujection to error','select any graph')
```

```
    global fram_gra_con
    global list_box_gra
    if fram_intro != None:
        fram_intro.destroy()
    if fram_graph != None:
        fram_graph.destroy()
    fram_gra_con = Frame(win)
    fram_gra_con.pack()
    Label(fram_gra_con,text = 'View Graph',pady = 0,font=('Times New Roman',17)).grid(row = 0,column
= 1)
    list_box_gra = Listbox(fram_gra_con,width = 50,height = 17,font=('Times New Roman',15))
    list_box_gra.grid(row = 1,column = 0,columnspan = 4)
    for i in oxi_gra_info:
        list_box_gra.insert(END,i[0])
    Button(fram_gra_con,text = 'View Graph',font=('Times New Roman',13),padx=15,command =
check_select_gra).grid(row = 2,column = 0)
    Button(fram_gra_con,text = 'Back',command = intro,font=('Times New Roman',13),padx =
20).grid(row = 3,column = 3)
    Label(fram_gra_con,text='',pady = 50).grid(row = 4,column = 0)


def add_fun_gra():
    global fram_add_f_g
    if fram_intro != None:
        fram_intro.destroy()


    fram_add_f_g = Frame(win)
    fram_add_f_g.pack()
    Label(fram_add_f_g,text = 'in view -',pady = 0).grid(row = 0,column = 0)
    Button(fram_add_f_g,text = 'Add physic formula',font = ('Times New Roman',15)).grid(row =
1,column=0)
    Button(fram_add_f_g,text = 'Add graphical formula',font = ('Times New Roman',15)).grid(row =
2,column = 0)
    Button(fram_add_f_g,text = 'Back',command = intro,padx = 30,font = ('Times New
Roman',15)).grid(row = 4,column = 0)
```

```
intro()
win.mainloop()
```

project functions

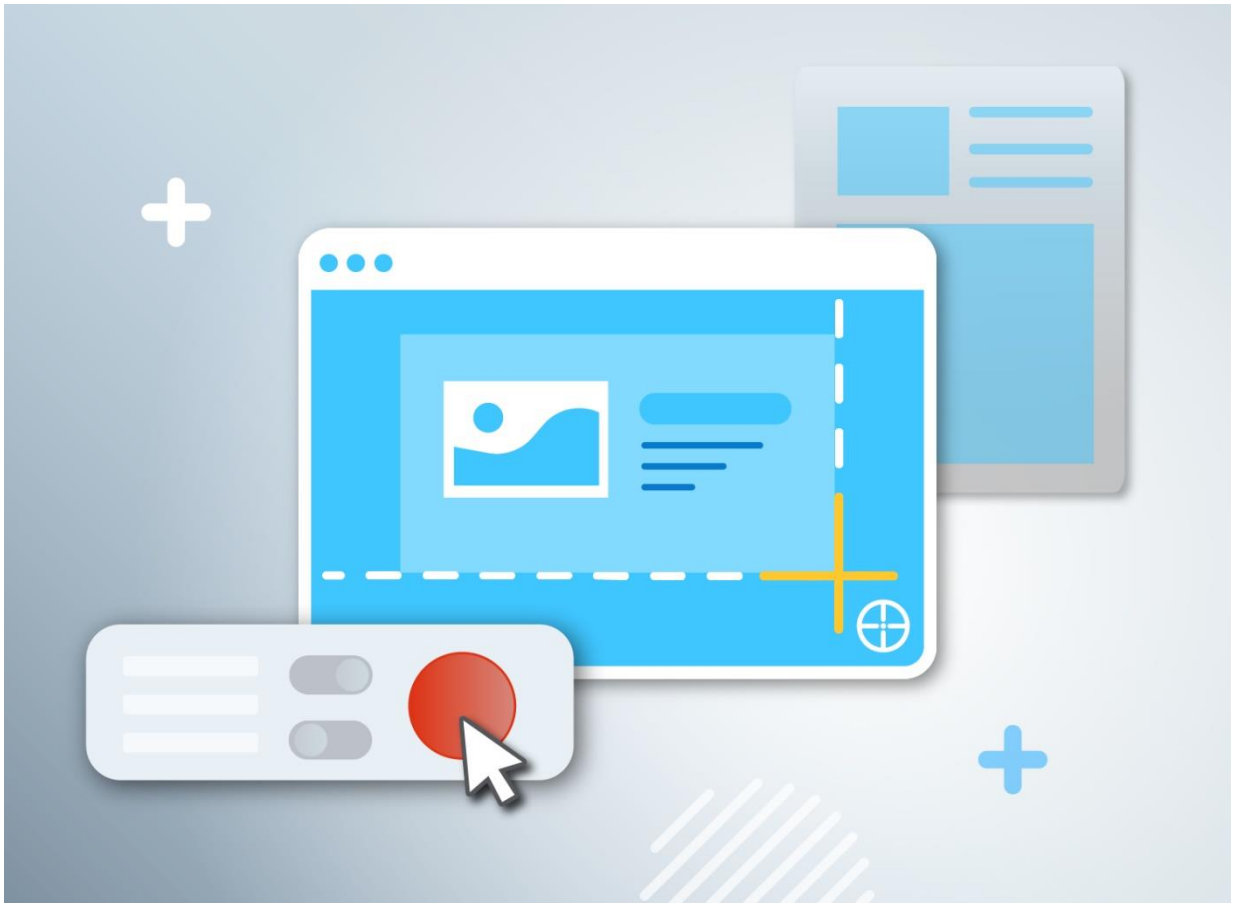
```
import math
from math import *
def quantization_of_charge__q(n):
    return("Q=",float(n)*(1.6)*(10**-19),"\nFormula:Q=±ne")
def quantization_of_charge_number_of_electrons_in_given_charge__n(Q):
    return("n=",float(Q)/(1.6)*(10**19),"\nFormula:n=Q/e")
def Coulombs_law_Force_between_two_charges__F(q1,q2,r):
    return("F=",float(q1)*float(q2)/((float(r)**2)*9)*(10**9),"N\nFormula:F=k*q1*q2/r**2")
def Coulombs_law_distance_between_two_charges__r(q1,q2,F):
    return("r=",math.sqrt((float(q2)*float(q2)*9)*10**9/float(F)),"m\nFormula:r=(q1*q2*k/F)**1/2")
def Coulombs_law_first_charge__q1(r,q2,F):
    return("q1=",float(r)**2*float(F)/(float(q2)*9*10**9),"C\nFormula:q1=F*r**2/q2*k")
def Coulombs_law_second_charge__q2(q1,r,F):
    return("q2=",float(r)**2*float(F)/(float(q1)*9*10**9),"C\nFormula:q2=F*r**2/q1*k")
def Relation_between_F_and_E__F(q,E):
    return("F=",float(q)*float(E),"N\nFormula:F=qE")
def Relation_between_E_anf_F_Electric_field__E(q,F):
    return("E=",float(F)/float(q),"v/m\nFormula:E=F/q")
def Relation_between_E_and_F_charge__q(E,F):
    return("q=",float(F)/float(E),"C\nFormula:q=F/E")
def Electric_field_due_to_a_point_charge__E(Q,r):
    return("E=",9*float(Q)/float(r)**2*(10**9),"v/m\nFormula:E=k*Q/r**2")
def Electric_field_due_to_a_point_charge_distance_between_charge_and_point__r(Q,E):
    return("r=",math.sqrt((9*10**9)*float(Q)/float(E)),"m\nFormula:r=(k*Q/E)**1/2")
```

```
def Electric_field_due_to_a_point_charge_charge__q(r,E):
    return("Q=",(float(E)*float(r)**2)/(9*10**9),"C\nFormula:Q=E*r**2/k")
def Gauss_theorm__phi(q):
    return("phi=",float(q)/8.854*(10**12),"\nFormula:phi=q/epsilon0")
def Gauss_theorem_charge__q(phi):
    return("q=",8.854*float(phi)*(10**-12)," C\nFormula:q=phi*epsilon0")
def Electric_potential_due_to_a_point_charge__V(q,r):
    return("V=",float(q)*9/float(r)*(10**9),"\nFormula:V=k*q/r")
def Electric_potential_due_to_a_point_charge_charge__q(v,r):
    return("q=",(float(v)*float(r))/(9*10**9),"C\nFormula:q=v*r/k")
def Electric_potential_due_to_a_point_charge_distance__r(q,v):
    return("r=",9*10**9*float(q)/float(v),"m\nFormula:r=k*q/V")
def Electric_potential_due_to_dipole__V(p,theta,r):
    return("v=",9*float(p)*math.cos(theta)/(float(r)**2)*10**9,"\nFormula:V=k*p*costheta/r**2")
def Electric_potential_due_to_dipole_distance_between_two_charges__r(p,theta,v):
    return("r=",math.sqrt((9*10**9)*(float(p)*math.cos(float(theta)))/float(v)),"m\nFormula:r=(k*p*cos
theta/v)**1/2")
def Electric_potential_due_to_dipole_dipole_moment__p(v,r,theta):
    return("p=",float(v)*float(r)**2/(9*10**9*math.cos(float(theta))),"Cm\nFormula:p=v*r**2/k*costhet
a")
def Electric_potential_due_to_dipole_angle__theta(v,p,r):
    return("theta=",math.acos(((float(v)*10**-
9)*(float(r)**2))/9*float(p)),"degree\nFormula:theta=cos**-1(v*r**2/k*p)")
def Potential_energy_of_a_system_of_two_point_charges__U(q1,q2,r):
    return("U=",9*float(q1)*float(q2)/float(r)*(10**9)," J\nFormula:U=k*q1*q2/r")
def
Potential_energy_of_a_system_of_two_point_charge_distance_between_two_point_charges__r(q1,q2,U)
:
    return("r=",(9*10**9)*float(q1)*float(q2)/float(U),"m\nFormula:r=k*q1*q2/U")
def Potential_energy_of_a_system_of_two_point_charge_first_charge__q1(U,r,q2):
    return("q1=",float(U)*float(r)/(9*10**9*float(q2)),"C\nFormula:q1=U*r/k*q2")
def Potential_energy_of_a_system_of_two_point_charge_second_charge__q2(q1,U,r):
    return("q2=",float(U)*float(r)/(9*10**9*float(q1)),"C\nFormula:q2=U*r/k*q1")
def Field_intensity_due_to_infinitely_long_straight_uniformly_charged_wire__E(lamda,R):
    return("E=",float(lamda)/(55.603*(10**-12)*float(R)),"v/m\nFormula:E=lamda/2epsilon0*R")
```

```
def Field_intensity_due_to_infinitely_long_straight_uniformly_charged_wire_Resistance__R(lamda,E):
    return("R=",float(lamda)/(55.603*(10**-12)*float(E)),"m\nR=lamda/2pieepsilon0*E")
def Field_intensity_due_to_infinitely_long_straight_uniformly_charged_wire_lamda_value__lamda(R,E):
    return("lamda=",float(R)*55.603*(10**-12)*float(E),"\nFormula:lamda=R*2pieepsilon0*E")
def Field_intensity_due_to_uniformly_charged_spherical_shell_out__E(q,r):
    return("E=",9*float(q)/(float(r)**2)*(10**9),"V/m\nFormula:E=k*q/r**2")
def Field_intensity_due_to_uniformly_charged_spherical_shell_on__E(q,R):
    return("E=",9*float(q)/(float(R)**2 )*(10**9),"V/m\nFormula:E=k*q/R**2")
def Field_intensity_due_to_uniformly_charged_spherical_shell_on_radius_of_shell__r(q,E):
    return("R=",math.sqrt(9*10**9*float(q))/math.sqrt(float(E)),"m\nFormula:R=√(k*q/E)")
def Field_intensity_due_to_uniformly_charged_spherical_shell_on_charge__q(E,R):
    return("q=",float(E)*float(R)**2/(9*10**9),"C\nFormula:q=ER**2/k")
def Field_intensity_due_to_thin_infinite_plane_sheet_of_charge__E(sigma):
    return("E=",float(sigma)/8.854*(10**12),"V/m\nFormula:E=sigma/epsilon0")
def Field_intensity_due_to_thin_infinite_plane_sheet_of_charge_conductivity__sigma(E):
    return("sigma=",float(E)*8.854*(10**-12),"ohm**-1\nFormula:sigma=E*epsilon0")
def Current_in_a_current_carrying_conductor__I(Q,t):
    return("I=",float(Q)/float(t),"A\nFormula:I=Q/t")
def Current_in_a_current_carrying_conductor_charge__Q(I,t):
    return("Q=",float(I)*float(t),"C\nFormula:Q=I*t")
def Current_in_a_current_carrying_conductor_time__t(Q,I):
    return("t=",float(Q)/float(I),"seconds\nFormula:t=Q/I")
def Ohms_law__V(I,R):
    return("V=",float(I)*float(R),"\nFormula:V=I*R")
def Ohms_law_Current__I(V,R):
    return("I=",float(V)/float(R),"A\nFormula:I=V/R")
def Magnetic_field_due_to_a_straight_conductor_of_infinite_length__B(N,I,r):
    return("B=",12.57*float(N)*float(I)/(float(r)*2*3.14)*(10**-7),"T\nFormula:B=μ0*N*I/2*pie*r")
def Magnetic_field_due_to_a_straight_conductor_of_infinite_length_number_of_turns__N(B,r,I):
    return("N=",float(B)*2*3.14*float(r)/(12.57*10**-7*float(I)),"\nFormula:N=B*2*pie*r/μ0*I")
def Magnetic_field_due_to_a_straight_conductor_of_infinite_length_current__I(N,B,r):
    return("I=",float(B)*2*3.14*float(r)/(12.57*10**-7*float(N)),"A\nFormula:I=B*2*pie*r/μ0*N")
```

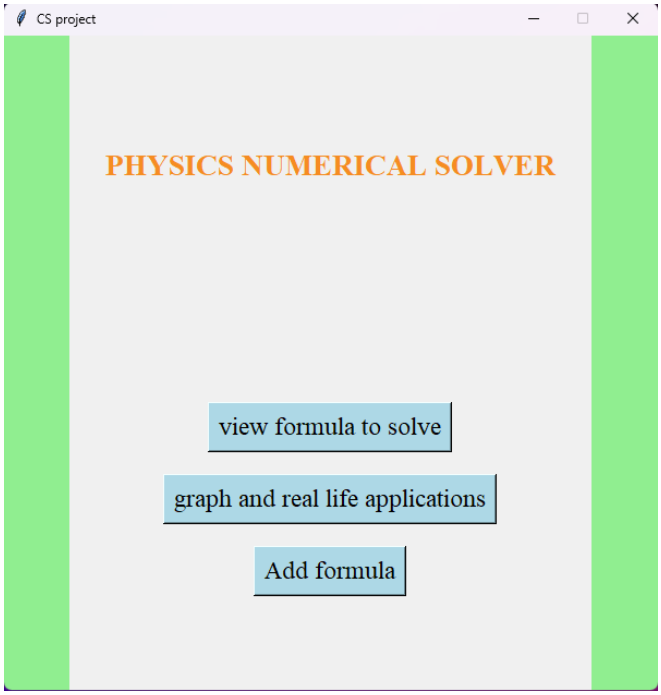
```
def find_velocity_of_light_in_second_medium_with_i_r_and_v_in_first_medium__v(v,i,r):
    return ('v =', v*sin(r)/sin(i),':v = v*sin(r)/sin(i)')
def critical_angle_in_TIR__i(a,s,i,n):
    return ('i =', asin(n),':i = asin(n)')
def radius_of_curvature_in_given_refractive_index_of_two_medium__r(N,n,u,v):
    return ('r=',(N-n)*u*v/(u*N-v*n),':r=(N-n)*u*v/(u*N-v*n)')
def focal_length_of_lens_in_given_distance_of_oject_and_image__f(u,v):
    return ('f=',u*v/(u-v),':f=u*v/(u-v)')
def distance_of_image_through_lens_in_given_focal_length_and_distance_of_object__v(u,f):
    return ('v=',u*f/(u+f),':v=u*f/(u+f)')
def distance_of_object_through_lens_in_given_focal_length_and_distance_of_image__u(v,f):
    return ('u=',v*f/(f-v),':u=v*f/(f-v)')
def angular_magnification_of_compound_microscope__m(d,f):
    return ('m=',1+d/f,':m=1+d/f')
def wave_length_of_light_in_medium_two_travelling_form_medium_one_to_two__L(V,v,l):
    return ('L=',V/v*l,':L=V/v*l')
def apparent_frequency_dopular_effect_in_given_velocites_of_observer_V_and_source_v__f(c,V,v,f):
    return ('f=',(c+V)/(c-v)*f,':f=(c+V)/(c-v)*f')
def distance_of_fringe_of_a_given_light__x(n,l,D,d):
    return ('x=',n*l*D/d,':x=n*l*D/d')
def resolving_power_of_lens__a(l,t):
    return ('a=',0.61*l/t,':a=0.61*l/t')
def wave_length_of_polarisation_of_light__l(k):
    return ('l=',2*pi/k,':l=2*pi/k')
def radius_of_orbit_at_nth_orbit__r(n):
    return ('r=',n**2*(1.097*10**7)**2*8.854187817*10**(-12)/(pi*(9.1093837*10**(-31))*(1.9*10**(-
19))**2),':r=n**2*(1.097*10**7)**2*8.854187817*10**(-12)/(pi*(9.1093837*10**(-31))*(1.9*10**(-
19))**2)')
def total_energy_of_electron_orbiting_at_nth_orbit__e(n):
    return ('e=',-2.18*10*(-19)/n**2,':e=-2.18*10*(-19)/n**2')
def energy_of_nuclear_binding_energy__e(m):
    return ('e=',m*(3*10**8)**2,':e=m*(3*10**8)**2')
def radioactivity_active_decay_at_given_time_t__n(N,l,t):
    return ('n=',N*e**(-l*t),':n=N*e**(-l*t)')
```

SCREENSHOTS

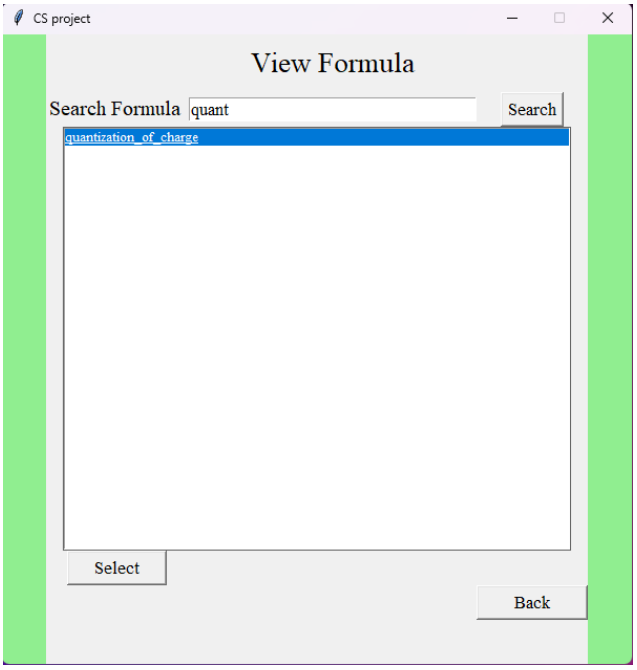


SCREENSHOTS

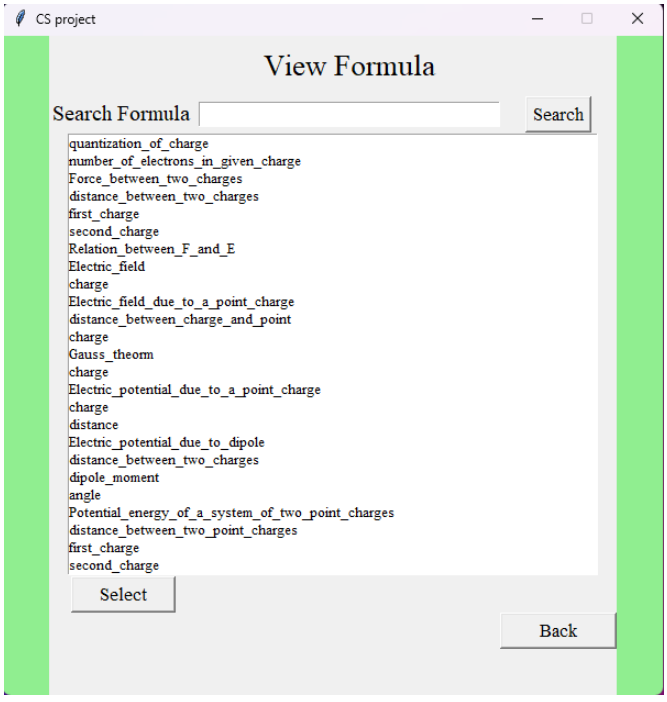
MAIN WINDOW



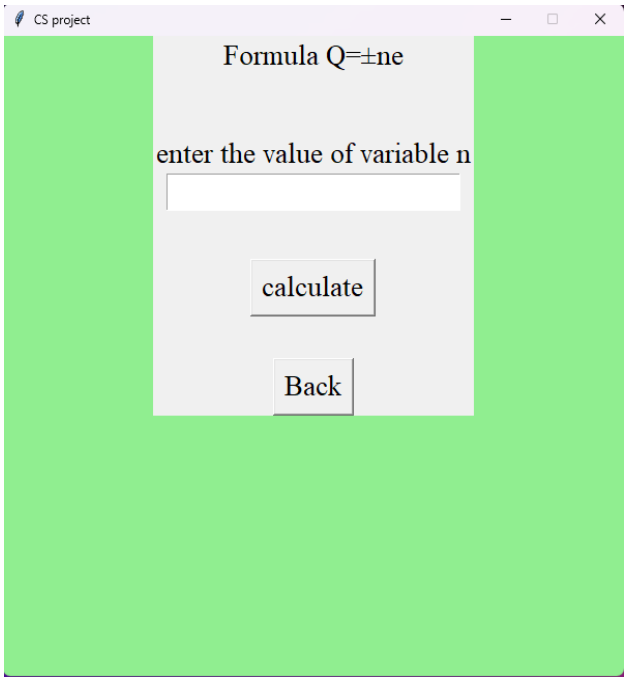
FORMULA SELECTION WINDOW



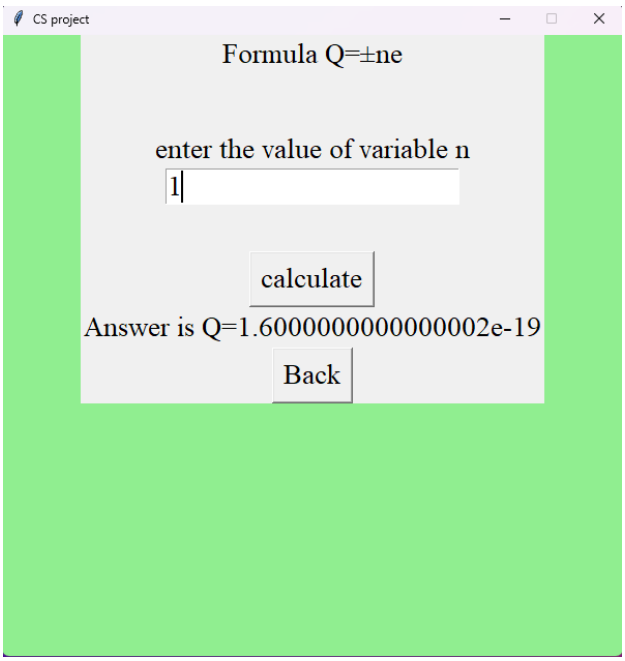
SEARCHING A FORMULA



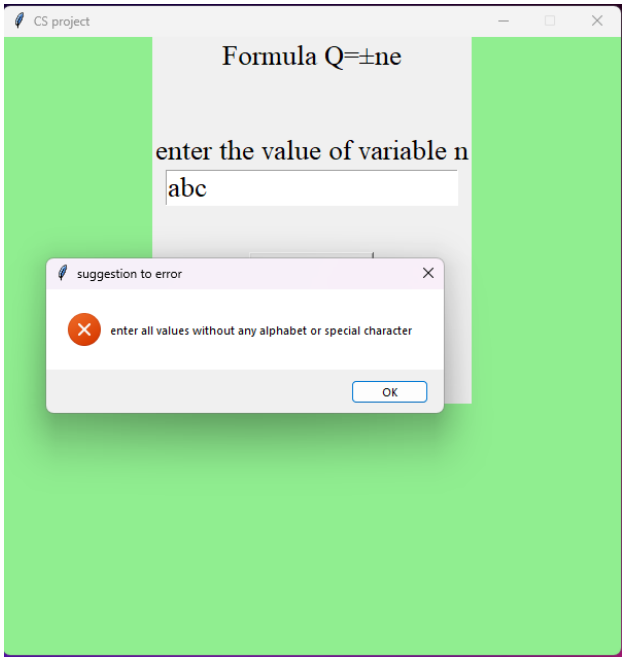
ENTERING THE VALUE OF A VARIABLE



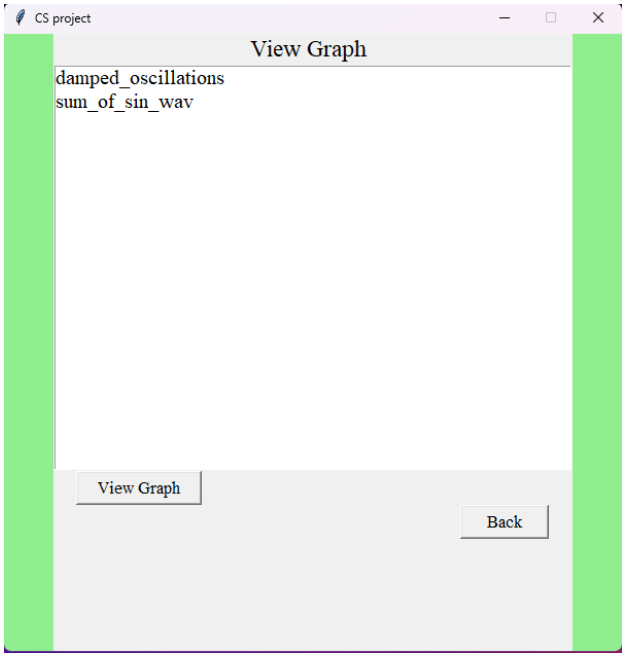
GETTING OUTPUT



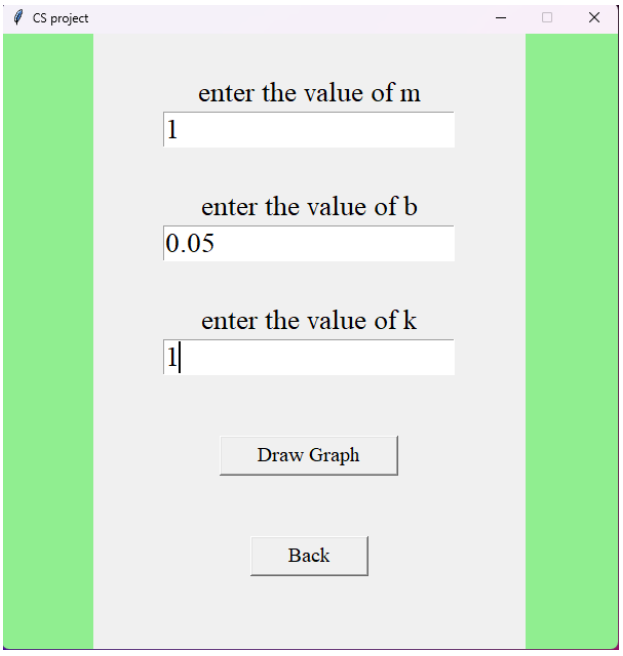
SUGGESTION FOR THE ERROR



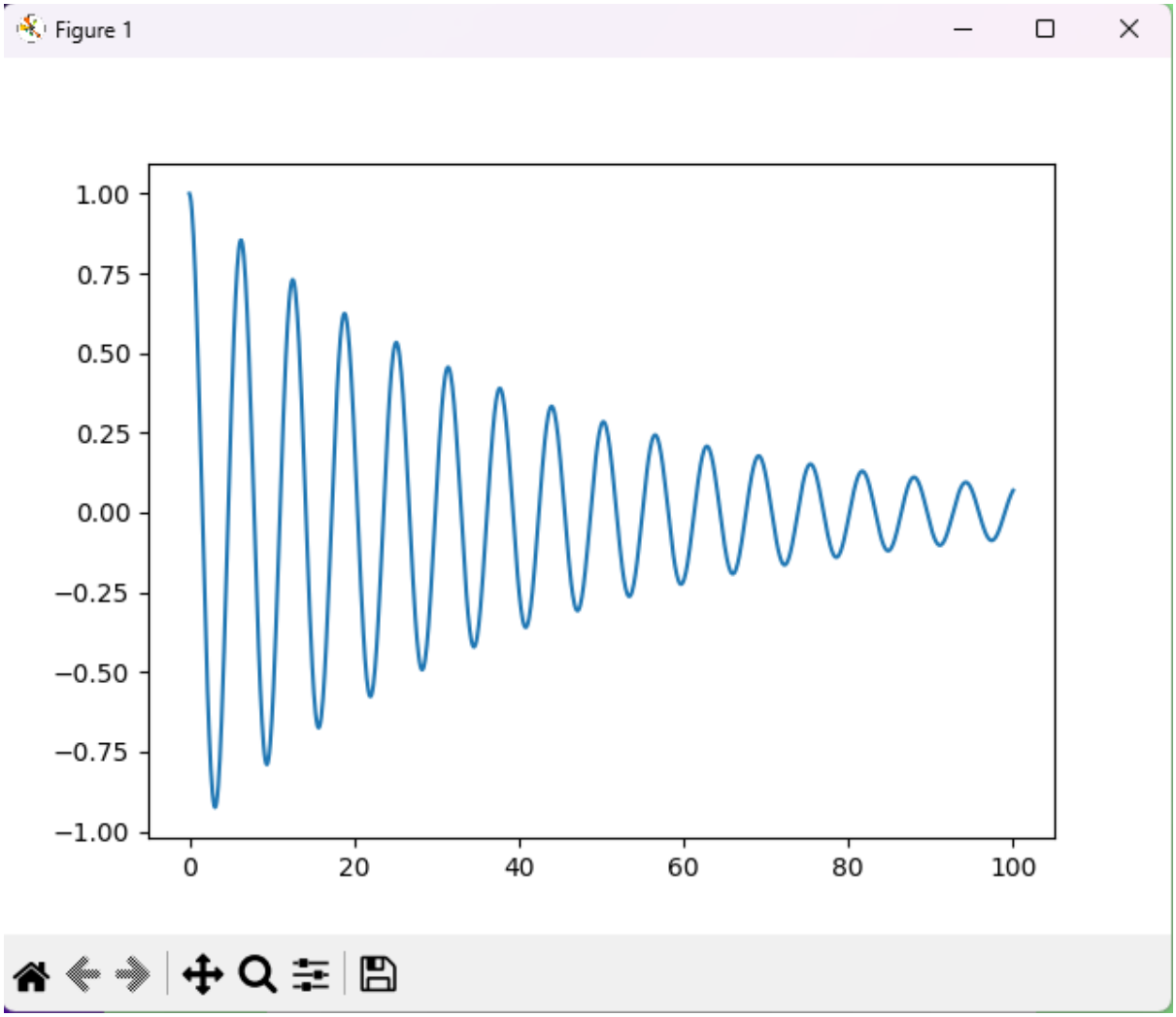
GRAPH SELECTION PAGE



ENTERING THE VALUE OF A VARIABLE



GETTING OUTPUT OF THE GRAPH



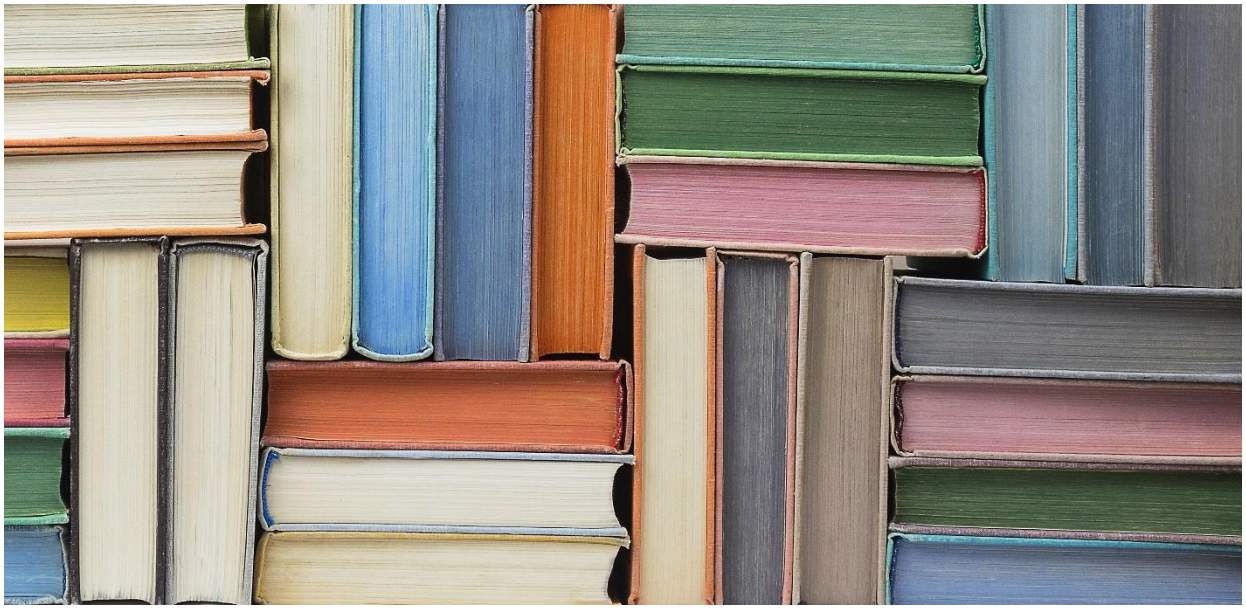
CONCLUSION

CONCLUSION

The project we made is of many hard work, so the chance of it rising an error is very less ,that is almost impossible.It had been run N mumber of times to find out the places which make the possibility for rising a error.And the arror rising parts of the code had been rectified,even if the user enters the improper value for the variable the code will not rise an error ,instead it rises a pop up error window instructing the user to enter the proper value for the variable of the selected formula.

Therefore, we conclude that the “PHYSICS NUMERICAL SOLVER” we made will be very much helpful for students who suffer a lot to solve the Physics numerical, and it will be helpful enough to make the student understand the answer by giving the appropriate formula for the selected concept. So, we made the code not only to find the answer but also to make the student understand the solution which helps them in solving the problems related to the same concept in the future.

BIBLIOGRAPHY



BIBLIOGRAPHY

- https://codemy.com/intro-tkinter-python-gui-apps/
- https://matplotlib.org/
- https://www.geeksforgeeks.org/
- https://www.pcmag.com/how-to/how-to-take-screenshots-in-windows-11
- https://ncerthelp.com/
- https://byjus.com/
- https://www.learncbse.in/