CS5001 HW6: Due on Canvas Friday Dec 8 at 11:59pm

You must work **either on your own or with one partner**. If you work with a partner, you and your partner must first register as a group with us then submit your work as a group on Canvas. To register, please send an email to the TAs at-- "Qing Chen" chen.qing1@northeastern.edu ; "Lingyu Hu" hu.lingyu@northeastern.edu ; "Mingtianfang Li" li.mingt@northeastern.edu ; "Kejian Tong" tong.ke@northeastern.edu

NOTE: If working in pairs, you can pair up with the same person for a maximum of two assignments/projects in this course. After that, you must either find another partner or complete the assignments individually. Any team in violation of this would receive a zero for their submissions after the first two submissions.

NOTE2: Since this assignment is due on the last day of classes, **you cannot use your late days on this assignment**. No submission or late submission will result in a score of zero.

You may discuss background issues and general solution strategies with others, but the programs you submit must be the work of just you (and your partner). We assume that you are thoroughly familiar with the discussion of academic integrity that is on the course website. Any doubts that you have about "crossing the line" should be discussed with Tas or the instructor before the deadline.

1. Introduction

In this assignment you will create a program for appointment management system. The system will manage appointment information for multiple users. Each user will have his/her own appointment diary. An appointment diary will allow the user to maintain their appointments from now until December 31st, 2024. (This is a soft deadline. At least dates until the end of 2024 must be supported. If your implementation supports more than that, no problem.) Each appointment entry in the diary contains information about date, time, and purpose of the appointment. Each user should be able to perform several operations, some of which are listed below (for complete list and description refer to Expected Program Behavior):

1. Schedule appointment for a certain date, time interval (indicated by start time and end time), and purpose. (Make sure you don't make an appointment at an already scheduled time.)
2. Cancel the appointment for a specified date and time interval.
3. Check whether user has an appointment during a given time interval on a date.
4. Retrieve the purpose of an appointment at a given date, time interval.
5. Reschedule an existing appointment to a new date or time.

2. Expected program behavior

On starting execution your program should show a welcome screen which should display the following options and information:

Welcome to Appointment Management System! What would you like to do?

- [a] Add new user
- [d] Delete an existing user
- [l] List existing users
- [s] Schedule an appointment
- [c] Cancel an appointment
- [f] Check for appointment on certain date and time
- [p] Retrieve purpose of an appointment
- [r] Reschedule an existing appointment
- [x] Exit the system

Entering the characters in square brackets above selects the respective option. For example, entering "l" (lower case "El") would allow user to list all existing users in the system, "s" will allow the user to schedule an appointment, etc. Until the user chooses to exit the system, the system should keep coming back to the above welcome screen message. If an invalid option is entered display message "Invalid Option," and display the above options again.

Below is a summary of expected system behavior for each of the above mentioned options. Note that asking for time in the below options implies asking for both start time and end time.

- [a] Add new user **(10 points)**

The system asks for a new username, adds the user to the system, and creates a new appointment diary for the specified user. Before adding the user, you should check if the user already exists in the system. If

so, print an error message and go back to the welcome screen. If adding was successful, print a message indicating that and go back to the welcome screen.

NOTES:

1. Usernames consist of alphabets, numbers, dots, and underscores. No other characters are permitted. If a user enters a username with characters other than these, print an appropriate error message and ask for another valid username.
2. Usernames are NOT case sensitive, i.e., topstudent and TopStudent will refer to the same user.

- [d] Delete an existing user **(10 points)**

The system asks for the username to be deleted. The system then checks if the user exists. If so, remove the user from the system, delete all entries in its diary, and any other relevant user specific data. On successful deletion, print a success message and go back to the welcome screen. If the user was not found, print an error message and return to the welcome screen.

- [l] List existing users **(5 points)**

Print a list of all the usernames currently holding appointment diaries in the system. The list should be sorted in Python dictionary ordering (A to Z, etc.) .Return to the welcome screen.

- [s] Schedule an appointment **(15 points)**

Ask for username (if invalid user entered, print error message and return to welcome screen), date (if invalid date entered, print error message and return to welcome screen), time (if invalid time entered, print error message and return to welcome screen), and purpose. Add the appointment as long as it does not conflict with an existing appointment. In case of conflict, print appropriate error message and go back to the welcome screen.

NOTES:

1. Invalid user means the user does not exist in the system, yet.
2. Time is accepted in the following format: HH:MM AM/PM. Your interface asking the user for time should clarify this. The same time format will be used throughout the assignment.
3. You should always ask user to enter the date in following format: YYYY-MM-DD

- [c] Cancel an appointment **(15 points)**

Ask for username (if invalid user entered, print error message and return to welcome screen), date (if invalid date entered, print error message and return to welcome screen), time (if invalid time entered, print error message and return to welcome screen). Remove the appointment at the specified date, time. If no matching appointment found, print error message. Return to welcome screen.

NOTE: To search for the appointment to cancel ask for the exact username, date, and starting time. Your interface should explicitly ask for these.

- [f] Check for appointment on certain date and time **(10 points)**

Ask for username (if invalid user entered, print error message and return to welcome screen), date (if invalid date entered, print error message and return to welcome screen), time (if invalid time entered, print error message and return to welcome screen). Check if the user has an appointment in the specified date and time; if so, print appointment found. Else, inform no appointment found. Return to the welcome screen.

NOTE: To search for the appointment ask for the exact username, date, and starting time. Your interface should explicitly ask for these.

- [p] Retrieve purpose of an appointment **(15 points)**

Ask for username (if invalid user entered, print error message and return to welcome screen), date (if invalid date entered, print error message and return to welcome screen), time (if invalid time entered, print error message and return to welcome screen). Check if the user has an appointment in the specified date and time; if so, print the purpose of the appointment. Else, inform no appointment found. Return to the welcome screen.

NOTE: To search for the appointment ask for the exact username, date, and starting time. Your interface should explicitly ask for these.

- [r] Reschedule an existing appointment **(15 points)**

Ask for username (if invalid user entered, print error message and return to welcome screen), old date (if invalid date entered, print error message and return to welcome screen), old time (if invalid time entered, print error message and return to welcome screen). Check if the user has any appointment in the entered date and time (if not print error message and return to welcome screen), if so, ask for new date and time (check for invalid date and time values). Reschedule the appointment to the new date and time as long as there is no conflict in the new time. Else print conflict error message. Return to welcome screen.

NOTE: To search for the appointment to reschedule ask for the exact username, date, and starting time. Your interface should explicitly ask for these.

- [x] Exit the system **(5 points)**

Print a goodbye message and terminate the program.

Prepare a “README.txt” file for your submission. The file should contain the following:

I. Instructions for compiling and executing your program(s). Include an example command line for the same.
II. If your implementation does not work, you should also document the problems in the README file, preferably with your explanation of why it does not work and how you would solve it if you had more time.

3. Implementation Notes:

1. You will need to implement the following classes:
    a. User class, representing each user of the system. It should contain:
        i. User name
        ii. AppointmentDiary object representing the user’s diary

   b. Time class representing time in hours and minutes, containing
      i. Hours
      ii. Minutes
      iii. AM or PM
   c. Date class, containing the following.
      i. Year
      ii. Month
      iii. Day
   d. Appointment class
      i. Date of appointment
      ii. Time of appointment (start time and end time)
      iii. Purpose of appointment
   e. AppointmentDiary class containing the following and any other information you may need to maintain for supporting the requested functionality.
      i. Some kind of a list of all the appointments that a given user has scheduled.
2. In addition to the above mentioned classes, data members you will need to create functions for several tasks, e.g., you may want to have a function for checking whether a provided date is valid or not, etc. These are left to you as your design decisions. You need to implement all necessary functions and any additional data members that might be needed for providing the requested functionality.
3. You should always ask user to enter the date in following format: YYYY-MM-DD
4. Time is always accepted in the following format: HH:MM AM/PM.
5. In your program you may need to maintain several lists, for example you can maintain a list of users in the system, for each user, you could have a list of appointments in their appointment diary, etc.
6. Implement different classes in different files/modules and then import them in the main script and anywhere else needed. For instance, create a separate Python modules for the Appointment, AppointmentDiary, User, etc., classes.
7. Note that the users and appointment diaries, etc., are not expected to survive across multiple program runs. Each time the program begins execution, it will start with an empty list of users and diaries. However, If you would like the user data persist across program runs, you can save it in a file in the current working directory. The next time you start your program, read the data stored in the file to populate your class/dict/list/etc. Nevertheless, please note that the assignment doesn't require you to implement this functionality. It is okay if the entered data does not show up in the next run of your program.

4. Sample output

Below is an excerpt of the expected output. Please note, this is not the complete sample illustrating all the options, but, hopefully, this will help you better understand the functionality. For this example, assume, we have already added a user "neu" to the system and this user has the following three appointments scheduled on 2023-04-16 between 9:30 AM to 10:10 AM, 11 AM to 11:20 AM, and 5:25 PM to 5:55 PM.

Welcome to Appointment Management System! What would you like to do?

- [a] Add new user
- [d] Delete an existing user
- [l] List existing users
- [s] Schedule an appointment
- [c] Cancel an appointment
- [f] Check for appointment on certain date and time
- [p] Retrieve purpose of an appointment
- [r] Reschedule an existing appointment
- [x] Exit the system

Enter choice: f

Enter username: neu

Enter date: 2023-04-16

Enter start time: 5:15 PM

No appointment found!

Welcome to Appointment Management System! What would you like to do?

- [a] Add new user
- [d] Delete an existing user
- [l] List existing users
- [s] Schedule an appointment
- [c] Cancel an appointment
- [f] Check for appointment on certain date and time
- [p] Retrieve purpose of an appointment
- [r] Reschedule an existing appointment
- [x] Exit the system

Enter choice: f

Enter username: neu

Enter date: 2023-04-16

Enter start time: 5:25 PM

Appointment found on 2023-04-16 between 5:25 PM to 5:55 PM.

Welcome to Appointment Management System! What would you like to do?

- [a] Add new user
- [d] Delete an existing user
- [l] List existing users

- [s] Schedule an appointment
- [c] Cancel an appointment
- [f] Check for appointment on certain date and time
- [p] Retrieve purpose of an appointment
- [r] Reschedule an existing appointment
- [x] Exit the system

Enter choice: x

Goodbye!

5. Rubrics

Your programs will be tested against several test cases. For full credit, a program should pass all the test cases. Programs failing several test cases will receive a very low grade.

Rubrics mentioned in Section-2 (Expected program behavior) above will be used for grading your work. Note that you need to handle invalid inputs as directed in Section-2. Your program should not crash or terminate until the user explicitly exits using option "x". If a program crashes/terminates for invalid inputs, points will be deducted from every feature that crashes for such cases.

6. What to submit?

1. Prepare a README.txt file containing the following:

a) Include a quick summary of how you run your program**.**
b) If any of the functionality in your programs is not working, include what is the issue in your opinion and how would you fix it if you had more time?

2. Submit README.txt, and all your code files and any other files that might be needed for executing your program, inside a single zip file on Canvas.

7. Additional directions

Following directions apply to all the problems above. Negative numbers indicate the penalty to be applied on your final score for a problem/direction if the directions are not followed.

1. Start your programs with a docstring (comment) summarizing what it is doing, what are the inputs, and the expected output. (-5 points)
2. Every function definition should start with a docstring explaining the what the function returns/does, and pre-conditions on the passed parameters. (-10)
3. At the beginning, as three comments, include your name (both teammates if applicable), date, and your email ids. (-5 points)
4. For every variable defined in your program, include a brief comment alongside explaining what it stores. (-5 points)
5. README.txt file needs to be submitted as described above. (-10 points)
6. Submit all the files (.py and .txt files) as a single zip folder/file on Canvas. (-5 points)

7. No new submissions, code files can be accepted after the deadline. Please double check and ensure that you are submitting everything needed to run your programs, and the code files are the best versions you want evaluated. Only the material submitted in the zip file uploaded on Canvas before the deadline shall be graded.