Old Dominion University

Department of Computer Science

Semester Project

Hospital Management System

CS250

Spring 2017

Dr. A Elmesalami

Introduction:

Everyone has been to a doctor's office or a hospital. The operation requires a high level of accuracy when scheduling appointments, handling patient's records, and generating reports. This is a key factor for the flow of operations that take place in such facilities. This project is aimed to automate all patients' record handling and appointment scheduling. This project will expose Computer Science students to a real world problem that will prepare them to be a good problem solvers in the industry of software development.

The Basics: The Big Picture:

Consider that you are tasked with building a Hospital Management System(H.M.S). H.M.S allows patients to book doctor appointments. It also allows doctors to view their patients' medical progress, produce reports on appointment slots (reserved or empty), and manage these appointment slots. In this problem, the system will manage appointment data for multiple doctors of various dates and times. Every time a patient visits a doctor, his/her medical records are updated. Additionally, his/her appointment records and doctor notes are also updated. Report production is essential in H.M.S. The system allows the doctor to view medical histories and lab results for his/her patients.

Description: Project Details:

ADT's

1. Patient: This ADT contains all information related to the patient. It has the following data members.
    a. id: A unique item number that identifies the patient
    b. doctor_id: The id number of the patient's doctor
    c. record_id: The id number of the patient's record
    d. name: Patient's name
    e. phone: Patient's phone number
    f. gender: The patient's gender
    g. email: The patient's Email
    h. Blood_Type: the blood type (A+,O- …etc)
2. Doctor: This ADT contains all information related to the doctor. It has the following data members.
    a. id: A unique item number that identifies the doctor
    b. username: The doctor's username
    c. password: The doctor's password
    d. Name: The doctor's full name
    e. phone: doctor's phone number

    f. skill: The doctor's specialty (e.g. General, Hematology, Gynecology …etc)
3. Slot: This ADT contains all information related to available slots for doctors starting from current_date + 1 day to current_day + 30 days. It has the following data members. Each slot is 1 hour.
    a. id: A unique number that identifies the slot
    b. date_id: The id number of the date and time of the slot
    c. doctor_id: The id number of the doctor
    d. available: The slot status (1 for available, 0 for unavailable)
4. Date_Time: This ADT contains dates and times.
    a. id: A unique number that identifies the date
    b. day: of the month (1-31)
    c. month: of the year (1-12)
    d. year: 2 digits (eg. 17 for 2017)
    e. hours: of the day (1-12)
    f. Minutes: (0-59)
    g. AM_PM: 1 char for morning or afternoon (A for AM and P for PM)
5. Record: This ADT contains all information related to the Patient Record. It has the following data members.
    a. id: A unique item number that identifies the record
    b. patient_id: The patient's id.
    c. Notes: The most recent doctor note about the patient
6. Diagnoses : This ADT contains all information related to the Diagnoses. It has the following data members.
    a. id: A unique number that identifies the diagnosis/disease.
    b. Disease _name: heart attack, broken arm
7. Appointment:
    a. Id: A number that identifies the appointment (it should be autoincrement)
    b. Doctor_id: The doctor's id
    c. Patient_id: The patient_id
    d. Slot_id: The slot_id associated with the appointment
    e. Reason: The reason for the visit (Back pain, Short breath, …etc)
    f. Diagnosis_id: Filled after the appointment is concluded and an update to the patient's record is performed.
    g. Prescribed_medication: Filled after the appointment is concluded and an update to the patient's record is performed.

Required ADT's:

1. Patient
2. Doctor
3. Slot
4. Date_Time
5. Record
6. Diagnosis
7. Appointment

   Feel free to make more if you think you need to; you must justify why you need more ADT's. This is the minimum amount of ADT's allowed in this project.

Functional Requirements:

<u>Program Flow:</u>

The operator of this program will be a hospital clerk with little knowledge about computers. The operator, like all data entry people, is not the smartest person in the world. However, the operator can follow instructions. Moving forward, this data entry person who operates the program is called a "user".

After launching the program, the main menu is displayed on the screen for the user, something like this:

Welcome to HMS. Please choose an action:

Main Menu:

1. Manage Appointments
2. Manage Slots
3. Manage Doctors
4. Manage Patients
5. Manage Patients' records

Extra credit: 6. Manage labs

After selecting 1 or 2 or …. or 5, the program prints a different sub-menu for each menu.

Option 1:

1. Schedule a new appointment
2. Show all appointments
3. Modify/Reschedule an existing appointment
4. Remove an appointment
5. Main menu: takes the user back to the main menu

Option 2:

1. Show available slots
2. Show unavailable slots
3. Add new slot(s)
4. Show slots between two times
5. Show all slots
6. Show all slots for a certain doctor
7. Modify slot information
8. Remove a slot
9. Main menu: takes the user back to the main menu

Option 3:

Same as options 1 and 2 with minor changes; at this point, the idea is rather clear.

Note: You can assume that all input from the user will not have any errors in it.

User Stories:

Creating a list of user stories is the best way to break up a project into multiple smaller projects/functions that are easier to implement.

1. As a user, I want to be able to add a new patient.
2. As a user, I want to be able to update a patient's info.
3. As a user, I want to be able to delete a patient.
4. As a user, I want to be able to show all patients. (must show their doctor's name, diagnosis, …etc).
5. As a user, I want to be able to search for a patient by his/her name or email. (must show their doctor's name, diagnosis, …etc).
6. As a user, I want to be able to show all patients assigned to a certain doctor by doctor's name.
7. As a user, I want to be able to show all patients who have appointments within a date and time range.

8. As a user, I want to be able to show all doctors.
9. As a user, I want to be able to update a doctor's info.
10. As a user, I want to be able to delete a doctor. A doctor cannot be deleted if he/she has upcoming appointments unless these appointments are canceled first.
11. As a user, I want to be able to search for a doctor by his/her name.
12. As a user, I want to be able to update a slot.
13. As a user, I want to be able to delete a slot.
14. As a user, I want to be able to show all slots.
15. As a user, I want to be able to show all available slots.
16. As a user, I want to be able to show all unavailable slots.
17. As a user, I want to be able to search for an available slot doctor's name.
18. As a user, I want to be able to show all available slots within date and time range.
19. As a user, I want to be able to update a patient's record.
20. As a user, I want to be able to update a patient's diagnosis.
21. As a user, I want to be able to show all appointments for a doctor by doctor's name.
22. As a user, I want to be able to show all appointments for a patient by patient's name, email, and dob.
23. As a user, I want to be able to update an appointment.
24. As a user, I want to be able to book a new appointment.
25. As a user, I want to be able to cancel an appointment (The slot associated with the deleted appointment must be changed to available). Confirmation message is required.

EXTRA CREDIT 10pts

26. As a user, I want to be able to create an appointment for a lab. A lab can only be used once a day.
27. As a user, I want to be able to delete a lab appointment.
28. As a user, I want to be able to show all lab appointments.

Data Storage:

It is completely up to the student how to store the records in files as long as the program functions properly.

Input files:

5 input files will be given to you. The first integer located in the file indicates how many of the objects are listed. "Doctors.txt" will hold the current doctors in the hospital and will include personal information such as an id, a username, a password, a name (first and last combined), a phone number and a specialty. "Patients.txt" will hold the current patients being seen at the

hospital. This file will also hold an id, a doctor's id, a record id, a name (first and last combined), a phone number, a gender, an email, and a blood type. "Slots.txt" holds an id, a date_time id, a doctor's id, and an indication of whether the slot is available or not. You will notice that the first 30 slots hold valid doctor ids while the last 10 slots do not. These last 10 slots can be ignored or used for the extra credit portion. "Date_Time.txt" will hold a day, month, year, hour, minute, and an indication of whether the time is am or pm. Lastly, "Diagnosis.txt" will hold a list of strings. The order that the information was listed here corresponds to the order it is listed in the input file.

Grading

- Compilation 15%
- **Correct Output**
  - Menu Functions Correctly 5%
  - Manage Appointments Correctly 5%
  - Manage Doctors Correctly 5%
  - Manage Records Correctly 5%
  - Manage Slots Correctly 10%
  - Manage Patients Correctly 10%
- Appropriate ADT's 30%
- Using Member Functions 10%
- Comments and Indentation 5%
- Extra Credit +10%

Penalties

- Including .cpp files in another .cpp file -20%

These penalties will be deducted from your total points obtained, i.e. if a student obtains 95 out of 100 and that student included .cpp files in another .cpp file, then his/her grade will be 75 out of 100.

Example of output: