**ENGR 101 - Introduction to Programming**

**Mini Project 2**

(Posted on December 5, 2019, Due on December 19, 2019)

In this mini project, you are going to develop a console-based Course Management System! This software will be used by students to add/drop courses. The primary goal of this mini project is to practice using various data structures (i.e., list, dictionary). Hence, please use every opportunity to employ a data structure in your code. At various points, you will be forced to use particular data structure as explained in the “Implementation Notes” section at the end. The detailed explanations about how you are going to develop this application are provided below. (Note: example console output from the application is written in Blue).

**Text-based User Interface:**

When your program first runs, it will ask for an ID and password to login. Based on the entered user ID, different menus will be displayed (Admin menu/Student menu). In your code, at least, the following admin and students should be already defined:

* Admin: ID: ‘admin’ password: ‘sehir123’’
* Students: ID:“Ahmet” password:123 and ID: “ayse” password: 456

You may define additional admins/students if you like (Please use the most proper data structure to store ID / password information. Hint: Consider a dictionary).

As long as the user enters invalid id or password, the program should ask the user to try again. The user successfully logs in when the provided credentials are valid. The following shows the interface for the initial login screen with a scenario where the user first enters incorrect ID/password, and then successfully logs in during the second trial.

\*\*\*\*Welcome to Course Management System\*\*\*\*

Please provide login information

Id: admin

Password: 123

Invalid id or password please try again

Id: admin

Password: sehir123

Successfully logged in!

Once the user enters valid credentials, your program should greet the user and provide admin menu if user is an admin or student menu if the user is a student.

**Admin menu**

User will do a selection by entering the corresponding menu number, and accordingly different information will be shown and/or requested. If the user enters an invalid menu entry, the program should keep asking the user to provide a valid menu number.

Welcome Admin! What do you want to do?

1-List courses

2-Create a course

3-Delete a course

4-Show students registered to a course

5-Users Budget Menu

6-List Users

7-Create User

8-Delete User

9-Exit

Your choice: 1

When **main menu item 1** is chosen, the list of available courses will be shown as follows. After listing courses, your program should display the admin menu again.

\*\*\* Offered Courses \*\*\*

Course name Credit

1- physics 4

2- mathematics 3

3- programming 3

When **main menu item 2** is chosen, the user will be asked to write the name of the course he/she would like to add(ex: chemistry), then the program will ask about the number of credits for the added course(ex: 4). Once the course’s name and credits are provided by user the program will print the course’s name along with credits number and ask the user if he/she is sure about adding this course, input should be Y/N, other than that the program should ask for a valid input. Your program should properly save the course’s name along with its credits and go back to the admin menu.

Welcome Admin! What do you want to do?

1-List courses

2-Create a course

3-Delete a course

4-Show students registered to a course

5-Users Budget Menu

6-List Users

7-Create User

8-Delete User

9-Exit

Your choice: 2

What is the name of the course that you want to add? chemistry

How many credits this course has? 4

chemistry will be added with 4 credits.

Are you sure?[Y/N] Y

chemistry has been added to courses with 4 credits

Welcome Admin! What do you want to do?

1-List courses

2-Create a course

3-Delete a course

4-Show students registered to a course

5-Users Budget Menu

6-List Users

7-Create User

8-Delete User

9-Exit

Your choice:

When **main menu item 3** is chosen, the user will be able to see courses with their credits which were already stored. By default, you must store the following courses and their credits using a proper data structure. (Hint: consider a dictionary). Then the program asks the user which course to delete (You should properly update student budgets and students courses accordingly), then return to admin menu. (Note: the program should keep asking for a valid input)

Your choice: 3

Course name Credit

1- physics 4

2- mathematics 3

3- programming 3

Which course do you want to delete? 1

physics has been deleted and money has been transferred back to student accounts

When **main menu item 4** is chosen, the user should be asked to choose an exist course name to display students registered to that course, then return to admin menu.

Your choice:4

Which course do you want to show? chemistry

This course doesn't exist, please provide a valid input

Which course do you want to show? mathematics

Course Name: mathematics

Students taking mathematics:

1- Ahmet

When **main menu item 5** is chosen, student budgets are shown, and a menu is displayed to ask the user if he/she wants to add money, subtract money or to go back to admin menu. (your program should update student’s budget accordingly) then your program should go back to admin menu. (Note: the program should keep asking for a valid input)

Your choice:5

User Money

1-Ahmet 900

2-Ayse 400

What do you want to do?

1-Add money to user

2-Subtract money from user

3-Back to admin menu

Your choice:1

Which user do you want add money to their account?

1-Ahmet

2-Ayse

Your choice:1

How much money do you want to add? 20

20$ will be added to Ahmet.

Are you sure?[Y/N]:Y

**Note: A similar process will take place for Subtract money from user, but instead of adding money, your program will subtract money from student’s budget.**

When **main menu item 6** is chosen, the current list of students should be shown as follows:

Current Users:

1-Ahmet

2-Ayse

When **main menu item 7** is chosen, the user will be asked to set a name, password, and budget for the new created user as following:

What is the name of user that you want to create? Fatih

What is the password for account? 006

How much money do you want user to have? 500

The new user has been added successfully!

After adding new user, your program should go the admin menu

When **main menu item 8** is chosen, current students should be displayed, then the user should be asked which user the program should delete. Then your program should go back to admin menu. (Note: the program should keep asking for a valid input)

Current Users:

1-Ahmet

2-Ayse

Which user do you want to delete: 1

Ahmet is deleted!

When **main menu item 9,** the program should go back to Login page.

**Student menu**

When a student logs in, the following menu will be displayed:

\*\*\*\*Welcome to Course Management System\*\*\*\*

Please provide login information to log in:

ID: Ahmet

Password: 123

Welcome Ahmet! What do you want to do?

1-Add courses to my courses

2-Delete a course from my courses

3-Show my courses

4-Budget Menu

5-Exit

Your choice:

When **main menu item 1**, your program should print the available courses, then asks the user which course he/she wants to add.

If the course is already taken by the user, your program should tell so and ask the user to choose a different course.

If the budget of the user is lower than the cost of the course, the program should not allow the user to choose the course. In order to compute the cost of a course, you may assume that each credit costs $100.

Your program should return to student menu.

Your choice:1

Course Name Credit

1- physics 4

2- mathematics 3

3- programming 3

Which course do you want to take (Enter 0 to go to main menu)? 1

You don't have enough money in your account. Please deposit money, or choose a course with lesser credit.

Which course do you want to take (Enter 0 to go to main menu)? 2

This course is already in your profile, please choose another course:

Which course do you want to take (Enter 0 to go to main menu)? 3

programming has been successfully added to your courses.

Welcome Ahmet! What do you want to do?

1-Add courses to my courses

2-Delete a course from my courses

3-Show my courses

4-Budget Menu

5-Exit

Your choice:

When **main menu item 2,** the program first will show courses taken by the student and ask him/her which course to delete, the amount of money for the deleted course should be added back to the student’s budget.

Your choice:2

Course Name Credit

1- mathematics 3

2-programming 3

Which course do you want to remove? 1

You have chosen: mathematics

300$ will be returned to your account

Are you sure that you want to remove this course? [Y/N] Y

Course has been deleted from your profile

After deleting the course the program return the user to student menu.

When **main menu item 3,** Courses chosen by the user should be displayed. (in your code deleted courses should not be shown) then go back to student menu.

Your choice: 3

Your courses:

Course Name Credit

1- programming 3

When **main menu item 4,** the current budget of the user should be displayed, then the user is being asked to add money to his account or to go back to student menu as following:

Your choice:4

#### BUDGET MENU #####

Your budget is: 500$

What do you want to do?

1-Add Money

2-Go to main menu

Your choice:1

Amount of money:200

Your budget has been updated.

Your program then will go back to student menu.

When **main menu item 5,** the program should go back to Login page.

**Implementation Notes:**

* Use a dictionary to store courses where course name is the key, and the number of credits is the value.
* Use a dictionary to store users where user name is the key, and value is a list in which password will be the first item, budget will be the second item, and the third item will be a dictionary of course names that the user takes.

**Warnings:**

* **Do not** talk to your classmates on project topics when you are implementing your projects(This is serious). **Do not** show or email your code to others (This is even more serious). If you need help, talk to your TAs or the instructor, not to your classmates. If somebody asks you for help, explain them the lecture slides, but do not explain any project related topic or solution. Any similarity in your source codes will have **serious** consequences for both parties.
* Carefully read the project document, and pay special attention to sentences that involve

“**should**”, “**should not**”, “**do not**”, and other underlined/bold font statements.

* If you use code from a resource (web site, book, etc.), make sure that you reference those resource at the top of your source code file in the form of comments. You should give details of which part of your code is from what resource. Failing to do so **may result in** plagiarism investigation.
* Even if you work as a group of two students, each member of the team should know every line of the code well. Hence, it is **important** to understand all the details in your submitted code. You may be interviewed about any part of your code.

**How and when do I submit my project?:**

* Projects may be done individually or as a small group of two students (doing it individually is recommended for best learning experience). If you are doing it as a group, only **one** of the members should submit the project. File name will tell us group members (Please see the next item for details).
* Submit your own code in a **single** Python file. Name your code file with your and your partner’s first and last names (see below for naming).

o If your team members are Deniz Barış and Ahmet Çalışkan, then name your code file as deniz\_baris\_ahmet\_caliskan.py (Do **not** use any Turkish characters in file name).

o If you are doing the project alone, then name it with your name and last name similar to the above naming scheme.

o Those who **do not** follow the above naming conventions **will get 5 pts off** of their grade.

* Submit it online on LMS by **5 pm on December 19, 2019**

**Late Submission Policy:**

* -10%: Submissions between 17:01 – 18:00 on the due date
* -20%: Submissions between 18:01 – midnight (00:00) on the due date
* -30%: Submissions which are 24 hour late.
* -50%: Submissions which are 48 hours late.
* Submission more than 48 hours late will not be accepted.

**Grading Criteria?**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Code Organization** | | | **Functionality** | | | |
| Not using meaningful variable names  (-10% cut off from your overall grade) | Not properly using functions, code repetitions (-10% cut off from your overall grade) | Insufficient commenting (-10% cut off from your overall grade) | Student  functionality with all of its submenus   (20 pts) | Admin  functionality with all of its submenus  (30 pts) | Proper use of data structures to implement the student and admin menu items  (40 pts) | Others (10 pts) |

**Have further questions?**

Please contact your TAs if you have further questions. If you need help with anything, please use the office hours of your TAs and the instructor to get help. **Do not walk in randomly (especially** **on the last day) into your TAs’ or the instructor’s offices. Make an appointment first. This is important. Your TAs have other responsibilities. Please respect their personal schedules!**

Good Luck!