main.py

```
1  from calendarApp import shell, models
2  import os
3
4
5  def main():
6      os.system("clear")
7      calendar = models.Calendar("main")
8      shell.main_screen(calendar)
9
10
11 if __name__ == "__main__":
12     main()
```

calendarApp/models.py

```
from datetime import datetime
import uuid


class Calendar():
    def __init__(self, name):
        self.schedule = []
        self.name = name

    def add_event(self, event_name, start_time, end_time):
        # Create Event() object and append it to schedule list
        event = Event(event_name, start_time, end_time)
        self.schedule.append(event)
        print(f"[INFO] Event {event_name} added")

        # Reorders the schedule list
        self.order_events()

    def delete_event(self, event_ids):
        # Creates a list of doomed events whose ids match those in event_ids
        doomed_events = list(filter(lambda event: (
            str(event.id) in event_ids), self.schedule))
        doomed_event_ids = [str(event.id) for event in doomed_events]

        # Make sure that all event_ids given exist in the schedule
        if all(id in doomed_event_ids for id in event_ids):
            for doomed_event in doomed_events:
                self.schedule.remove(doomed_event)
            print(f"[INFO] Event(s) {event_ids} removed")
        else:
            print("[WARNING] Invalid ID included in selection. Operation aborted.")

    def print_events(self):
        for event in self.schedule:
            print(event)

    def order_events(self):
```

```
        # Sort events by their start time
        self.schedule.sort(key=(lambda event: event.start_time))


class Event():
    def __init__(self, name, start_time, end_time):
        self.id = uuid.uuid4()
        self.name = name
        self.start_time = self.parse_time(start_time)
        self.end_time = self.parse_time(end_time)

    def __str__(self):
        return f"Event ID: {self.id} \n\tName: {self.name} \n\tTime:{self.start_time} to {self.end_time}"

    def parse_time(self, time_str):
        # Parses the given string data into datetime objects
        obj_str = datetime.strptime(time_str, "%d/%m/%Y %H:%M:%S")
        return obj_str
```

---

calendarApp/shell.py

```
def main_screen(calendar):
    while True:
        print("What would you like to do? ")
        print("\ta) Add event")
        print("\tb) Delete event")
        print("\tc) Print events")
        print("\td) Exit")
        answer = input(" $ ")

        if "a" in answer:
            add_event(calendar)
        elif "b" in answer:
            delete_event(calendar)
        elif "c" in answer:
            print_events(calendar)
        elif "d" in answer:
            break
        else:
            pass


def add_event(calendar):
    print("What is the name of the event that you would like to add?")
    name = input(" $ ")
    print("When does your event start? FORMAT: dd/mm/yyyy hh:mm:ss")
    start_time = input(" $ ")
    print("When does your event end? FORMAT: dd/mm/yyyy hh:mm:ss")
    end_time = input(" $ ")
    try:
        calendar.add_event(name, start_time, end_time)
    except ValueError:
        print("Check your inputs, they don't seem to be in the right format.")


def delete_event(calendar):
    calendar.print_events()
    print("What is the ID of the event(s) you would like to delete? Separate ids with ,.")
```

```
    event_ids = input(" $ ")
    event_ids = event_ids.split(",")
    calendar.delete_event(event_ids)


def print_events(calendar):
    calendar.print_events()
```

PDF document made with CodePrint using Prism