MORSE CODE CONVERTER

Section: K19AP
Registration Number: 11901629

Submitted by Aman Kumar

Submitted to Mrs. Pooja Rana

Project Description

Project Name: Morse Code Converter.

We have to create a converter which can convert morse code to text and text to morse code.

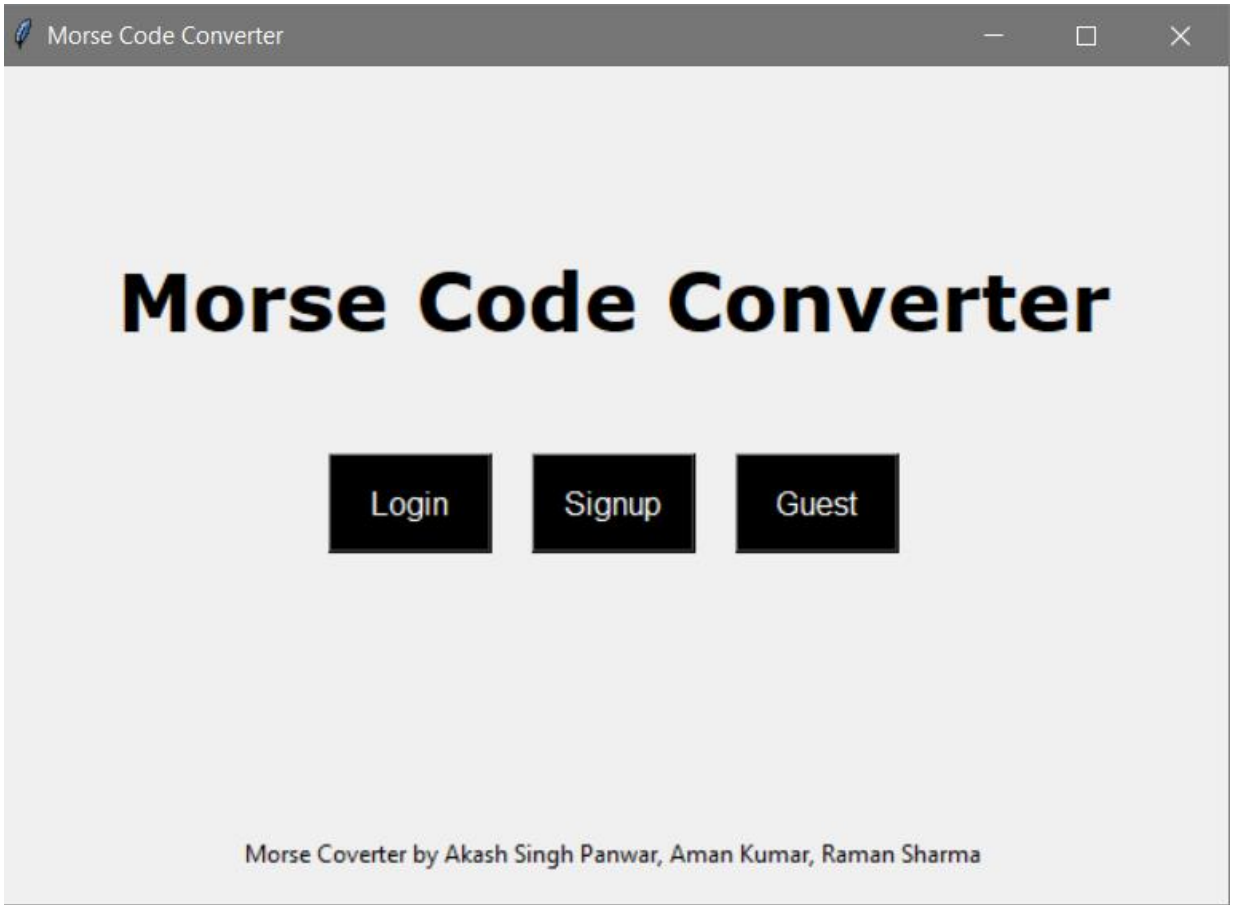
Project GitHub Link: https://github.com/RamanSharma100/morse-code-converter-python-project

My Contribution

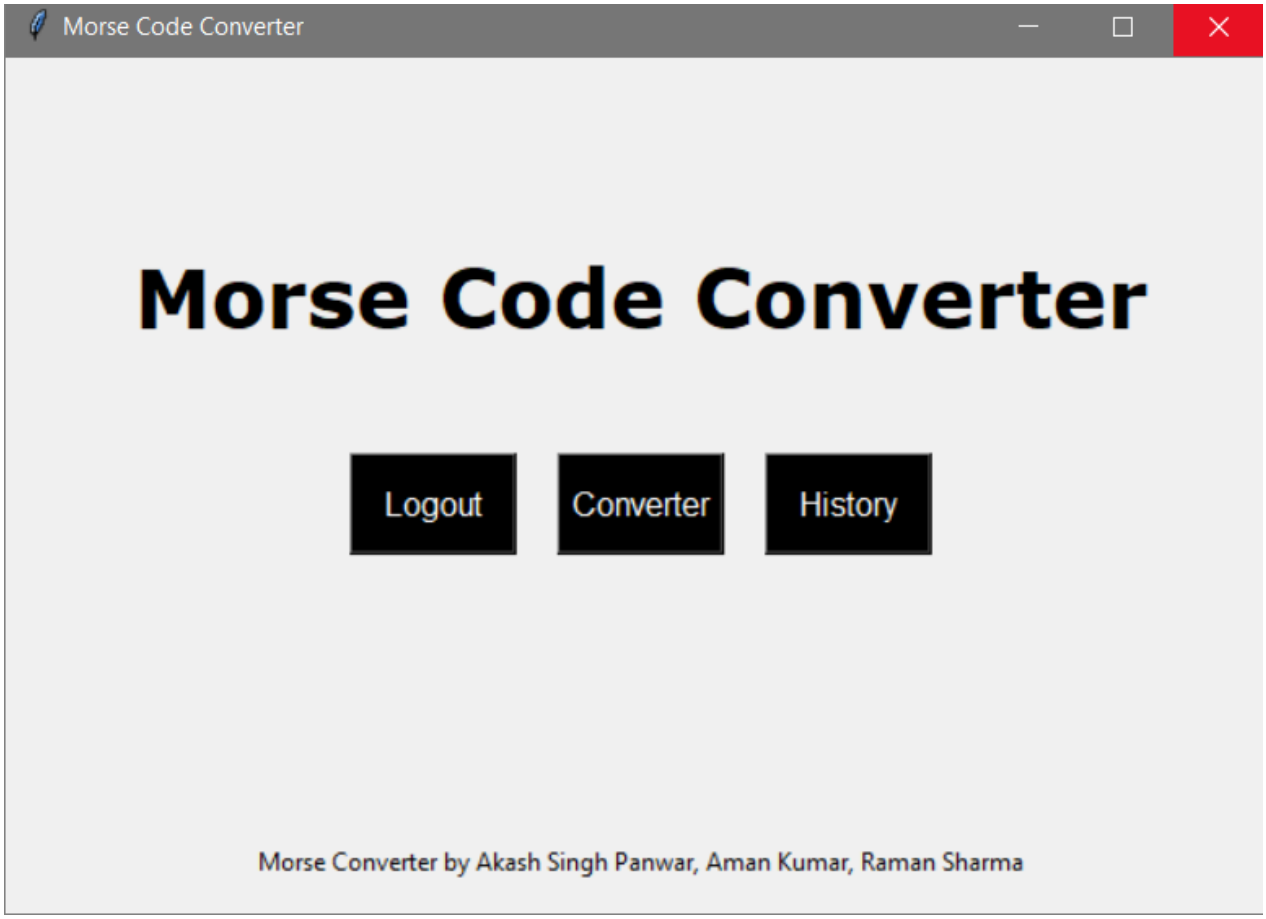
In this project I have contributed in creating

1. GUI of Index
2. GUI of Login with validation of login form with regex and database and GUI after user will logged in.
3. GUI of Morse to English with database as well as validation.

Page 1: Index



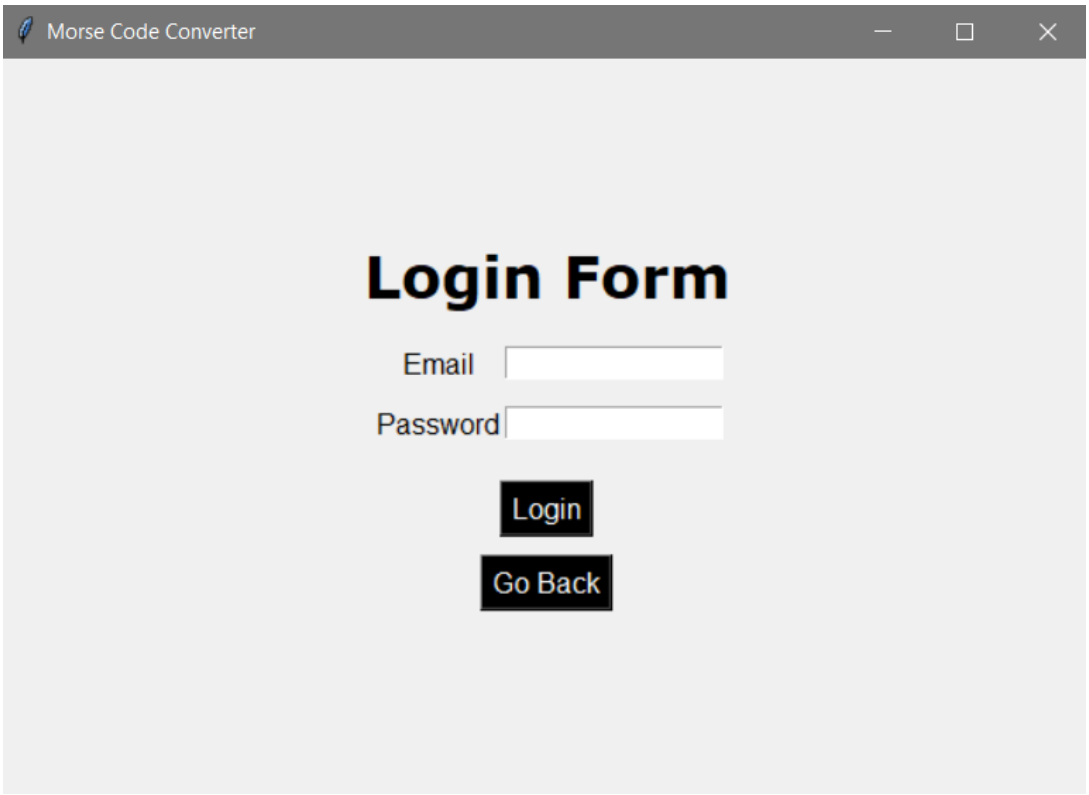
Index(Screen Shot)



Index after user logged in(Screen Shot)

As you can see above is the screen shot of index and index after user logged in. Index(screen Shot) GUI is the main window of our program whenever user will run the program this will appear first. And user will see these three buttons Login, signup and Guest. And when user logged in then Index after user logged in(Screen Shot) GUI will we displayed.

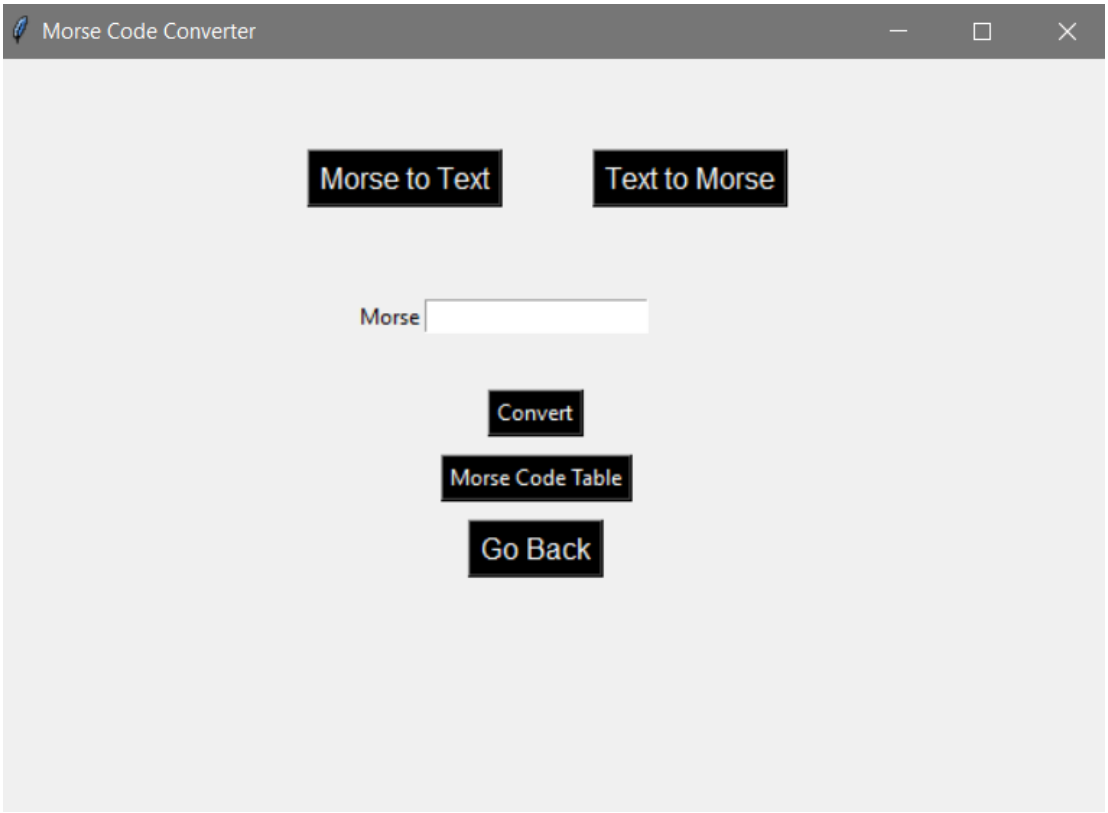
Page 2: Login



Login Page(Screen Shot)

Above screenshot is of Login Page. Here registered user can login to user morse converter as well to view the history of conversion. User have to sign up before they can login. In this page user have to fill their registered email they provided during sign up process and password. Here there is a **Go Back** button if user don't want to login and want go back then he/she can use this button.

Page 3: Morse to Text



Morse to text GUI (screen shot)

In this user will see converter page in which there is to button morse to text and text to morse. Its user wishes which conversion he wants to do. If he/she click on morse to text button an input field will appear with 3 more buttons **Convert, Morse Code Table** and **Go Back**. In input field when user write morse code and hit convert button then a text field will show up in which converted morse code will see.

Code Documentation

1. Index (def mainWindow())

```python
# main window
def mainWindow():

    global windowIndex
    global headFrame, head
    global buttonFrame, loginButton, signupButton, guestButton
    global bottomFrame
    global logoutButton,converterButton,historyButton

    windowIndex = Tk()
    windowIndex.geometry('620x420')
    windowIndex.maxsize(620,420)
    windowIndex.minsize(620,420)

    windowIndex.title('Morse Code Converter')

    headFrame = Frame()
    headFrame.grid(row = 0, column = 0)

    head = Label(windowIndex, text="Morse Code Converter", font = "Verdana 30 bold")
    head.grid(row = 0, column = 0, padx = 60, pady = (90,0))

    buttonFrame = Frame()
    buttonFrame.grid(row = 1, column = 0, pady = 40)

    # if the user is not logined then this will happen, logout button will only show up when user is loged in
    logoutButton = Button(buttonFrame, text='Logout', font = "15", height = 2, width = 8, bg = '#000000', fg = '#fff', command = logout)
    logoutButton.grid(row = 2, column = 0, padx = 10, pady = 10)
    # go to converter
    converterButton = Button(buttonFrame, text='Converter', font = "15", height = 2, width = 8, bg = '#000000', fg = '#fff', command = guest)
    converterButton.grid(row = 2, column = 1, padx = 10, pady = 10)
    # if user is logined he can watch history
    historyButton = Button(buttonFrame, text='History', font = "15", height = 2, width = 8, bg = '#000000', fg = '#fff', command = history)
    historyButton.grid(row = 2, column = 2, padx = 10, pady = 10)


    loginButton = Button(buttonFrame, text='Login', font = "15", height = 2, width = 8, bg = '#000000', fg = '#fff', command = loginForm)
    loginButton.grid(row = 2, column = 0, padx = 10, pady = 10)

    signupButton = Button(buttonFrame, text='Signup', font = "15", height = 2, width = 8, bg = '#000000', fg = '#fff', command = signupForm)
    signupButton.grid(row = 2, column = 1, padx = 10, pady = 10)

    guestButton = Button(buttonFrame, text='Guest', font = "15", height = 2, width = 8, bg = '#000000', fg = '#fff', command = guest)
    guestButton.grid(row = 2, column = 2, padx = 10, pady = 10)

    bottomFrame = Frame()
    bottomFrame.grid(row = 3, column = 0, pady = (90,0))

    foot = Label(bottomFrame, text = "Morse Coverter by Akash Singh Panwar, Aman Kumar, Raman Sharma")
    foot.grid(row = 3, column = 0)

    # when user will cross(X) button(show up in windows title at the top right side) then this function will work
    def on_closing():
        if messagebox.askokcancel("Quit", "Do you want to quit?"):
            windowIndex.destroy()

    windowIndex.protocol("WM_DELETE_WINDOW", on_closing)
    windowIndex.mainloop()

#when program will this function will be invoked
mainWindow()
```

Index Code snippet Screen Shot

First in this I declare global variables to use these variables anywhere in our program. Next I created object of Tkinter class named **windowIndex** and fix it geometry(dimension) and I also uses function **maxsize** and **minsize.** Here maxsize is used to fix the maximum size of window and minsize is used to fix minimum size of window. Then next, I give title **Morse Code Converter** to our window.

Then I created a frame **headFrame** for our head which will display in our window. Next I created label named **Morse Code Converter** having font style **verdana,** font size **30** and font weight **bold**. Now to show this I use grid and use **padx = 60** to give space around label in x-

axis as well as **pady = (90,0)** to give space around label in y-axis but here space will given from top not in bottom that's why its written as (90,0) means give 90pixel space from top and 0pixel in bottom.

Next I created another frame **buttonFrame** for our buttons. There is three buttons **Logout, Converter** and **History** these three buttons will be showed up only when user will login if user is not logged in then another three buttons will be showed those are **login, Signup** and **Guest.** Then I created frame **bottomFrame** for bottom label. Then created a label named **Morse Converter by Akash Singh Panwar, Aman Kumar, Raman Sharma** and place it at bottom.

Then I created a function named **on_closing** in this when user press X button to close the window a message box will popup asking for confirmation **Do you want to quit** is user will press ok or quit then the window will we destroyed and all process related to this window will also get destroyed. This function will be invoked using protocol method which handle deletion of top level windows.

There is also a statement **windowIndex.mainloop()** this line ensures that are window did not get close until and unless user wants to close.

At last I invoked my function **mainWindow()**.

2. Login (def loginForm())

```
# login form
def loginForm():

    head.grid_remove()
    loginButton.grid_remove()
    signupButton.grid_remove()
    guestButton.grid_remove()
    logoutButton.grid_remove()
    converterButton.grid_remove()
    historyButton.grid_remove()
    bottomFrame.grid_remove()

    #new login form frame created here

    loginHead = Label(windowIndex, text = "Login Form", font = "Verdana 25 bold")
    loginHead.grid(row = 0, column = 0, columnspan = 2, padx = 205, pady = (100,0))

    loginFrame = Frame()
    loginFrame.grid(row = 1, column = 0, columnspan = 2)

    usernamevalue = StringVar()
    passwordvalue = StringVar()

    username = Label(loginFrame, text = "Email", font = "12")
    username.grid(row = 1, column = 0, pady = (15,5))
    usernameEntry = Entry(loginFrame, textvariable = usernamevalue)
    usernameEntry.grid(row = 1, column = 1, pady = (15,5))

    password = Label(loginFrame, text = "Password", font = "12")
    password.grid(row = 2, column = 0, pady = 5)
    passwordEntry = Entry(loginFrame, show = "*", textvariable = passwordvalue)
    passwordEntry.grid(row = 2, column = 1, pady = 5)

    def goBack():
        loginHead.grid_remove()
        loginFrame.grid_remove()
        head.grid()
        loginButton.grid()
        signupButton.grid()
        guestButton.grid()
        bottomFrame.grid()
```

Login Code Snippet Part a.

head.grid_remove(), loginButton.grid_remove(), singupButton.grid_remove(), guestButton.grid_remove(), logout_button.grid_remove(), converterButton.grid_remove(), historyButton.grid_remove() and **bottomFrame.grid_remove()** in this **grid_remove()** widget. This removes the grid when login button on index page will be pressed.

grid_remove()

Remove this widget from the grid manager. The widget is not destroyed, and can be displayed again by **grid** or any other manager.

Then I created label named as **Login Form** and it act as a head for our login page displayed inside our window.

Next I created a new frame **loginFrame** for login form which contains input fields for email and passwords. These input fields are created using **Entry()** widget and showed and placed using grid.

Next there is function named **goBack()** in this **loginHead.grid_remove(), loginFrame.grid_remove()** these two lines remove head og login page and login form frame when go back button is pressed. Next **head.grid(), loginButton.grid(), signupButton.grid(), guestButton, bottomFrame.grid()** here **grid()** widget is used to display back those grid which have been removed.

```
# login user
def loginUser():
    global registered
    global loginedUserData
    # validate the user
    if(usernamevalue.get() == '' or passwordvalue.get() == ''):
        messagebox.showwarning('Warning','Please fill all the fields!!')
    else:
        errors = 0
        #validating email
        if(not re.search("^[a-z0-9]+[\._]?[a-z0-
        9]+[@]\w+[.]\w{2,3}$",usernamevalue.get().lower())):
            messagebox.showwarning("Warning", "Please enter valid email")
            errors = errors + 1
        if(errors == 0):
            # if there is no error connect to the database
            db = mysql.connector.connect(host='localhost',user='root',password='Akku2017@123',databa
            se='pythonproject')
```

```
        # checking if database is connected or not
        if(db):
            # creating database cursor
            myCursor = db.cursor()
            # creating database first if it is not created
            myCursor.execute("CREATE TABLE IF NOT EXISTS users(userId
            int(255) PRIMARY KEY AUTO_INCREMENT,name varchar(255) NOT
            NULL,email TINYTEXT NOT NULL,phonenumber int(11) NOT NULL,pass
            varchar(255) NOT NULL)")
            # checking if the email is already present or not
            emailCheckSql = r"SELECT * FROM users WHERE
            email='"+usernamevalue.get()+"'"
            myCursor.execute(emailCheckSql)
            result = myCursor.fetchone()
            if(myCursor.rowcount > 0):
                # here email is present then verify the user password
                if(passwordvalue.get() == result[-1]):
                    # setting login varible true
                    registered = True
                    # saving the logined user information to
                    loginedUserData variable
                    loginedUserData = result
                    myCursor.close()
                    db.close()



                    #send user back to home screen and show to logout and
                    converter and history button
                    loginHead.grid_remove()
                    loginFrame.grid_remove()
                    head.grid()
                    logoutButton.grid()
                    converterButton.grid()
                    historyButton.grid()
                    bottomFrame.grid()
                    messagebox.showinfo("Success","You are successfully
                    logged in!!")
                else:
                    myCursor.close()
                    db.close()
```

```
            messagebox.showwarning("Warning","Wrong password or
            email!!")
        else:
            myCursor.close()
            db.close()
            messagebox.showwarning("Warning","Please register first!!")
    else:
        messagebox.showwarning("Warning","Database is not connected!")
```

Login Code Snippet Part b.

Now inside **loginForm()** there is another function **loginUser()** in this first I declare I a global variables. Then next I wrote some condition related to login. First if username input field or password input field is empty then show of message will be a popup saying **Please fill in all the fields**. Next else declaring a variable named **error** and initializing it zero basically it keeps the counts of errors in else I put another if statement that will check whether user is entering valid email for this I used regex.

Python has a built-in package called *re* which can we used to work with regular expressions. First we imported *import re* this allows us to use regular expression. The *re* module offers set of function that allows us to search string for a match and we used one of the function **search().** The search function searches the string for a match and returns a match object if there is a match.

So, if validation is successful then next step execute but if user doesn't provide valid email then during validation it program knows it and display a popup saying **Please enter a valid email**.

Next condition if errors are zero then connect to our database. Next condition to check whether our database is connected or not. If database is connected then create database cursor then if database is not created then create it and if database is already created it will not create new database. Now next is for checking email is it present in our database or not.

Then if email is present then verify password and after verifying password I made the global variable **registered = True** and also saving logged in user data in another variable for futher use elsewhere in our program. Then after that close cursor user will displayed a page in which head **Morse Code Converter, logout button, converter button and history button** will be showed up and login form will be removed. And a popup message will be displayed **You are successfully logined.**

Now, If email is not present in a database then a popup message will appear **Please register first** and if user enters a wrong email or password then a popup message will appear **wrong email or password** and if database is not connected then a popup message will showed up **Database is not connected.**

In next statements I have created buttons **Login** and **Go Back** displayed in login form.

3. Morse to English (def Morse_to_English())

```
global tapmtoe
tapmtoe = True

global countMtoE
countMtoE += 1

if(tapetom == True):
    global countEtoM
    etomFrame.grid_remove()
    #updating countMtoE and making it zero to start over when english_to_morse was call tap-english-to-morse(tapetom) becomes true and this condition execute
    countEtoM = 0
#fix end

global mtoeFrame
if(countMtoE == 1):
    mtoeFrame = Frame(windowIndex)
    mtoeFrame.grid(row = 1, column = 0, columnspan = 2)

    label2 = Label(mtoeFrame, text = "Morse")
    label2.grid(row = 1, column = 0, pady = (50,0))

    label3 = Label(mtoeFrame, text = "Text")
    label3.grid(row = 2, column = 0, pady = (10,0))
    label3.grid_remove()

    entry = StringVar()
    get = StringVar()

    value = Entry(mtoeFrame, textvariable = entry)
    value.grid(row = 1, column = 1, pady = (50,0))

    getvalue = Entry(mtoeFrame, textvariable= get)####################################
    getvalue.grid(row = 2, column = 1, pady = (10,0))
    getvalue.grid_remove()

    Convert = Button(mtoeFrame, text = "Convert", bg = '#000000', fg = '#fff', command = convertion)
    Convert.grid( row  =  3, column = 1, pady = (30,0))

    def goBack():
        global countMtoE
        buttonFrame2.grid_remove()
        mtoeFrame.grid_remove()
        #updating countMtoE and making it zero to start over after go back is press
        countMtoE = 0 c
        if registered == True:
            head.grid()
            logoutButton.grid()
            converterButton.grid()
            historyButton.grid()
        else:
            head.grid()
            loginButton.grid()
            signupButton.grid()
            guestButton.grid()
            bottomFrame.grid()

    gobackButton = Button(mtoeFrame, text = "Go Back", font = "10", bg = '#000000', fg = '#fff', command = goBack)
    gobackButton.grid(row = 5, column = 1, pady = (10,0))

    code = Button(mtoeFrame, text = "Morse Code Table", bg = '#000000', fg = '#fff', command = morseCodeWin)
    code.grid(row = 4, column = 1, pady = (10,0))
```

morse to English GUI code snippet (screen shot)

So here a global variable is declared **tapmtoe** and it is a Boolean having value **True.** Next another global variable is declared **countMtoE** this variable basically keep count by incrementing its value by one that how many time this function is invoked.

Now next there is condition having if statement which is checking whether **tapetom is equal to True or not**. This tapetom variable is declared in English to morse function.

It basically checks that tapetom is true or not. Tapetom will be true when English to morse function will be invoked so when English to morse function invoked it execute all the statement written inside and display its gui. So to remove that gui I wrote this condition that if tapetom is true then remove etomframe and reset countEtoM. So that our morse to English gui displays correctly without overlapping.

Next I declare a global variable **mtoeFrame**. Now then I write a condition if countMtoE is equal to 1 then execute the inside statement. I do this to stop overlapping of frames if user will click **morse to text** button more than one time. Now inside this condition a new frame is created and assigned to **mtoeFrame** variable. Next I created two labels named as **morse** and **text**, and two input fields using **Entry()** widget . Here then I use **grid_remove()** to remove the **label3** it showed up when convert button is pressed.

Then three buttons are created named **Convert, Go Back** and **Morse code Table.**

Also inside morse to English there is an another function named **goBack().** This function is connected to go back button. So, when we press go back button it removes **buttonFrames2** and **mtoeFrame.** Then it updates the **countMtoE** to zero. Next there is condition if **registered is True** then show **head, logout button, converter button** and **history button.** And if not true then show **head, login button, sign up button, guest button** and **bottom frame** (in short display index window)**.**