```
from datetime import datetime, timedelta

def time_to_minutes(time_str):              //O(1)
    h, m = map(int, time_str.split(':'))    //O(1)
    return h * 60 + m                       //O(1)

def minutes_to_time(minutes):
    return f"{minutes // 60:02}:{minutes % 60:02}"  //O(1)

def calculate_free_intervals(busy_intervals, daily_start, daily_end):
    free_intervals = []                             //O(1)
    last_end = daily_start                          //O(1)
    for start, end in busy_intervals:               //o(n)
        if last_end < start:                        //O(1)
            free_intervals.append([last_end, start]) //O(1)
        last_end = max(last_end, end)                //O(1)
    if last_end < daily_end:                        //O(1)
        free_intervals.append([last_end, daily_end]) //O(1)
    return free_intervals                            //O(1)

def find_common_intervals(all_free_intervals, meeting_duration):
    common_intervals = all_free_intervals[0]           //O(1)
    for free_intervals in all_free_intervals[1:]:      //O(n)
        new_common = []                                //O(1)
        i = j = 0                                      //O(1)
        while i < len(common_intervals) and j <
len(free_intervals):  //O(n) n is number of participants
            start = max(common_intervals[i][0], free_intervals[j]
[0]) //O(1)
            end = min(common_intervals[i][1], free_intervals[j]
[1])   //O(1)
            if end - start >=
meeting_duration:                        //O(1)
                new_common.append([start,
end])                        //O(1)
            if common_intervals[i][1] < free_intervals[j]
[1]:         //O(1)
                i +=
1                                              //O(1)
            else:
                j +=
1                                              //O(1)
        common_intervals =
new_common                                  //O(1)
    return [[minutes_to_time(start), minutes_to_time(end)] for start,
end in common_intervals]   //O(m) //m is the number of common
intervals

//^^^^ O(m*n)
```

```
def group_schedule_match(schedules, working_periods,
meeting_duration):
    all_free_intervals =
[]
//O(1)
    for busy_intervals, (login, logout) in zip(schedules,
working_periods):                                       //O(n)
        busy_intervals = [[time_to_minutes(start),
time_to_minutes(end)] for start, end in busy_intervals]      //O(p) p
is number of busy intervals
        daily_start =
time_to_minutes(login)
//O(1)
        daily_end =
time_to_minutes(logout)
//O(1)
        free_intervals = calculate_free_intervals(busy_intervals,
daily_start, daily_end)                         //O(n) n is number of
participants

all_free_intervals.append(free_intervals)
//O(1)
    return find_common_intervals(all_free_intervals,
meeting_duration)                                             //O(m*n)

//^^^^^^ O(m*n) + O(p*n) = O(n(m+p))

def main():
    num_participants = int(input("Enter the number of participants:
"))     //O(n*p) n is participants and p is busy intervals
    schedules =
[]                                                             //O(1)
    working_periods =
[]                                                    //O(1)

    for i in range(num_participants):
        print(f"\nEntering schedule for Participant {i + 1}:")
        daily_login = input("Enter daily login time (HH:MM): ")
        daily_logout = input("Enter daily logout time (HH:MM): ")
        working_periods.append([daily_login, daily_logout])

        busy_intervals = []
        num_busy_periods = int(input("Enter number of busy periods for
this participant: "))

        for j in range(num_busy_periods):
            start = input(f"  Start time for busy period {j + 1}
(HH:MM): ")
            end = input(f"  End time for busy period {j + 1} (HH:MM):
")
```

```
            busy_intervals.append([start, end])

        schedules.append(busy_intervals)

    meeting_duration = int(input("\nEnter the minimum meeting duration
(in minutes): "))
    available_slots = group_schedule_match(schedules, working_periods,
meeting_duration) /// //^^^^^^ O(m*n) + O(p*n) = O(n(m+p)) n is number
of participants p is number of busy intervals m is the number of
common intervals

    print("\nAvailable meeting times for all participants:")
    if available_slots:
        for start, end in available_slots:
            print(f"  {start} -
{end}")                                     //O(m) number of common
intervals
    else:
        print("  No common available times meet the required
duration.")  //O(m) number of common intervals

if __name__ == "__main__":
    main()
```