



**PROJECT BASED LEARNING (PBL-2) LAB**  
**(CSP 297)**

**PROJECT REPORT**  
**TECHNOVA EVENT SCHEDULER**

**B.Tech 2<sup>nd</sup> Year**

**Semester:4<sup>th</sup>**

**Session:2023-24**

**Submitted By:**

**Saumya Suman (2022438521)**

**Aman Sinha (2022534811)**

**Naveen Kumar Jha (2022518370)**

**SECTION: B**

**Group 1**



**Submitted to:**

**Dr. Priyanka Tyagi**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**  
**SHARDA SCHOOL OF ENGINEERING & TECHNOLOGY**  
**SHARDA UNIVERSITY, GREATER NOIDA**

# Table of Contents

# Page No

 <b>SHARDA</b> UNIVERSITY <i>Beyond Boundaries</i>		..... 1
<b>Project Title</b> .....		3
<b>Project Description</b> .....		3
<b>Literature Survey</b> .....		3
<b>Problem Statement</b> .....		4
<b>Objectives</b> .....		4
<b>Technologies to be used</b> .....		4
<b>Methodology for Task Scheduler</b> .....		4
<b>Algorithm for Technova Event Scheduler:</b> .....		5
<b>Future Enhancements</b> .....		6
<b>References</b> .....		6
<b>Implementation</b> .....		7
<b>Outcomes:</b> .....		10

## Project Title

### Technova Event Scheduler

## Project Description

In response to the escalating number of events hosted by the Technova society, there is a pressing need for a sophisticated solution to streamline event management processes. This project aims to develop a robust application tailored to meet the specific requirements of the Technova society, providing an effective and efficient tool for organizing and overseeing events.

**The star features included are:** Event Scheduling: The application will provide an intuitive interface for scheduling events, enabling organizers to set dates, times, and venues effortlessly. Communication Hub: A centralized platform for communication among event organizers, volunteers, and participants will be integrated to enhance collaboration and information dissemination. Smart Conflict Resolution: The application will act as an intelligent conflict resolution system that alerts organisers to potential conflicts in scheduling. Suggestions will be given for alternative dates or times based on availability

## Literature Survey

**GUI Development with Tkinter:** "Python GUI Programming with Tkinter" by Alan D. Moore - This book provides a comprehensive guide to building graphical user interfaces using Tkinter.

**Calendar Widgets in Tkinter:** "Tkinter GUI Application Development Cookbook" by Alejandro Rodas de Paz - This resource includes practical examples of building Tkinter applications, including the integration of calendar widgets.

**Task Scheduling:** "Task Scheduling in Python" by Real Python - Real Python is a valuable resource for Python developers, and this article provides insights into various aspects of task scheduling in Python.

**Database Integration for Task Storage:** "Python and SQLite: Perfect Partners for Embedded Databases" by Practical Business Python - SQLite is a lightweight embedded database that can be easily integrated into a Tkinter application for task storage.

**Voting Options and Event Handling:** "Python Tkinter Event Handling" by GeeksforGeeks - This article covers the basics of event handling in Tkinter, which is essential for implementing voting options.

**Combining Calendar and Voting in a GUI Application:** GitHub repositories and Stack Overflow threads - Search for open-source projects or discussions related to combining calendar and voting options in a Tkinter application. These resources often provide practical examples and insights from the community.

**User Experience Design:** "Don't Make Me Think" by Steve Krug - While not Python-specific, this book is a classic in UX design and can help you design an intuitive and user-friendly interface for your Tkinter application.

**Best Practices in Tkinter Development:** Tkinter documentation and tutorials on the official Python website - Understanding the best practices recommended by the official documentation can greatly enhance the quality of your Tkinter application.

## Problem Statement

Event management process within the Technova society is becoming denser as: The addition of new clubs into the society. To host and manage the events that are rising day by day a proper system of voting and an application to manage the dated events is required another problem is to view events dynamically with the help of a voting system and the final acceptance of the Technova President is required to solve the problem of scheduling and managing the events with visual representation of the date been taken.

## Objectives

**Efficient Event Management:** Focus on user-friendly application for streamlined event organization. Features include seamless scheduling, coordination, and execution. **Data Storing:** Comprehensive feature for tracking key Events. Ensures accurate records and aids in better planning and resource allocation. **Visual Approach:** Visually appealing interface that can easily represent scheduled events of the clubs as required. **User Integration:** Integration with popular event and community platforms for expanded outreach and engagement. **Security and Authorization:** Encapsulation through passwords.

## Technologies to be used

Python3, GUI (Tkinter/Pygames)

## Methodology for Task Scheduler

**Introduction:** The development of the Task Scheduler in Python using the tkinter library involved a systematic process to meet user requirements for task management. The project's primary goals were to create an intuitive and interactive application allowing users to schedule, view, and manage events within a graphical user interface.

**Requirements Analysis:** To initiate the project, a comprehensive analysis of functional requirements was conducted. Key features identified included month selection, dynamic calendar display, event entry, and addition functionalities. This phase aimed to define the scope and specifications of the application based on user needs.

**Environment Setup:** The development environment was set up to ensure a consistent and stable platform for coding. Python and the tkinter library were installed, and version compatibility was maintained among team members. The chosen development tools provided an efficient workspace for the coding process.

**GUI Design:** The graphical user interface was designed using tkinter widgets. The layout was organized to offer an intuitive and visually appealing experience for users. Elements such as labels, option menus, entries, and buttons were strategically placed to facilitate seamless interaction.

**Calendar Display Logic:** The logic for dynamically generating and displaying a calendar for the selected month was implemented. The algorithm determined the number of days in the chosen month, and labels were dynamically generated for each day. Existing events for each date were retrieved and displayed beneath the calendar.

**Event Handling:** Mechanisms for handling events, including the addition of new events, were implemented. User inputs were validated to ensure data integrity, and appropriate feedback was provided through warning messages or success notifications. For simplicity in this demonstration, an in-memory storage approach was employed.

**Data Storage:** Event storage was simulated using an in-memory dictionary. While this approach is basic and limited to demonstration purposes, it allowed for the development of core functionalities without the need for external databases or more complex storage solutions. Although we might still try to connect the database through RDBMS systems.

**Testing:** GUI testing and user acceptance testing are to be conducted to ensure that all components of the application functioned as expected. Identified issues were promptly addressed and resolved.

### Algorithm for Technova Event Scheduler:

- I. Import necessary libraries such as calendar, tkinter etc.
- II. Define a class to encapsulate the functionality of the application.
- III. In the method inside a class say `_innit_`
- IV. Initialize the Tkinter root window and set its title.
- V. Create variables to hold the input values for year, months and Events.
- VI. Create labels, entry widgets, and buttons for year, month, task functionality.
- VII. Initialize an empty dictionary to store tasks and their corresponding Information
- VIII. Define a method to handle the Add Event functionality:
  - IX. Retrieve the task and input by the user.
  - X. Add the task to the dictionary initialize above if it doesn't exist, or update its information if it does.
- XI. Reset the entry fields.
- XII. Handle the case where the date input is not a valid integer.
- XIII. Define a method to display the calendar with events and colors.
- XIV. Retrieve the year and month input by the user.
- XV. Generate the calendar text using the function(tkcalendar).
- XVI. Append event details from the dictionary to the calendar text.
- XVII. Display the calendar text in the Show All Events label.
- XVIII. Handle the case where the colour is not valid.
- XIX. This algorithm allows users to interact with the GUI to input year, month, tasks, and vote counts, and then displays a calendar along with the entered tasks and their vote counts for the specified month and year.

## Future Enhancements

**Enhancements:** Consideration was given to potential enhancements for the application, such as the integration of date pickers and persistent storage solutions. Feedback from users and evolving requirements are to be taken into account for possible future iterations.

**Deployment:** The application is prepared for deployment, ensuring compatibility across different systems. Options for packaging the application for distribution were explored to simplify deployment processes.

**User Training and Support:** Plans for user training and support are being outlined. Documentation and training materials are thought to be prepared to assist users in navigating the application. A support system is also being thought to address user inquiries and issues post-deployment.

**Improved Interface:** The improved interface will make the project more likely to be understood easier and to improve the easiness and the complexity to learn the software purpose.

## Conclusion

This project will after completion make an easier way for events to be handled in the technova society, greatly benefiting the event scheduling time and human resource needed at any given time. This program made using Python3 will use the GUI modules like Tkinter or Pygames to greatly benefit the graphical representation and make an easier way for a voting system per day basis and an option to let choose the deciding vote through the president account only. This will also help in the attendance and maintain a better decorum for planning of events with proper data.

## References

- Beniz, D. a. (2016). In *Using Tkinter of python to create graphical user interface (GUI) for scripts* (pp. 25-28).
- Lutz, M. 2. (2001). *Programming python*. O'Reilly Media, Inc.

# Implementation

```
import tkinter as tk
from tkinter import ttk, messagebox, simpledialog
import json
from tkcalendar import Calendar
import datetime
import os
import hashlib

# Load club schedules from a file if it exists, otherwise initialize as an empty dictionary
club_schedules_file = "club_schedules.json"

if os.path.exists(club_schedules_file):
    with open(club_schedules_file, "r") as file:
        club_schedules = json.load(file)
else:
    club_schedules = {}
    messagebox.showinfo("File Not Found", "Club schedules file not found. It will be created as an empty file.")

# President password
president_password = "123" # password

# Global variable to track whether the president is logged in
president_logged_in = False

# Global variables for UI elements
club_menu = None
club_var = None
president_window = None
cal = None # Calendar widget

# Define available colors for clubs
available_colors = ["#FF6347", "#4169E1", "#32CD32", "#FF8C00", "#9932CC", "#00CED1", "#FFD700", "#FF69B4", "#A52A2A"]

def save_club_schedules():
    with open(club_schedules_file, "w") as file:
        json.dump(club_schedules, file)

def update_club_dropdown():
    global club_menu
    club_menu['menu'].delete(0, 'end')
    for club in club_schedules.keys():
        club_menu['menu'].add_command(label=club, command=tk._setit(club_var, club))

def show_schedule():
    schedule_window = tk.Toplevel(root)
    schedule_window.title("Club Meeting Schedule")
    schedule_window.geometry("600x400") # Set window dimensions

    schedule_text = tk.Text(schedule_window, wrap="word", width=60, height=30)
    schedule_text.pack(padx=10, pady=10, fill="both", expand=True)

    for club_name, events in club_schedules.items():
        schedule_text.insert("end", f"({club_name}) Schedule:\n")
        for date, event in events.items():
            schedule_text.insert("end", f"({date}): {event}\n")
        schedule_text.insert("end", "\n")

    schedule_text.config(state="disabled")

def show_all_events():
    all_events_window = tk.Toplevel(president_window)

    all_events_window = tk.Toplevel(president_window)
    all_events_window.title("All Events")
    all_events_window.geometry("600x400") # Set window dimensions

    all_events_text = tk.Text(all_events_window, wrap="word", width=60, height=30)
    all_events_text.pack(padx=10, pady=10, fill="both", expand=True)

    for club_name, events in club_schedules.items():
        all_events_text.insert("end", f"({club_name}) Events:\n")
        for date, event in events.items():
            all_events_text.insert("end", f"({date}): {event}\n")
        all_events_text.insert("end", "\n")

    all_events_text.config(state="disabled")

def president_login():
    global president_logged_in
    password = simpledialog.askstring("President Login", "Enter the president password:", show='')
    # Hash the entered password and compare with the hashed password stored
    if hashlib.sha256(password.encode()).hexdigest() == hashlib.sha256(president_password.encode()).hexdigest():
        president_logged_in = True
        messagebox.showinfo("Success", "President logged in successfully.")
        root.destroy() # Close the main window upon successful login
        open_president_window() # Open the president window
    else:
        messagebox.showerror("Error", "Incorrect password.")

def add_event_window():
    club_name = club_var.get()
    if club_name:
        if club_name in club_schedules:
            event_window = tk.Toplevel(president_window)
            event_window.title("Add Event")
            event_window.geometry("400x300") # Set window dimensions

            cal = Calendar(event_window, selectmode='day', year=datetime.datetime.now().year, month=datetime.datetime.now().month, day=datetime.datetime.now().day, date_pattern="Y-mm-dd", day_abbr_font_size=20)
            cal.pack(pady=20, fill="both", expand=True) # Fill both vertically and horizontally

            # Mark booked dates on the calendar
            mark_booked_dates(cal)

            event_entry = tk.Entry(event_window)
            event_entry.pack(pady=5)

            def add_event_to_schedule():
                club_name = club_var.get()
                if not club_name:
                    messagebox.showerror("Error", "Please select a club.")
                    return

                date_str = cal.get_date()
                event = event_entry.get().strip()

                if not event:
                    messagebox.showerror("Error", "Please enter an event.")
                    return

                try:
                    date = datetime.datetime.strptime(date_str, "%Y-%m-%d").date() # Convert date string to datetime.date
                except ValueError:
                    messagebox.showerror("Error", "Invalid date format. Please use YYYY-MM-DD.")
```

```

        # Check if the date is already booked by another club
        for existing_club, events in club_schedules.items():
            if date_str in events and existing_club != club_name:
                messagebox.showerror("Error", f"The date {date_str} is already booked by '{existing_club}'.")
                return

        # If the date is not already booked, register the event
        if date_str not in club_schedules[club_name]:
            club_schedules[club_name][date_str] = event
            save_club_schedules() # Save the updated schedules
            messagebox.showinfo("Success", "Event added successfully.")

            # Mark the calendar with the booked event date
            year, month, day = map(int, date_str.split("-"))
            cal.date = datetime.date(year, month, day)
            color = available_colors[(len(available_colors) + list(club_schedules.keys()).index(club_name)) % len(available_colors)]
            cal.tag_config(club_name, background=color)
            cal.event_create(cal.date, text=club_name, tags=club_name)
            event_window.destroy()
        else:
            messagebox.showerror("Error", f"An event already exists for this date by '{club_name}'.")

            add_button = tk.Button(event_window, text="Add Event", command=add_event_to_schedule)
            add_button.pack(pady=5)
        else:
            messagebox.showerror("Error", f"Club '{club_name}' not found.")
    else:
        messagebox.showerror("Error", "Please select a club.")

def add_new_club_window():
    new_club_name = simpledialog.askstring("New Club", "Enter the name of the new club:")
    if new_club_name:
        new_club_name = new_club_name.strip()
        if new_club_name:
            if new_club_name not in club_schedules:
                club_schedules[new_club_name] = {}
                save_club_schedules() # Save the updated schedules
                messagebox.showinfo("Success", f"New club '{new_club_name}' added successfully.")
                update_club_dropdown() # Update club dropdown with new club
            else:
                messagebox.showerror("Error", f"Club '{new_club_name}' already exists.")
        else:
            messagebox.showerror("Error", "Please enter a valid club name.")

def remove_club_window():
    selected_club = club_var.get()
    if selected_club:
        confirmation = messagebox.askokcancel("Confirm Removal", f"Are you sure you want to remove '{selected_club}' club?")
        if confirmation:
            del club_schedules[selected_club]
            save_club_schedules() # Save the updated schedules
            update_club_dropdown() # Update club dropdown
            messagebox.showinfo("Success", f"Club '{selected_club}' removed successfully.")
        else:
            messagebox.showerror("Error", "Please select a club to remove.")

def remove_event_window():
    club_name = club_var.get()
    if club_name:
        event_date = simpledialog.askstring("Remove Event", "Enter the date of the event to remove (YYYY-MM-DD):")
        if event_date:
            del club_schedules[selected_club]
            save_club_schedules() # Save the updated schedules
            update_club_dropdown() # Update club dropdown
            messagebox.showinfo("Success", f"Club '{selected_club}' removed successfully.")
        else:
            messagebox.showerror("Error", "Please select a club to remove.")

def remove_event_window():
    club_name = club_var.get()
    if club_name:
        event_date = simpledialog.askstring("Remove Event", "Enter the date of the event to remove (YYYY-MM-DD):")
        if event_date:
            if event_date in club_schedules[club_name]:
                confirmation = messagebox.askokcancel("Confirm Removal", f"Are you sure you want to remove the event on {event_date}?")
                if confirmation:
                    del club_schedules[club_name][event_date]
                    save_club_schedules() # Save the updated schedules
                    messagebox.showinfo("Success", "Event removed successfully.")
            else:
                messagebox.showerror("Error", "No event found on the specified date.")
        else:
            messagebox.showerror("Error", "Please enter a valid date.")
    else:
        messagebox.showerror("Error", "Please select a club.")

def add_event():
    if president_logged_in:
        add_event_window()
    else:
        messagebox.showerror("Error", "You need to login as president to add events.")

def add_new_club():
    if president_logged_in:
        add_new_club_window()
    else:
        messagebox.showerror("Error", "You need to login as president to add new clubs.")

def open_president_window():
    global club_menu
    global club_var
    global president_window
    global cal # Calendar widget

    president_window = tk.Tk()
    president_window.title("President Options")
    president_window.geometry("800x600") # Set initial window size

    # Calendar
    cal = Calendar(president_window, selectmode='day', year=datetime.datetime.now().year, month=datetime.datetime.now().month, day=datetime.datetime.now().day, date_pattern="y-mm-dd", day_abbr_font_size=20)
    cal.pack(pady=20, fill="Both", expand=True) # Fill both vertically and horizontally

    # Mark booked dates on the calendar
    mark_booked_dates(cal)

    # Club name selection
    ttk.Label(president_window, text="Select Club:").pack(pady=(10, 0))
    club_var = tk.StringVar()
    club_menu = tk.OptionMenu(president_window, club_var, "Select Club")
    club_menu.pack(pady=5)

    update_club_dropdown() # Initialize club dropdown

```



```

# Mark booked dates on the calendar
mark_booked_dates(cal)

# Club name selection
ttk.Label(president_window, text="Select Club:").pack(pady=(10, 0))
club_var = tk.StringVar()
club_menu = ttk.OptionMenu(president_window, club_var, "Select Club")
club_menu.pack(pady=5)

update_club_dropdown() # Initialize club dropdown

add_event_button = ttk.Button(president_window, text="Add Event", command=add_event)
add_event_button.pack(pady=5)

add_club_button = ttk.Button(president_window, text="Add New Club", command=add_new_club)
add_club_button.pack(pady=5)

remove_club_button = ttk.Button(president_window, text="Remove Club", command=remove_club_window)
remove_club_button.pack(pady=5)

remove_event_button = ttk.Button(president_window, text="Remove Event", command=remove_event_window)
remove_event_button.pack(pady=5)

show_all_events_button = ttk.Button(president_window, text="Show All Events", command=show_all_events)
show_all_events_button.pack(pady=5)

def mark_booked_dates(calendar):
    for Club, events in club_schedules.items():
        for date_str in events.keys():
            year, month, day = map(int, date_str.split("-"))
            cal_date = datetime.date(year, month, day)
            color = available_colors[(len(available_colors) + list(club_schedules.keys()).index(Club)) % len(available_colors)]
            calendar.tag_config(Club, background=color)
            calendar.calevent_create(cal_date, text=Club, tags=Club)

# Create main window
root = tk.Tk()
root.title("Club Meeting Schedule")

# Styling
style = ttk.Style(root)
style.theme_use('clam')

root.geometry("800x600") # Set initial window size

# Calendar
cal = Calendar(root, selectmode='day', year=datetime.datetime.now().year, month=datetime.datetime.now().month, day=datetime.datetime.now().day, date_pattern="%Y-%m-%d", day_abbr_font_size=20)
cal.pack(pady=20, fill='both', expand=True) # Fill both vertically and horizontally

# Mark booked dates on the calendar
mark_booked_dates(cal)

# Buttons
show_schedule_button = ttk.Button(root, text="All Event Schedules", command=show_schedule)
show_schedule_button.pack(pady=5)

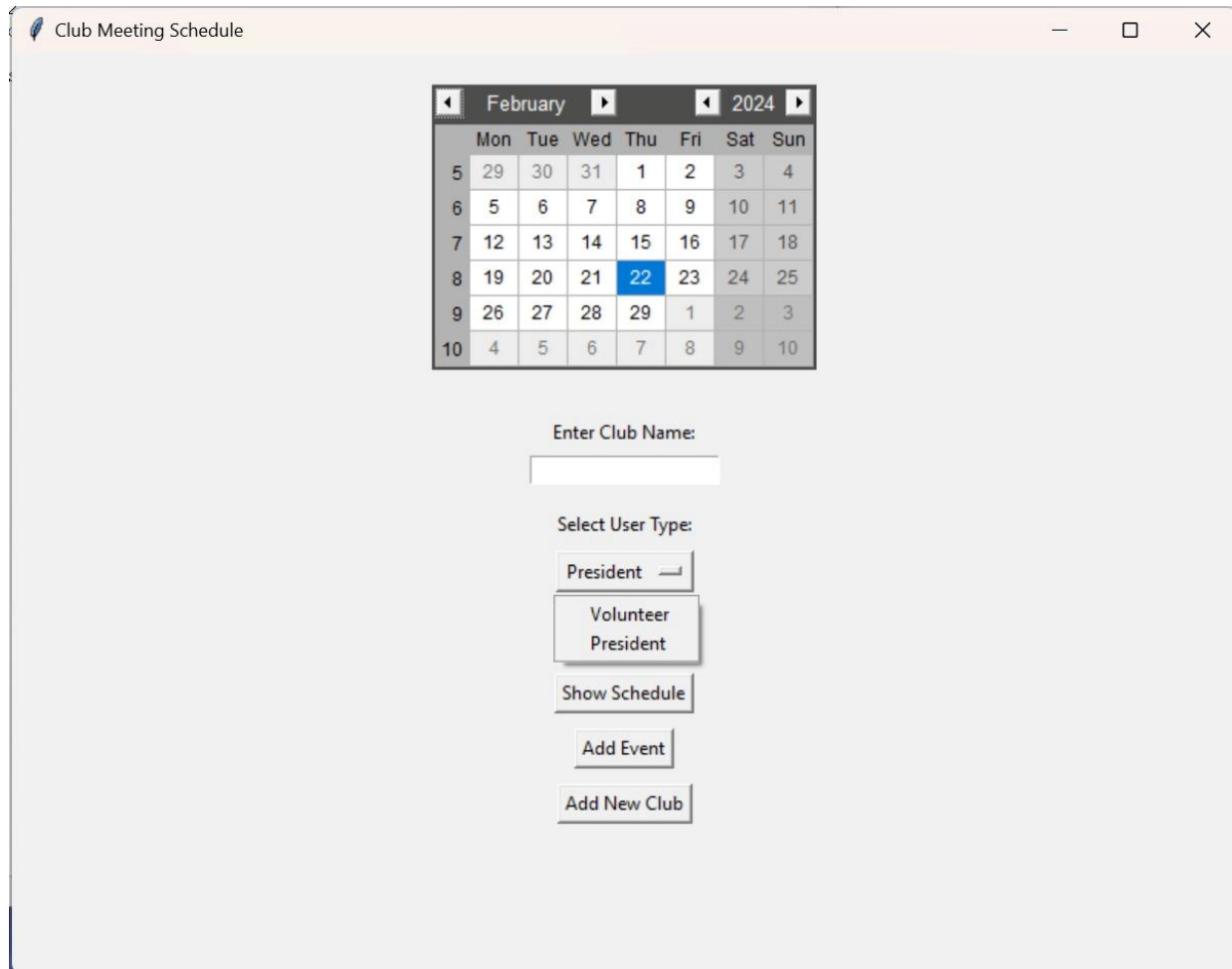
login_button = ttk.Button(root, text="Login", command=president_login)
login_button.pack(pady=5)

root.mainloop()

```

## Outcomes:

### Week 3-



The week 3<sup>rd</sup> application window visual had, The Calendar widget from TkCalendar. The option to choose between 'President' and 'Volunteer' at the beginning of the application main window. It allowed access to everyone to choose between volunteer and President at any given time. It had the options of 'Adding Events', 'Show all events' and 'Add New Club'. It had the option to write the club name to search

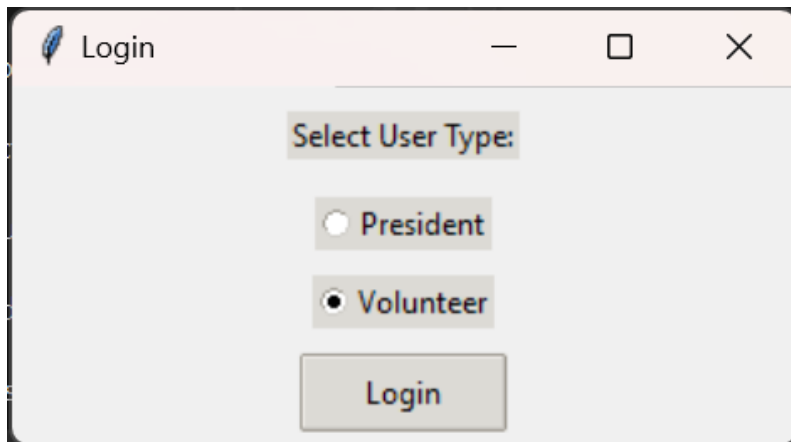
## Week 4,5,6-

The screenshot shows a web application titled "Club Meeting Schedule". At the top, there is a calendar for February 2024. The calendar grid shows dates from 29 to 10. The date 22 is highlighted in blue. Below the calendar, there is a form with the following elements:

- A label "Enter Club Name:" followed by a text input field.
- A button labeled "Under work".
- A button labeled "Select User".
- A button labeled "Show Schedule".
- A button labeled "Show Club Events".
- A button labeled "Add Event".
- A button labeled "Add New Club".
- A button labeled "Remove Club".

The week 5<sup>th</sup> 6<sup>th</sup> and 7<sup>th</sup> went under bugs fixes and proper implementation of idea of the scheduler. It now consists of selection user through a new window and requires a password to add new events. It now also had new option to Remove Clubs by typing the name under 'Enter Club Name'. Show Schedule now works as intended. It loads old events into current session.

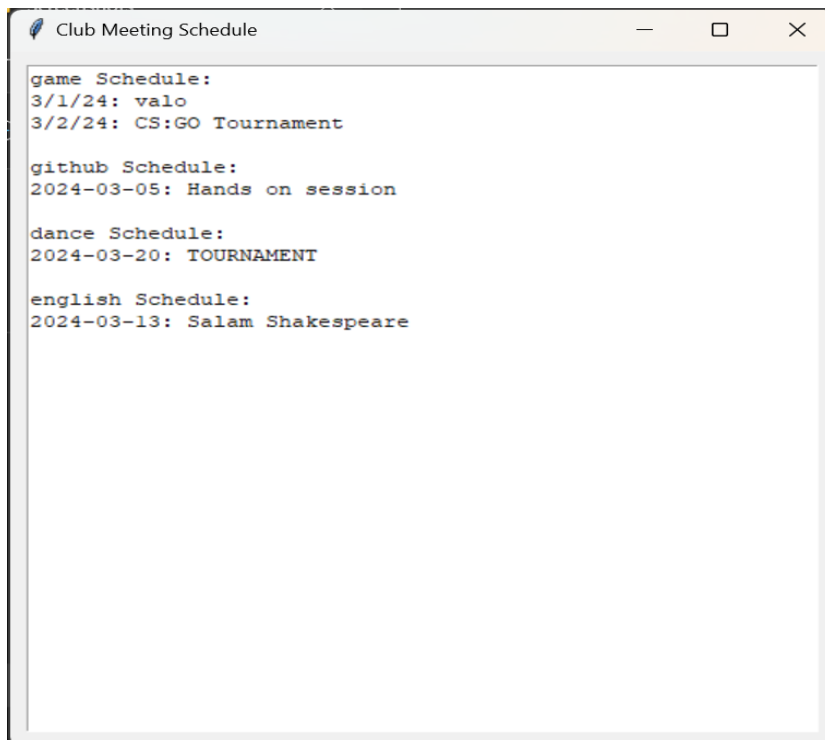
## Login Window



A screenshot of a login window titled "Login". It features a "Select User Type:" label with two radio button options: "President" and "Volunteer". The "Volunteer" option is selected. Below the options is a "Login" button.

By default it is selected to the Volunteer account

## Show Schedule



A screenshot of a window titled "Club Meeting Schedule". It displays a list of events categorized by activity type and date.

```
game Schedule:
3/1/24: valo
3/2/24: CS:GO Tournament

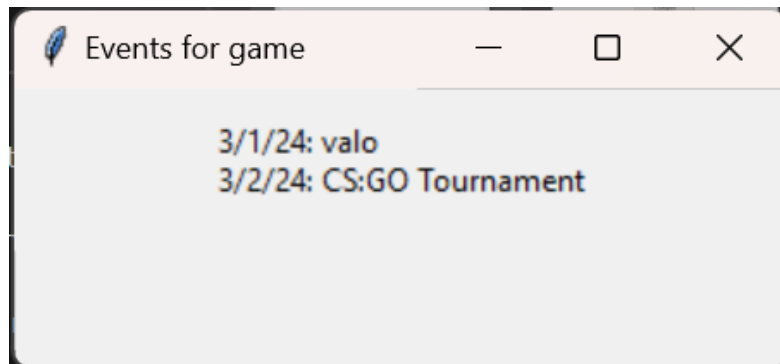
github Schedule:
2024-03-05: Hands on session

dance Schedule:
2024-03-20: TOURNAMENT

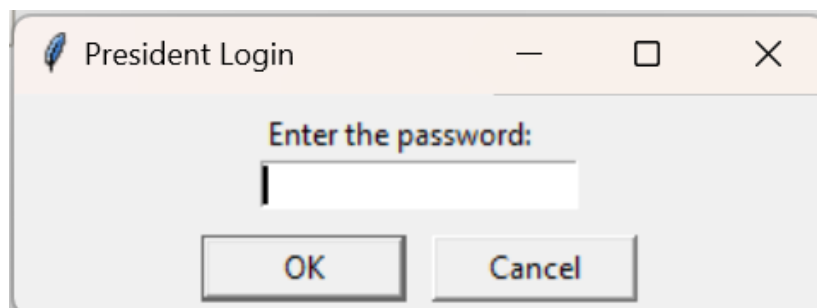
english Schedule:
2024-03-13: Salam Shakespeare
```

Displays all events that has been registered. Provides structured and grouped club meetings with dates of events. Easier way of storing data, using Python .json file

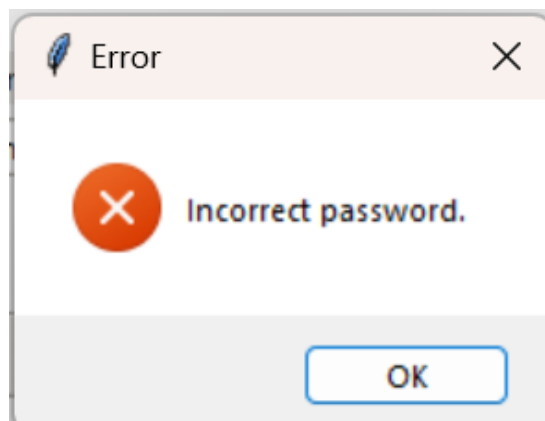
### Specific events of club window



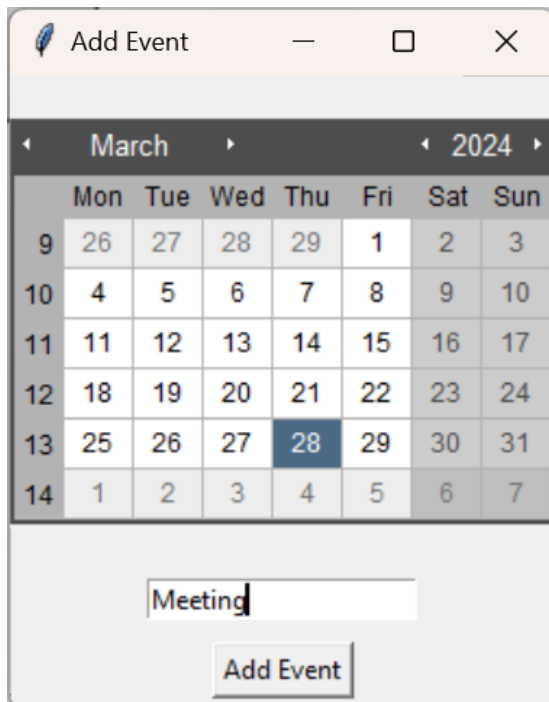
### President Login window



The President login window will be showed upon choosing the Add Event button



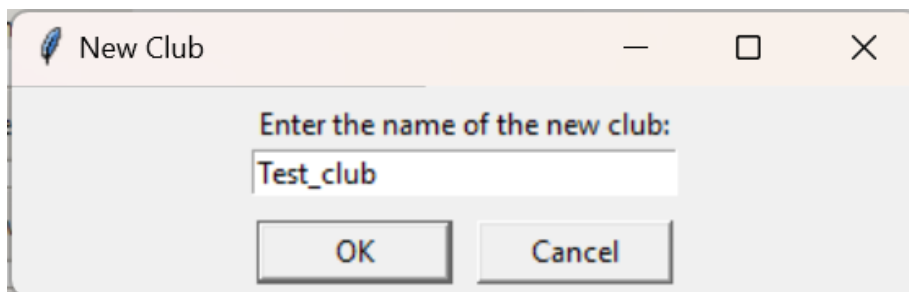
### Add event Window



The 'Add Event' window features a calendar for March 2024. The calendar grid shows dates from 1 to 31. The date 28 is highlighted in blue. Below the calendar is a text input field containing the word 'Meeting' and an 'Add Event' button.

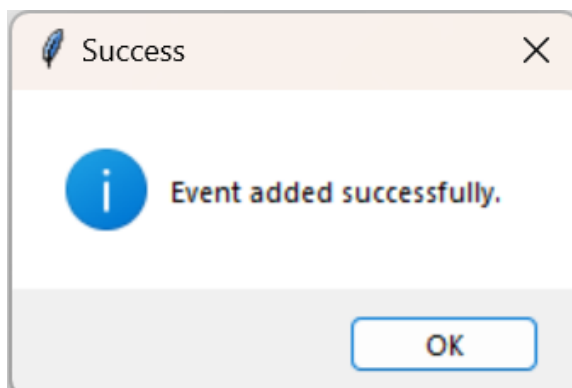
	Mon	Tue	Wed	Thu	Fri	Sat	Sun
9	26	27	28	29	1	2	3
10	4	5	6	7	8	9	10
11	11	12	13	14	15	16	17
12	18	19	20	21	22	23	24
13	25	26	27	28	29	30	31
14	1	2	3	4	5	6	7

### New Club Window

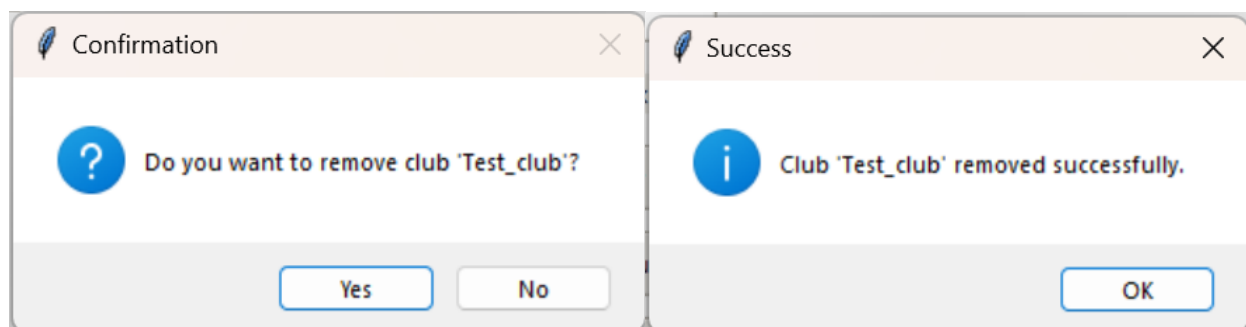
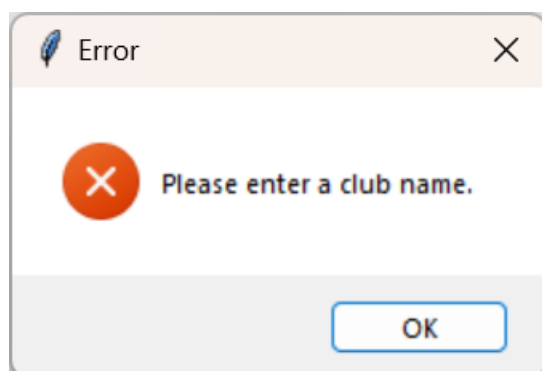
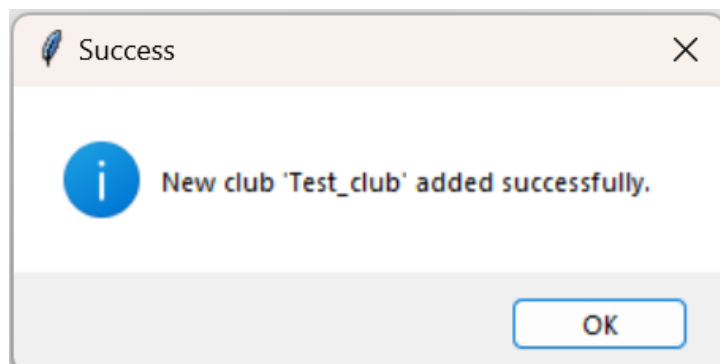


The 'New Club' window prompts the user to 'Enter the name of the new club:'. The text input field contains 'Test\_club'. Below the input field are 'OK' and 'Cancel' buttons.

### All Other Prompts:



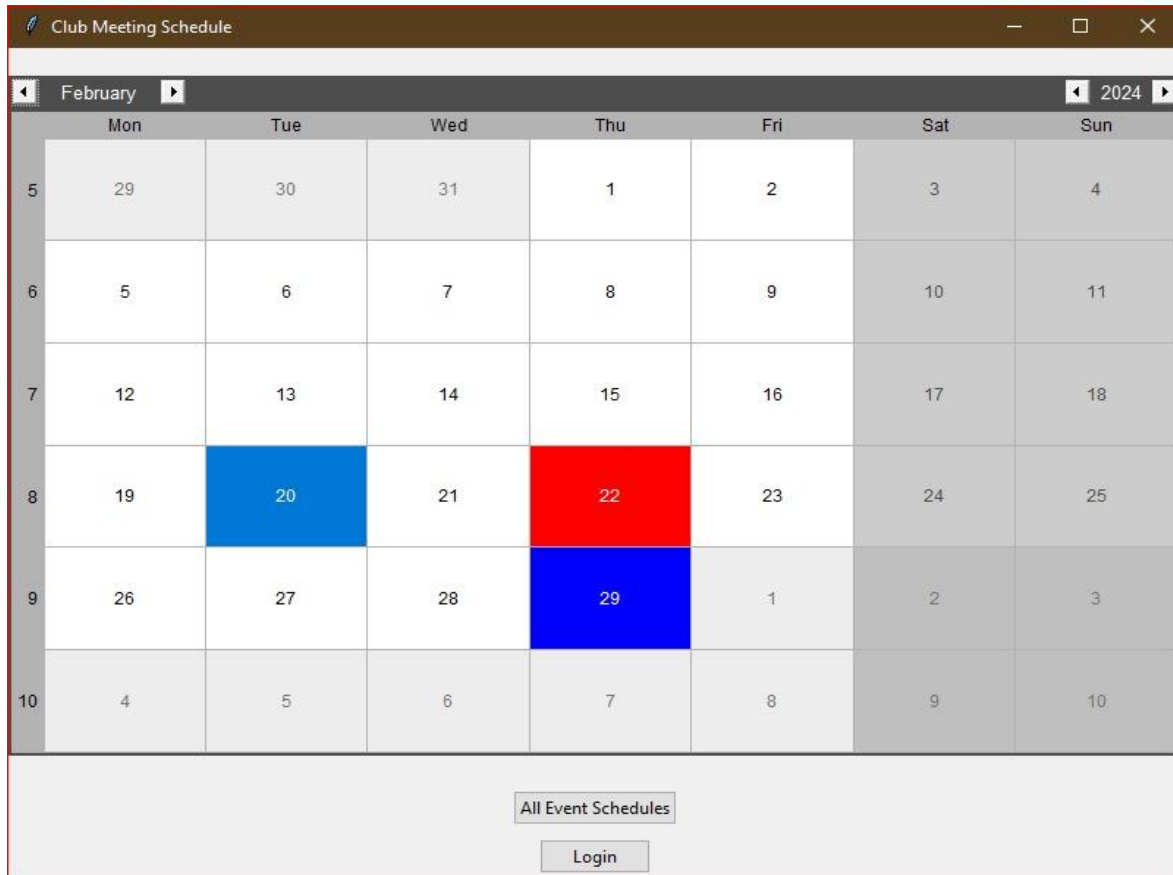
The 'Success' prompt window displays a blue information icon and the message 'Event added successfully.'. An 'OK' button is located at the bottom.



## Week 7,8,9-

### Updated Application Window

Addes visually attractive calendar with colors representing specific clubs. Everyone can now check which events are going to organise in coming months. The default login is volunteer. Added better approach to change months





## Updated President Panel

This is the updated main window of president. Added drop-down option to select club, it refreshes dynamically. Added better logic to delete events. Modified code such that data is stored in .json file easily. Added option to show all events of every club in president panel

President Options

February

2024

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
5	29	30	31	1	2	3	4
6	5	6	7	8	9	10	11
7	12	13	14	15	16	17	18
8	19	20	21	22	23	24	25
9	26	27	28	29	1	2	3
10	4	5	6	7	8	9	10

Select Club:

Select Club ▾

Add Event

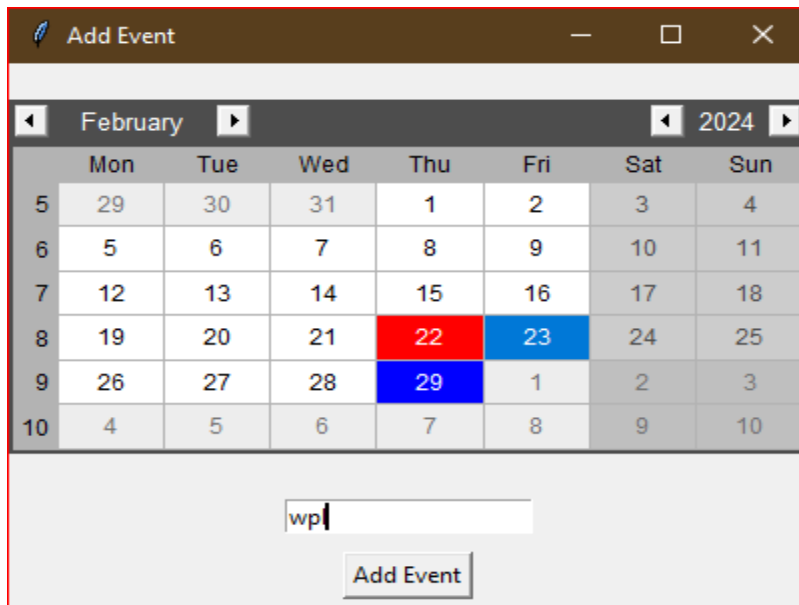
Add New Club

Remove Club

Remove Event

Show All Events

## Add Event window



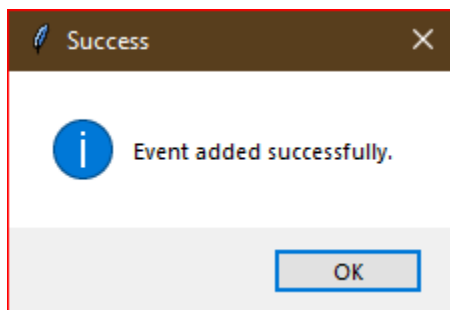
**Add Event**

February 2024

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
5	29	30	31	1	2	3	4
6	5	6	7	8	9	10	11
7	12	13	14	15	16	17	18
8	19	20	21	22	23	24	25
9	26	27	28	29	1	2	3
10	4	5	6	7	8	9	10

wp

Add Event

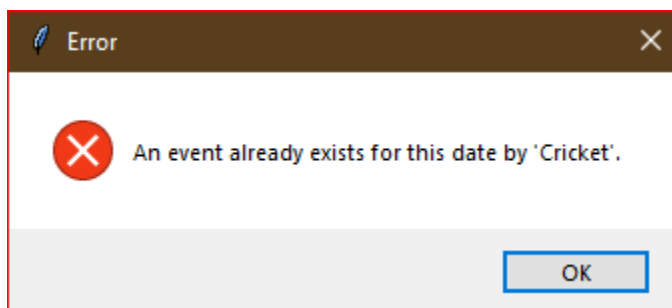


**Success**

Event added successfully.

OK

## Added logic to show which club has booked the date

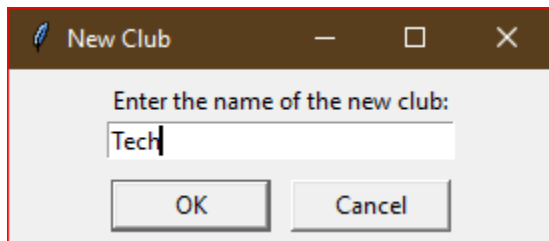


**Error**

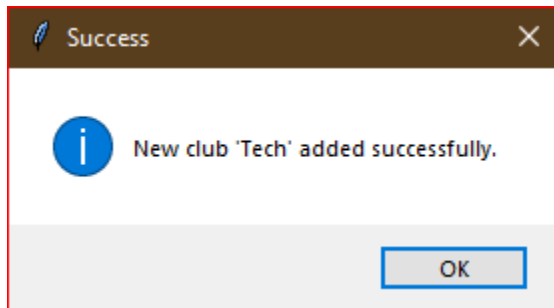
An event already exists for this date by 'Cricket'.

OK

### Add New Club Window

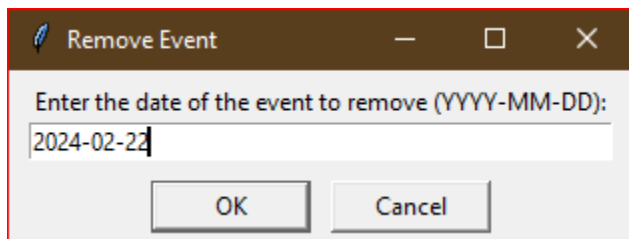


A dialog box titled "New Club" with a feather icon in the top-left corner. It contains a text input field with the label "Enter the name of the new club:" and the text "Tech" entered. Below the input field are two buttons: "OK" and "Cancel".

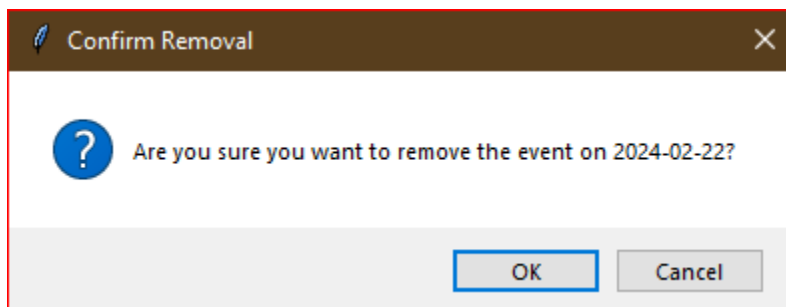


A dialog box titled "Success" with a feather icon in the top-left corner. It contains an information icon (a blue circle with a white 'i') and the text "New club 'Tech' added successfully." Below the text is an "OK" button.

### Remove Event Window



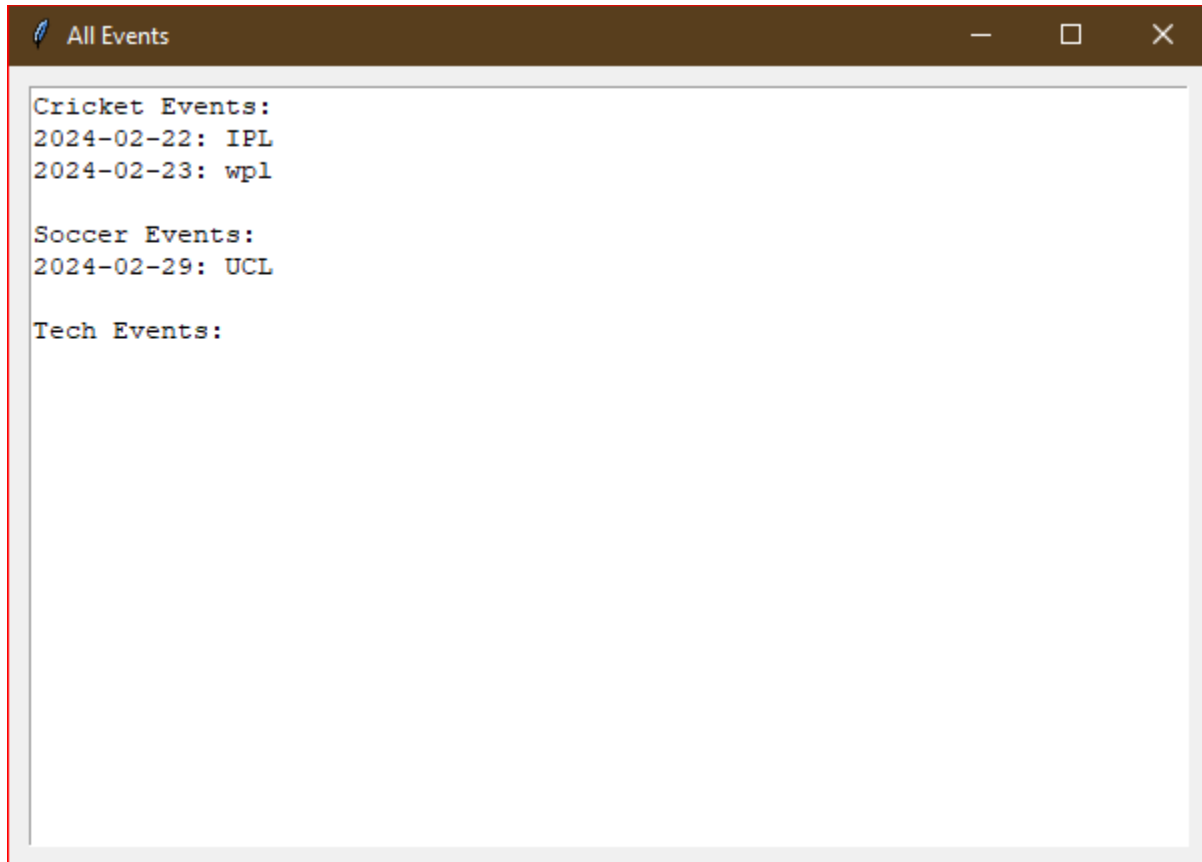
A dialog box titled "Remove Event" with a feather icon in the top-left corner. It contains a text input field with the label "Enter the date of the event to remove (YYYY-MM-DD):" and the date "2024-02-22" entered. Below the input field are two buttons: "OK" and "Cancel".



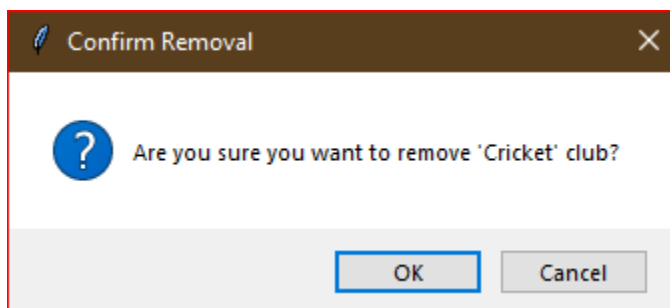
A dialog box titled "Confirm Removal" with a feather icon in the top-left corner. It contains a question mark icon (a blue circle with a white '?') and the text "Are you sure you want to remove the event on 2024-02-22?". Below the text are two buttons: "OK" and "Cancel".

### Showing All Events Window

Added logic to dynamically refresh all registered events. Added logic that all registered events can be viewed by everyone



### All Other Prompts



The current logic displays a better output in terms of both the prompts which helps manage the root cause in case of an intersection of event or any error is found during evaluation of events.

**Faculty Signature**