Student Name: **Weight:** **25%**

Student ID: **Marks:** **/100**

Type: Group Assignment

- Students should **ONLY USE** programming constructs covered in the course material.
- **Submissions using programming concepts that are not covered in the course material will be penalized. Penalty with no limit could be applied.**
- **Late submissions will not be accepted**

Scenario

Alberta Hospital (AH) is a new healthcare provider in Alberta. To complement the existing large-scale hospitals located in urban settings, AH is building a network of smaller scale mini-hospitals which target underserved rural populations. AH has hired your company to create a management system which is customized to meet their unique operational needs.

Equipment and Materials

For this assignment, you will need:

- Python IDE
- GitHub repository

Instructions

This assignment consists of **three** sections, all completed outside of class time. See the course outline and Brightspace for exact dates.

1- GitHub Training LinkedIn Learning (5%)

Learning how to work efficiently with a team in a hosting platform such as GitHub is an essential skill for programmers. A group coding project such as this one provides the perfect opportunity to learn about and then practice this essential skill.

Complete the following LinkedIn Learning courses:

- GitHub Essential Training [2 h 48 m] (https://www.linkedin.com/learning/github-essential-training/version-control-and-collaboration-with-github)
- Git Essential Training: The Basics [2 h 55 m] (https://www.linkedin.com/learning/git-essential-training-the-basics/use-git-version-control-software-to-manage-project-code)
- Any other course that is pre-approved by your instructor

Submit a copy of your certificate of completion or other evidence of completion, as approved by your instructor.

Note: There should be no out of pocket expenses for the LinkedIn Learning course. As a SAIT student, you have free access to thousands of professional development courses through LinkedIn Learning. Ask your instructor if you run into issues accessing the courses.

2- Project Submission (90%)

Part 1: (No Submission)

1. Working **individually**, review the Scenario and the Management System Details sections of this document.
2. Create the classes outlined in the document.
3. Test the classes to verify their correct operation.
4. Each student is encouraged to individually complete Part 1. At the very least, all group members MUST participate in Part 1.

Part 2: (Submission)

1. Have one member of your group set up a GitHub repository for Part 2 of this project. Make sure that this GitHub repository is **private**. Add group members and your instructor as collaborators.
2. Create a separate branch in GitHub for each group member, containing the part they will work on. The branch name should include the task name and the student's name.
3. Develop the code for the Management System as specified in this document. As a team, you will also need to assess each member's work and then finalize one version for your group's solution.
4. GitHub must be effectively used for group collaboration.
5. Ensure all group members publish their parts to their respective GitHub branches.
6. The final project code must be published to master/main on GitHub.
7. Check your solution against the detailed marking criteria available in Brightspace.
8. **Create a project video**.
   Each group will submit a **video** to Brightspace or their GitHub Repository (either as a URL or an uploaded video file). In the video:
   - EACH team member will introduce themselves – camera should be on. Please use laptop camera/screen share to create video (not a phone).
   - Share the **group's final solution** (Python code) on screen and have EACH team member explain a **portion of the code** that they developed. A debugger may be used to step through sections of code (optional).
   - Provide a demonstration of the application covering all the functionality CORRESPONDING TO THE SAMPLE RUN PROVIDED.
   - Ensure your group's video is 5 to 10 minutes (max!) in length and double check that it is playable and has good audio quality.

9. **Submit the following files to Brightspace**:
    - A ZIP file for the whole project folder that includes:
        - The code of the program that you implemented (.py files).
        - A copy of your sample runs (.txt or .pdf file(s)) that correspond to the sample runs provided.
        - All data files updated by program after the test runs have been completed.
    - A link to your group's GitHub project repository.
    - Video presentation.
    - Peer assessment.

3- Peer Assessment (5%)

- Each member should assess the contribution/participation of all other group members.
- Each member should assign a mark out of 10 to other members.
- Ensure that the task description is clear and accurately reflects the actual participation.
- Please use the following table to complete the peer assessment. Marks/tasks are just samples.

Member/Reviewer	Member 1 Name	Member 2 Name	Member 3 Name	Completed Tasks
Member 1 Name	N/A	9	10	• Task 1 description • Task 2 description
Member 2 Name	8	N/A	9	• Task 3 description • Task 4 description
Member 3 Name	9	10	N/A	• Task 5 description • Task 6 description
Average	8.5	9.5	9.5	

- A student can receive a deduction on their individual grade for the assignment if they do not meet the expected contribution/participation based on the feedback of other group members.

Management System Details

Alberta Hospital (AH) requires that their management system application meets the following criteria.

- Supports data entry as well as report generation
- Uses the following classes throughout the application:
  1 - Doctor
  2 - Facility
  3 - Laboratory
  4 - Patient
  5 - Management
- Uses classes to create objects that interact with each other
- Uses the methods/functions listed below for each class.
- Each object has descriptive properties/ characteristics that represent the work and actions of the class as outlined below.

Class 1: Doctor

Properties

ID, Name, Specialization, Working Time, Qualification, Room Number

Methods

Method Name	Description
formatDrInfo	Formats each doctor's information (properties) in the same format used in the .txt file (i.e., has underscores between values)
enterDrInfo	Asks the user to enter doctor properties (listed in the Properties point)
readDoctorsFile	Reads from "doctors.txt" file and fills the doctor objects in a list
searchDoctorById	Searches whether the doctor is in the list of doctors/file using the doctor ID that the user enters
searchDoctorByName	Searches whether the doctor is in the list of doctors/file using the doctor name that the user enters
displayDoctorInfo	Displays doctor information on different lines, as a list
editDoctorInfo	Asks the user to enter the ID of the doctor to change their information, and then the user can enter the new doctor information
displayDoctorsList	Displays all the doctors' information, read from the file, as a report/table
writeListOfDoctorsToFile	Writes the list of doctors to the doctors.txt file after formatting it correctly
addDrToFile	Writes doctors to the doctors.txt file after formatting it correctly

Sample data: doctors.txt (data file provided)

Class 2: Facility

Properties

Facility name

Methods

Method name	Description
addFacility	Adds and writes the facility name to the file
displayFacilities	Displays the list of facilities
writeListOffacilitiesToFile	Writes the facilities list to facilities.txt

Sample data: facilities.txt (data file provided)

Class 3: Laboratory

Properties

Lab name, cost

Methods

Method Name	Description
addLabToFile	Adds and writes the lab name to the file in the format of the data that is in the file
writeListOfLabsToFile	Writes the list of labs into the file laboratories.txt
displayLabsList	Displays the list of laboratories
formatDrInfo	Formats the Laboratory object similar to the laboratories.txt file
enterLaboratoryInfo	Asks the user to enter lab name and cost and forms a Laboratory object
readLaboratoriesFile	Reads the laboratories.txt file and fills its contents in a list of Laboratory objects

Sample data: laboratories.txt (data file provided)

Class 4: Patient

Properties

pid, name, disease, gender, age

Methods

Method Name	Description
formatPatientInfo	Formats patient information to be added to the file
enterPatientInfo	Asks the user to enter the patient info
readPatientsFile	Reads from file patients.txt
searchPatientById	Searches for a patient using their ID
displayPatientInfo	Displays patient info
editPatientInfo	Asks the user to edit patient information
displayPatientsList	Displays the list of patients
writeListOfPatientsToFile	Writes a list of patients into the patients.txt file
addPatientToFile	Adds a new patient to the file

Sample data: patients.txt (data file provided)

Class 5: Management

Create a function called **DisplayMenu** to display the menu shown in the Sample Run section. The program should continue until the user enters **0**.

Test Plan

Please refer to the accompanying document, *Project-SampleRun.pdf*.

Make sure your program output matches this sample run. When doing your final test run, be sure to begin with a fresh starting data file.

Marking Criteria

	Needs Improvement (0–50%)	Good (51–75%)	Excellent (76–100%)	Marks
Working code	• The project doesn't run in all scenarios • Input requests work but don't match the scenario • No conversion of data types • Syntax of if/else statements has mistakes • Use of classes is poor • Use of functions is poor • Output works but doesn't match the scenario • Unable to read/write to files, or with many mistakes	• The project runs in all scenarios • Input requests work but don't match the scenario • Some functions or methods are missing • Correct use of if/else statements • Correct use of classes • Output works but doesn't completely match the scenario • Able to read/write to files but with a few mistakes	• The project runs in all scenarios • Input requests match the scenario exactly • Correct functions and methods developed • Correct use of if/else statements • Correct use of classes • Output matches the scenario • Able to read and write to files with no mistakes	**/55**

Style	• Indentation – not consistent
	• Readability – poor variable names
	• Documentation
	  ○ No comments are included at the top.
	• No comments indicating major code sections or what they do
		• Indentation – some parts are consistent and some are not
		• Readability – some variable names are not ideal
		• Documentation
		  ○ Comments at the top are missing or incomplete.
		• Comments indicating major code sections and what they do are incomplete
			• Indentation – consistent
			• Readability – good variable names
			• Documentation
			  ○ Comments at the top are complete and include name, date, program description including details on inputs, processing and outputs (4–5 sentences minimum).
			  ○ Comments indicate major code sections and what they do
				/15

Testing	• Sample output doesn't match the provided sample run • Output is not formatted according to the specification (sample run)	• Parts of the sample output don't exactly match the provided sample run • Output formatted according to the specification (sample run)	• Sample output exactly matches the provided sample run • Output formatted according to the specification (sample run)	**/10**
Version control (evaluated in GitHub)	• No evidence that group members practiced version control best practices	• Some evidence that some group members adhered to version control best practices.	• It is evident that all group members are consistently adhering to version control best practices.	**/5**
Peer assessment	• Not submitted	• Unrealistic or incomplete	• Completed for all group members	**/5**
GitHub training	• No evidence submitted of having completed a LinkedIn Learning course.		• LinkedIn Learning Certificate of Completion submitted OR • Other, instructor-approved proof of completion submitted	**/5**
Video demonstration	• Not submitted	• Incomplete	• Completed for all group members	**/5**
			Total	**/100**