



LAXMI CHARITABLE TRUST'S
SHETH L.U.J. COLLEGE OF ARTS & SIR M.V. COLLEGE OF
SCIENCE & COMMERCE
DR. S. RADHAKRISHNAN MARG, ANDHERI(EAST), MUMBAI - 400069

Department of Computer Science

Academic Year

2025-2026

A GUI-based Python application that allows users to track their learning progress by adding, updating, and managing courses.

ONLINE LEARNING TRACKING

Name:Gupta Trisha Dinesh

Class: FYBSC.CS

Roll no:F085

TECHNOLOGY USED :Tkinter ,SQLite, OOPs,Exception handling.

SUBJECT NAME: ADVANCED PYTHON PROGRAMMING

SEMESTER: Semester-II

SUBJECT INCHARGE: DR.MAHENDRA KANOJIA

& Prof:ROHIT SAHU

CODE

```
print("PYTHON PROJECT")
print("WELCOME TO TRISHA GUPTA F085 ONLINE LEARNING TRACKING.....!!!")

# IMPORTS
import tkinter as tk
from tkinter import messagebox
import sqlite3

#LOGIN FORM
def login():
    username = entry_username.get()
    password = entry_password.get()

    if username == "admin" and password == "1234":
        messagebox.showinfo("Login Successful", "Welcome! Login successful.")
    else:
        messagebox.showerror("Login Failed", "Invalid Username or Password")

# Create Window
root = tk.Tk()
root.title("Login Form")
root.geometry("350x250")
root.configure(bg="#E6E6FA"># Lavender color

# Title
tk.Label(root, text="Login System", font=("Arial", 16, "bold"), bg="#E6E6FA").pack(pady=10)

# Username
tk.Label(root, text="Username", font=("Arial", 12), bg="#E6E6FA").pack(pady=5)
entry_username = tk.Entry(root)
entry_username.pack(pady=5)

# Password
tk.Label(root, text="Password", font=("Arial", 12), bg="#E6E6FA").pack(pady=5)
entry_password = tk.Entry(root, show="*")
entry_password.pack(pady=5)

# Login Button
tk.Button(root, text="Login", command=login, bg="#BA55D3", fg="white", font=("Arial", 12)).pack(pady=15)
root.mainloop()
```

```
# Now start a main part of body(project)
```

```
# CourseDatabase class
```

```
class CourseDatabase:
```

```
    # INITIALIZING DATABASE
```

```
    def __init__(self):
```

```
        self.conn = sqlite3.connect("learning_tracker.db", check_same_thread=False)
```

```
        self.cursor = self.conn.cursor()
```

```
    # DELETING ANY OLD DATA
```

```
    self.cursor.execute("DROP TABLE IF EXISTS courses")
```

```
    # CREATING NEW TABLE COURSES
```

```
    self.cursor.execute(
```

```
        "CREATE TABLE courses (id INTEGER PRIMARY KEY AUTOINCREMENT, title TEXT,  
provider TEXT, duration INTEGER, status TEXT DEFAULT 'Not Started')"
```

```
    )
```

```
    self.conn.commit()
```

```
# ADDING COURSE TO DATABASE
```

```
def add_course(self, title, provider, duration):
```

```
    self.cursor.execute(
```

```
        "INSERT INTO courses (title, provider, duration) VALUES (?, ?, ?)",
```

```
        (title, provider, duration),
```

```
    )
```

```
    self.conn.commit()
```

```
# FETCHING ALL COURSES
```

```
def get_courses(self):
```

```
    self.cursor.execute("SELECT * FROM courses")
```

```
    return self.cursor.fetchall()
```

```
# MARK A COURSE AS COMPLETED
```

```
def complete_course(self, course_id):
```

```
    self.cursor.execute(
```

```
        "UPDATE courses SET status = 'Completed' WHERE id = ?",
```

```
        (course_id,),
```

```
    )
```

```
    self.conn.commit()
```

```
# DELETE SELECTED COURSE
```

```
def remove_course(self, course_id):
```

```
    self.cursor.execute("DELETE FROM courses WHERE id = ?", (course_id,))
```

```

        self.conn.commit()

# DELETE ALL COURSES
def remove_all_courses(self):
    self.cursor.execute("DELETE FROM courses")
    self.conn.commit()

# Online Learning Tracker App Class
class LearningTrackerApp:
    # INITIALIZING THE GUI
    def __init__(self, master):
        self.db = CourseDatabase()
        master.title("Learning Tracker")
        master.configure(bg="#E6E6FA")

        self.course_title = tk.StringVar()
        self.provider = tk.StringVar()
        self.duration = tk.StringVar()
        self.course_id = tk.StringVar()

        for label, var in [
            ("Course Title:", self.course_title),
            ("Source:", self.provider),
            ("Duration (hrs):", self.duration),
        ]:
            tk.Label(
                master,
                text=label,
                bg="#D8BFD8",
                fg="black",
                font=("Arial", 12, "bold"),
            ).pack(pady=5)
            tk.Entry(
                master,
                textvariable=var,
                bg="#EEE8AA",
                fg="black",
                font=("Arial", 12),
            ).pack(pady=5)

        button_bg = "#BA55D3"
        button_fg = "white"

```

```
tk.Button(  
    master,  
    text="Add a Course",  
    command=self.add_course,  
    bg=button_bg,  
    fg=button_fg,  
    font=("Arial", 12, "bold"),  
).pack(pady=5)
```

```
tk.Button(  
    master,  
    text="Show all Courses",  
    command=self.show_courses,  
    bg=button_bg,  
    fg=button_fg,  
    font=("Arial", 12, "bold"),  
).pack(pady=5)
```

```
self.output_text = tk.Text(  
    master,  
    height=10,  
    width=50,  
    bg="#EEE8AA",  
    fg="black",  
    font=("Arial", 12),  
)  
self.output_text.pack(pady=10)
```

```
tk.Label(  
    master,  
    text="Course ID:",  
    bg="#D8BFD8",  
    fg="black",  
    font=("Arial", 12, "bold"),  
).pack(pady=5)
```

```
tk.Entry(  
    master,  
    textvariable=self.course_id,  
    bg="#EEE8AA",  
    fg="black",  
    font=("Arial", 12),  
).pack(pady=5)
```

```

tk.Button(
    master,
    text="Mark as Completed",
    command=self.complete_course,
    bg=button_bg,
    fg=button_fg,
    font=("Arial", 12, "bold"),
).pack(pady=5)

```

```

tk.Button(
    master,
    text="Delete Selected ID Course",
    command=self.delete_course,
    bg=button_bg,
    fg=button_fg,
    font=("Arial", 12, "bold"),
).pack(pady=5)

```

```

tk.Button(
    master,
    text="Delete All Courses",
    command=self.delete_all_courses,
    bg=button_bg,
    fg=button_fg,
    font=("Arial", 12, "bold"),
).pack(pady=5)

```

ADDING A COURSE

```

def add_course(self):
    if not self.course_title.get() or not self.provider.get() or not self.duration.get().isdigit():
        messagebox.showerror("Error", "Invalid input! Please enter valid course details.")
        return

    self.db.add_course(
        self.course_title.get(),
        self.provider.get(),
        int(self.duration.get()),
    )
    messagebox.showinfo("Success", "Course Has Been Added Successfully...!!!")
    self.clear_inputs()

```

SHOW ALL COURSES

```

def show_courses(self):
    self.output_text.delete(1.0, tk.END)

```

```

for course in self.db.get_courses():
    self.output_text.insert(
        tk.END,
        f"{course[0]} | {course[1]} | {course[2]} | {course[3]} hrs | {course[4]}\n",
    )

# MARK COURSE AS COMPLETED
def complete_course(self):
    if not self.course_id.get().isdigit():
        messagebox.showerror("Error", "Oh...!!Invalid Course ID! Please enter a numeric ID.")
        return

    self.db.complete_course(int(self.course_id.get()))
    messagebox.showinfo("Success", "Course Has Been Marked As Completed Successfully...!!!")
    self.show_courses()

# DELETE SELECTED COURSE
def delete_course(self):
    if not self.course_id.get().isdigit():
        messagebox.showerror("Error", "Oops...!!Invalid Course ID! Please enter a numeric ID.")
        return

    self.db.remove_course(int(self.course_id.get()))
    messagebox.showinfo("Success", "Course Has Been Deleted Successfully...!!!")
    self.show_courses()

# DELETE ALL COURSES
def delete_all_courses(self):
    if messagebox.askyesno("Confirm", "Are you sure you want to delete all courses..???"):
        self.db.remove_all_courses()
        messagebox.showinfo("Success", "All Courses Deleted Successfully...!!!")

def clear_inputs(self):
    self.course_title.set("")
    self.provider.set("")
    self.duration.set("")
    self.course_id.set("")

if __name__ == "__main__":
    root = tk.Tk()
    LearningTrackerApp(root)
    root.mainloop()

```

```
Python project.py - C:\Users\De\l\Desktop\Python Project\Python project.py (3.13.5)
File Edit Format Run Options Window Help
print("PYTHON PROJECT")
print("WELCOME TO TRISHA GUPTA F085 ONLINE LEARNING TRACKING....!!!")

import tkinter as tk
from tkinter import messagebox, ttk
import sqlite3

#DATABASE
class CourseDatabase:
    def __init__(self):
        self.conn = sqlite3.connect("learning_tracker.db")
        self.cursor = self.conn.cursor()
        self.cursor.execute("""
            CREATE TABLE IF NOT EXISTS courses(
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                title TEXT,
                provider TEXT,
                duration INTEGER,
                status TEXT DEFAULT 'Not Started')
            """)
        self.conn.commit()

    def execute(self, query, data=()):
        self.cursor.execute(query, data)
        self.conn.commit()

    def fetch(self, query, data=()):
        self.cursor.execute(query, data)
        return self.cursor.fetchall()

    def progress_stats(self):
        total = self.fetch("SELECT SUM(duration) FROM courses")[0][0]
        completed = self.fetch(
            "SELECT SUM(duration) FROM courses WHERE status='Completed'"
        )[0][0]
        return total or 0, completed or 0

# LOGIN
class LoginSystem:
    def __init__(self, root):
        root.title("Login System")
        root.geometry("350x250")
        root.configure(bg="#E6E6FA")

        tk.Label(root, text="Login System",
            font=("Arial", 16, "bold"),
            bg="#E6E6FA").pack(pady=20)

        self.username = tk.Entry(root)
        self.password = tk.Entry(root, show="*")

        tk.Label(root, text="Username", bg="#E6E6FA").pack()
        self.username.pack()

        tk.Label(root, text="Password", bg="#E6E6FA").pack()
        self.password.pack()

        tk.Button(root, text="Login",
            bg="#BA55D3", fg="white",
            command=self.login).pack(pady=15)

        self.root = root

    def login(self):
        if self.username.get() == "Trisha" and self.password.get() == "1234":
            messagebox.showinfo("Success", "Login Successful!")
            self.root.destroy()
            open_main_app()
        else:
            messagebox.showerror("Error", "Invalid Username or Password")

# MAIN APP
class LearningTrackerApp:
    def __init__(self, root):
        self.db = CourseDatabase()
        self.root = root
        root.title("Online Learning Tracker Dashboard")
        root.geometry("900x600")
        root.configure(bg="#E6E6FA")
```

```
Python project.py - C:\Users\De\l\Desktop\Python Project\Python project.py (3.13.5)
File Edit Format Run Options Window Help
class LoginSystem:
    def __init__(self, root):
        root.title("Login System")
        root.geometry("350x250")
        root.configure(bg="#E6E6FA")

        tk.Label(root, text="Login System",
            font=("Arial", 16, "bold"),
            bg="#E6E6FA").pack(pady=20)

        self.username = tk.Entry(root)
        self.password = tk.Entry(root, show="*")

        tk.Label(root, text="Username", bg="#E6E6FA").pack()
        self.username.pack()

        tk.Label(root, text="Password", bg="#E6E6FA").pack()
        self.password.pack()

        tk.Button(root, text="Login",
            bg="#BA55D3", fg="white",
            command=self.login).pack(pady=15)

        self.root = root

    def login(self):
        if self.username.get() == "Trisha" and self.password.get() == "1234":
            messagebox.showinfo("Success", "Login Successful!")
            self.root.destroy()
            open_main_app()
        else:
            messagebox.showerror("Error", "Invalid Username or Password")

# MAIN APP
class LearningTrackerApp:
    def __init__(self, root):
        self.db = CourseDatabase()
        self.root = root
        root.title("Online Learning Tracker Dashboard")
        root.geometry("900x600")
        root.configure(bg="#E6E6FA")
```

```
Python project.py - C:\Users\De\l\Desktop\Python Project\Python project.py (3.13.5)
File Edit Format Run Options Window Help
root.geometry("900x600")
root.configure(bg="#E6E6FA")

tk.Label(root, text="Online Learning Tracker",
font=("Arial", 18, "bold"),
bg="#E6E6FA").pack(pady=10)

# Variables
self.title = tk.StringVar()
self.provider = tk.StringVar()
self.duration = tk.StringVar()
self.course_id = tk.StringVar()
self.search = tk.StringVar()

frame = tk.Frame(root, bg="#E6E6FA")
frame.pack()

self.input_box(frame, "Course Title", self.title, 0)
self.input_box(frame, "Provider", self.provider, 1)
self.input_box(frame, "Duration (hrs)", self.duration, 2)

tk.Button(frame, text="Add Course",
bg="#BA55D3", fg="white",
command=self.add_course).grid(row=3, column=0)

tk.Button(frame, text="Update Course",
bg="#BA55D3", fg="white",
command=self.update_course).grid(row=3, column=1)

tk.Button(frame, text="Show All",
bg="#BA55D3", fg="white",
command=self.show_courses).grid(row=3, column=2)

tk.Entry(frame, textvariable=self.search).grid(row=4, column=1)
tk.Button(frame, text="Search",
command=self.search_course).grid(row=4, column=2)

# Table
cols = ("ID", "Title", "Provider", "Duration", "Status")
self.tree = ttk.Treeview(root, columns=cols, show="headings")

Ln: 22 Col: 0
```

```
Python project.py - C:\Users\De\l\Desktop\Python Project\Python project.py (3.13.5)
File Edit Format Run Options Window Help
command=self.search_course).grid(row=4, column=2)

# Table
cols = ("ID", "Title", "Provider", "Duration", "Status")
self.tree = ttk.Treeview(root, columns=cols, show="headings")

for c in cols:
    self.tree.heading(c, text=c)

self.tree.pack(fill="both", expand=True, pady=20)

self.tree.bind("<ButtonRelease-1>", self.select_course)

bottom = tk.Frame(root, bg="#E6E6FA")
bottom.pack()

tk.Entry(bottom, textvariable=self.course_id, width=8).grid(row=0, column=0)

tk.Button(bottom, text="Start Course",
command=self.start_course).grid(row=0, column=1)

tk.Button(bottom, text="Complete",
command=self.complete_course).grid(row=0, column=2)

tk.Button(bottom, text="Delete",
command=self.delete_course).grid(row=0, column=3)

tk.Button(bottom, text="Delete All",
bg="red", fg="white",
command=self.delete_all).grid(row=0, column=4)

tk.Button(bottom, text="Logout",
bg="black", fg="white",
command=self.logout).grid(row=0, column=5)

self.progress = ttk.Progressbar(root, length=500)
self.progress.pack(pady=10)

self.show_courses()
self.update_progress()

Ln: 22 Col: 0
```

```
Python project.py - C:\Users\De\l\Desktop\Python Project\Python project.py (3.13.5)
File Edit Format Run Options Window Help

def show_courses(self):
    self.tree.delete(*self.tree.get_children())
    for row in self.db.fetch("SELECT * FROM courses"):
        status = self.format_status(row[4])
        new_row = (row[0], row[1], row[2], row[3], status)
        self.tree.insert("", tk.END, values=new_row)

def search_course(self):
    data = self.db.fetch(
        "SELECT * FROM courses WHERE title LIKE ?",
        ('%' + self.search.get() + '%',)
    )
    self.tree.delete(*self.tree.get_children())
    for row in data:
        status = self.format_status(row[4])
        new_row = (row[0], row[1], row[2], row[3], status)
        self.tree.insert("", tk.END, values=new_row)

def update_course(self):
    if not self.course_id.get().isdigit():
        return messagebox.showerror("Error", "Enter valid Course ID")

    self.db.execute(
        "UPDATE courses SET title=?,provider=?,duration=? WHERE id=?",
        (self.title.get(), self.provider.get(),
         int(self.duration.get()), int(self.course_id.get()))
    )
    messagebox.showinfo("Success", "Course Updated")
    self.clear()

def start_course(self):
    self.db.execute(
        "UPDATE courses SET status='In Progress' WHERE id=?",
        (self.course_id.get(),)
    )
    messagebox.showinfo("Started", "Course In Progress")
    self.refresh()

def complete_course(self):
    self.db.execute(
        "UPDATE courses SET status='Completed' WHERE id=?",
        (self.course_id.get(),)
    )

Ln: 22 Col: 0
```

```
Python project.py - C:\Users\De\l\Desktop\Python Project\Python project.py (3.13.5)
File Edit Format Run Options Window Help

def delete_all(self):
    if messagebox.askyesno("Confirm", "Delete all courses?"):
        self.db.execute("DELETE FROM courses")
        self.refresh()

def refresh(self):
    self.show_courses()
    self.update_progress()

def update_progress(self):
    total, completed = self.db.progress_stats()
    self.progress["value"] = (completed / total * 100) if total else 0

def clear(self):
    self.title.set("")
    self.provider.set("")
    self.duration.set("")
    self.course_id.set("")
    self.refresh()

def logout(self):
    self.root.destroy()
    start_login()

# START
def open_main_app():
    root = tk.Tk()
    LearningTrackerApp(root)
    root.mainloop()

def start_login():
    root = tk.Tk()
    LoginSystem(root)
    root.mainloop()

if __name__ == "__main__":
    start_login()

Ln: 22 Col: 0
```

Output











