**Python Programming Task**

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 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). Please strictly follow the implementation notes provided at the end of the document under “Implementation Notes”. Please use Python3.

**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: “istanbul123’’
* 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: istanbul123

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:**

The primary goal of this mini project is to practice using classes and objects. Hence, please use every opportunity to employ object-oriented programming concepts in your code. **At minimum**, you should create and use the following classes implemented with the below provided list of attributes and methods. If you see a need, you may add additional classes and/or methods/attributes that are not in the following list.

•        **Outside of class definitions, only 2 lines of code should exist** that do the following:

o   In one line you will create an object of CourseManagementSystem class.

o   In the other line, you will call a proper method of the CourseManagementSystem object that you created above to start the program.

**Everything else should be under a class.**

•        Please note that although \_\_init\_\_ methods are not specified, for every class in the above list, you should have an \_\_init\_\_ method in each class.

•        In each method of a class, self should be the first input parameter. Self parameter is not included in the above method specifications for brevity.

•        You should actively use all the methods listed under each class at least once or more. Make sure that you eliminate repeated or similar codes by defining new methods for them.

**List of Classes with their Attributes and Methods:**

* **User**
  + Attributes:
    - id
    - password
    - budget
    - registered\_courses (i.e., a list of Course objects that the student is taking)
  + Methods:
    - display: This method prints the current user’s object info (Id, budget..).Also, it should handle when to show the budget and when not to.
    - can\_take\_course: This boolean method will determine whether the current user’s budget is enough to register the selected course.
    - course\_exists: this boolean method checks whether a student has already registered to a given course or not.
* **Course**
  + Attributes:
    - name
    - credits
    - registered\_users (i.e., a list of User objects registered to this course)
  + Methods:
    - display: It should show current object’s details.
    - delete\_course: It should handle deleting courses by admin or by student, it should delete the given course.
    - course\_price: It should return a given course’s price.

      •        **Menu**

    o   Attributes:

* items: list of MenuItem objects - Each option in a menu will be represented as a MenuItem object (see the next class) and stored in a list.
* header (str): header represents the introductory text that usually appears before the menu options, e.g., “Welcome to Admin Menu! ”

o   Methods:

* display(display\_header)

Displays the menu on the screen, prompts the user for his/her selection, gets the valid user selection, and returns it to the caller. It will display the header attribute before menu items if display\_header parameter is True. Otherwise, it will not show the header.

* add\_menu\_item(text, number)

It should build and add a new MenuItem object to the list of items attribute.

•    **MenuItem**

o   Attributes:

* text
  + Stores the text of the menu item, e.g., “Display questions for the next competition.”
* number
  + Stores the number of this menu item, e.g., 2 for the above item in admin menu, which will be used when printing the menu and getting user selection.

o   Methods:

* display()

•        This method displays the current MenuItem object’s number and text properly when called.

•       **CourseManagementSystem**

* Attributes:
  + courses: list of offered  courses
  + users: dict of User objects where key would be user id, value would be the corresponding User object.
  + admin\_menu (a Menu object)
  + student\_menu(a Menu object)
  + admin\_money\_sub\_menu (a Menu object)
  + current\_user
* Methods
  + login( )

This is the central method that runs the primary logic of the system. It will call methods of other objects whenever required and handle user credentials.

* build\_menus( )

This will be called from init(), it should build the admin menu, student menu and admin sub menu as a Menu class object with all of its options as listed in MP2 manual.

* show\_admin\_menu( )

This method will control the main flow of the admin menu when user signs in as admin.

* show\_student\_menu( )

This method will control the main flow of the student menu when user signs in as a student.

Good Luck!