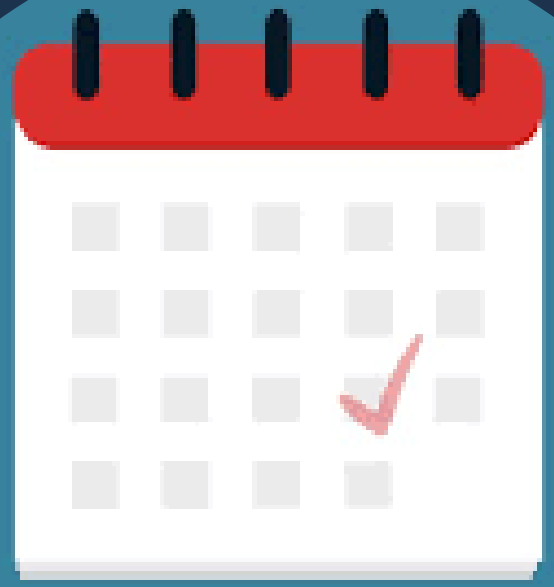
Calendar App

Capgemini Programming Assignment

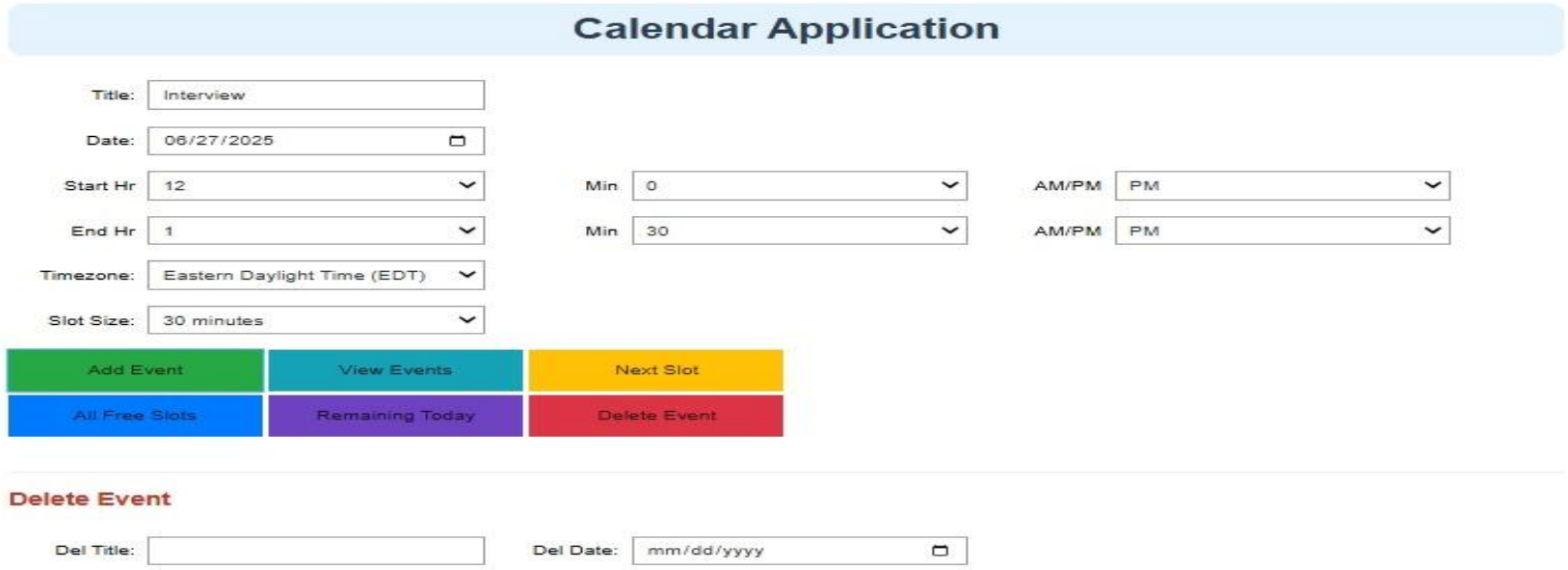
Keerthana Vemuganti



Problem Statement

- Build a calendar and appointment scheduling tool using Python.
- The application should allow users to: Add events with title, start time, and end timePrevent overlapping events.
- View all events for a specific dayView remaining events for todayFind next available time slot.
- Show all available free slots of a specific duration.
- Delete events to free memory and update the data fileNo database is required; events are stored in a .json file.

Output:



Requirements

App must support:

Listing all events for the current day

Listing all remaining events for today

Listing all events for any specified date

Suggesting the next available time slot of a given duration

Key Python Libraries

```
import json
from datetime import datetime, timedelta, date, time
import pytz
import ipywidgets as widgets
from IPython.display import display, clear_output
```

These libraries help:

- Create an interactive calendar input system
- Maintain and display events chronologically
- Provide time zone and AM/PM support

Project Structure

- Notebook Used: Calendar.ipynb
  Data File: calendar.json (stores added events)
  Classes and Functions:
- CalendarApp class with methods:
- add_event()
- list_all_events()
- list_remaining_events()
- list_events_by_day()
- find_next_available_slot()

Key Features Implemented

Feature – Add EventPurpose: Add a new event if it doesn't overlap with existing events

Code Snippet :

```
def add(self, title, start, end):
    for e in self.events:
        if e.start < end and start < e.end:
            return False
    self.events.append(Event(title, start, end))
    self.save()
    return True
```

Description : Checks for time conflict. If there's no overlap, adds the event and saves the updated list to calendar.json.

Key Features Implemented

Feature – View Events for Selected Day

Purpose: Display all events scheduled on a given day

Code Snippet:

```
def on_day(self, day):
    return [e for e in self.events if e.start.date() == day]
```

Description: Filters events stored in memory based on the selected date and returns only those matching the day.

Key Features Implemented

Feature – Delete Event

Purpose: Delete an event by title and date from memory and JSON file

Code Snippet:

```
def delete(self, title, day):
    before = len(self.events)
    self.events = [e for e in self.events if not (e.title == title and e.start.date() == day)]
    self.save()
    return len(self.events) < before
```

Description: Deletes events that match the given title and date, then rewrites the updated list back to the file.

Key Features Implemented

Feature – View Next Available Slot

Purpose: Return the next available slot of given duration

Code Snippet:

```
def free_slot(self, mins, day, tz):
    t = tz.localize(datetime.combine(day, time(8, 0)))
    end = tz.localize(datetime.combine(day, time(23, 59)))
    while t + timedelta(minutes=mins) <= end:
        if all(t + timedelta(minutes=mins) <= e.start or t >= e.end for e in self.events):
            return t, t + timedelta(minutes=mins)
        t += timedelta(minutes=15)
    return None, None
```

Description: Iterates through the day from 8:00 AM to 11:59 PM and returns the first available time range that doesn't overlap with existing events.

Key Features Implemented

Feature – View All Free SlotsPurpose: List all free time slots of the selected duration

Code Snippet:

```
def all_free_slots(self, mins, day, tz):
    slots = []
    t = tz.localize(datetime.combine(day, time(8, 0)))
    end_of_day = tz.localize(datetime.combine(day, time(23, 59)))
    while t + timedelta(minutes=mins) <= end_of_day:
        candidate_end = t + timedelta(minutes=mins)
        overlap = any(e.start < candidate_end and t < e.end for e in self.events)
        if not overlap:
            slots.append((t, candidate_end))
            t = candidate_end
        else:
            t += timedelta(minutes=15)
    return slots
```

Description: Finds all non-overlapping intervals of the selected duration throughout the day and returns them in a list.

Key Features Implemented

Feature – Convert AM/PM to 24-Hour Time

Purpose: Convert user input time into 24-hour format

Code Snippet:

```
def to24(h, ap):
    return h if ap == "AM" and h != 12 else (h + 12 if ap == "PM" and h != 12 else 0)
```

Description: Used to convert user selections for hour and AM/PM into 24-hour time format for consistent processing.

Key Features Implemented

Feature – Clear Input Fields

Purpose: Reset the form after adding an event

Code Snippet:

```
def clear_inputs():
    title.value = ""
    date_picker.value = None
    sh.value, sm.value, sap.value = 9, 0, "AM"
    eh.value, em.value, eap.value = 10, 0, "AM"
```

Description: Clears all inputs to default values so the user can add a new event without manually resetting fields.

Key Features Implemented

Feature – Time Zone Selection

Purpose: Ensure all events are scheduled and displayed in the user's selected time zone.

Code Snippet (exact from your code):

```
tz = widgets.Dropdown(
    options=[

        ("Eastern Daylight Time (EDT)", "US/Eastern"),
        ("Central Daylight Time (CDT)", "US/Central"),
        ("Mountain Daylight Time (MDT)", "US/Mountain"),
        ("Pacific Daylight Time (PDT)", "US/Pacific"),
        ("India Standard Time (IST)", "Asia/Kolkata"),
        ("Greenwich Mean Time (GMT)", "Etc/GMT")
    ],
    description="Timezone:"
)
```

Description: The dropdown widget allows the user to choose a timezone from predefined options.This selected timezone is used when creating or viewing events, ensuring times are localized.It works together with pytz.timezone(tz.value) to convert the selected date and time into a proper timezone-aware datetime object.

UI/UX Design Choices

Widgets used:

- DatePicker for selecting date
- Dropdowns for hour, minute, and AM/PMButtons to trigger actions

Other Design Choices:

- Form clears after adding an event
- Events are shown in chronological order
- Supports readable, minimal UI with labels like “Add Event,” “View Events,” etc.

Conclusion & Improvements

Fully functional basic calendar app built
All assignment features completed
Future Improvements:

Add event editing and deletion

Integrate with real calendars (Google Calendar API)

Support recurring events

Build responsive web UI using Streamlit


Thank you!!!