Youtube link: https://youtu.be/yEOQpw-DuQ8

Python Based Classroom Management System

Muhammad Umer Danka (22104053)

Objectives

- To achieve an effective Classroom Management System facilitating attendance tracking, timetable management, and assignment submission tracking for EduHub University.
- To implement robust programming techniques and user-friendly design principles in Python, ensuring seamless functionality and ease of use for both students and faculty members.
- To design a comprehensive system architecture that efficiently organizes and manages academic data while prioritizing user experience and system reliability.

Programming Techniques

If-else

- If-else statements was applied in a many parts of my code namely in the:
- The percentage calculation section to add one value to the variable ‘present’ if data found in that location is ‘present’. Same is done for absent while an error is shown if data is not either ‘present’ or ‘absent’.
- This allowed me to send a list to if-else which makes sure that different value is achieved for different subjects or students.

```
if record[i] == 'present':
    present += 1
elif record[i] == 'absent':
    absent += 1
else:
    messagebox.showerror('da
total += 1
```

```
if current_status == 'completed':
    assignment_data[j][index] = 'incomplete'
else:
    assignment_data[j][index] = 'completed'
```

For Loop

- For loop was the most common.
- In the faculty homepage and student homepage.
- To send different data to same user defined function.
- Allowed me to display multiple students for attendance marking in faculty homepage.
- And student would only get displayed their ow attendance marking functional GUI.

```
# Loop to populate the "Mark Attendance" tab
for faculty in faculty_info:
    if faculty[0] == username:
        subject = faculty[3]
        for i, data in enumerate(attendance_data):
            if data[1] == subject:
                display = data[0]
                row = i
                attendance_record = data[2:]
                mark_current_subject_attendance(row, subje
```

```
for i, data in enumerate(attendance_data):
    if data[0] == username:
        row = i
        subject = data[1]
        attendance_record = data[2:]
        mark_current_subject_attendance(row, sub
```

Try – Except and user-defined function

- Try-Except quite uncommon.
- Found in login function.
- To ensure username and password exists.
- If error occurred, it was handled instantly.

- User-defined functions was thee ‘code’.
- It was everywhere including login.
- For delegation purposes.
- Allowing me to handle one part of the code at a time.

```
def login(student_info, faculty_info):
    try:
        username = entry_username.get()
        password = entry_password.get()

        # Check if the username exists in student_info
        for student in student_info:
            if student[0] == username:
                if student[1] == password:
                    full_name = student[2]
                    messagebox.showinfo( title: "Login Successful", message: f"Welcome, {full_name}!")
                    window.destroy()  # Close login window
                    student_homepage(username)  # Call function to open student home page
                    return

        # Check if the username exists in faculty_info
        for faculty in faculty_info:
            if faculty[0] == username:
                if faculty[1] == password:
                    full_name = faculty[2]
                    messagebox.showinfo( title: "Login Successful", message: f"Welcome, {full_name}!")
                    window.destroy()  # Close login window
                    faculty_homepage(username)  # Call function to open faculty dashboard
                    return

        # If username or password is incorrect
        messagebox.showerror( title: "Login Failed", message: "Invalid username or password.")
    except IndexError:
        messagebox.showerror( title: "Error", message: "Index error occurred. Please check the data format.")
    except ValueError:
        messagebox.showerror( title: "Error", message: "Value error occurred. Please check the data format.")
    except Exception as e:
        messagebox.showerror( title: "Error", message: f"An unexpected error occurred: {str(e)}")
```

Troubleshooting

1. Indexing

```
# Add the subject_code to the timetable entry at the specified index
timetable_data[timetable_entry_index].append(subject_code)
# Add assignment for the new subject
assignment_data[timetable_entry_index].append('incomplete')
```

- To find the location of 2d list separately of timetable & assignment.

By keeping value of:

timetable_data[i][j]

Assignment_data[i][j]

- This allowed me to create:

A functional yet clean txt file.

```
0001,CSC1024,MTH1114,MPU3112,MPU3122,MPU3142
0002,MTH1114,MPU3112,MPU3122
0003,CSC1024,MTH1114,MPU3112,MPU3122
0004,MPU3122,MPU3142
0005,CSC1024,MTH1114,MPU3112,MPU3122,MPU3142
0006,MPU3122,MPU3142
0007,CSC1024,MPU3112,MPU3122
0008,CSC1024,MTH1114,MPU3112,MPU3122
0009,MTH1114,MPU3112,MPU3122
0010,CSC1024,MPU3122,MPU3142
0011,MPU3122,MPU3142
0012,CSC1024,MTH1114,MPU3122,MPU3142
0013,MPU3122,MPU3142
0014,CSC1024,MPU3112,MPU3122,MPU3142
0015,CSC1024,MTH1114,MPU3112,MPU3122
0016
0017
0018
0019
0020
```

```
0001,completed,completed,completed,completed,completed
0002,completed,completed,completed
0003,completed,incomplete,completed,incomplete
0004,completed,completed
0005,completed,completed,completed,completed,completed
0006,completed,completed
0007,completed,completed,completed
0008,completed,incomplete,completed,completed
0009,completed,completed,completed
0010,completed,completed,completed
0011,completed,completed
0012,completed,incomplete,incomplete,incomplete
0013,completed,completed
0014,completed,completed,completed,completed
0015,incomplete,incomplete,incomplete,incomplete
0016
0017
0018
0019
0020
```

2.Refreshing

```
def change_status(index, j, faculty_code, parent_frame):
    current_status = assignment_data[j][index]

    if current_status == 'completed':
        assignment_data[j][index] = 'incomplete'
    else:
        assignment_data[j][index] = 'completed'

    # Refresh the display of assignments
    display_students_assignments(faculty_code, parent_frame)
```

- When data was edited like this:
- The tab would still show new data.

- Hence the following code helped solve that issue:
- This made sure current tab was made empty so that the function can rerun smoothly

```
def display_students_assignments(faculty_code, parent_frame):
    # Destroy any existing widgets to clear the display
    for widget in parent_frame.winfo_children():
        widget.destroy()
```

Conclusion

- At the end of this project, we have achieved:
- Problem solving capability when troubleshooting
- Applying knowledge learned from class.
- While self-learning added to enhance program. (credit: BroCode (Youtube))
- Project management skills to communicate with team members and to delegate tasks effectively.
- Time management