**Course-End Project: EdTech Backend System**

Problem Statement: SL Tech is an edtech company that provides training

programs on various technical and functional skills. They are planning to update

their learner interface to enhance the learning experience. As part of the

development team, you have to support the backend development by creating

modules to manage the credentials, courses, and other activities of learners.

Objectives:

1. To design and implement a backend system for SL Tech's learner interface

2. To create and manage user credentials, course enrollments, and assignments

3. To integrate all modules into a comprehensive backend system

Steps to Perform:

1. Create a User Class:

• Design the User class with specified attributes and methods

• Implement methods to update the email and password and validate credentials

**Remarks:**

* **The user class defines a specific user.**
* **The user class should contain a method \_\_init\_\_ that receives a user\_id, name, email and password in arguments and declares attributes user\_id, name, email and password equal to the user\_id, name, email and password received in arguments**
* **The user class should further include a method that receives a new email in argument and replaces the current email with the new email**
* **The user class should further include a method that receives a new password in argument and replaces the current password with the new password**
* **The user class should further include a method that validates credentials: the method receives a password and an email and checks whether the received password and email are equal to the password and email of the user**

**Example:  
def check\_credentials (self,input\_pass,input\_email):  
 self.email = input\_email  
 etc…  
 etc…**

In question 7, this may happen:

user1 = user(my\_name,my\_id,my\_email,my\_password)

a=input(‘Enter a password’)

b=input(‘Enter an email’)

user1.check\_credentials(a,b)

2. Create a Learner Class:

• Design the Learner class that inherits from the User class

• Implement methods to enroll in and drop courses

**Remarks:**

* **The learner class defines a specific learner.**
* **The learner class receives a user from the user class in argument**
* **The learner class should contain a method \_\_init\_\_ that receives a user\_id, name, email and password in arguments and defines an attribute equal to an empty list of enrolled courses**
* **The learner class should further include a method (enroll course) that receives the name of a course in argument and appends it to the enrolled course list.**
* **The learner class should further include a method (drop course) that receives the name of a course in argument and removes it from the enrolled course list**

3. Define an Instructor Class:

• Design the Instructor class that inherits from the User class

• Implement methods to add and remove courses taught by the instructor

**Remarks:**

* **The instructor class defines a specific instructor.**
* **The instructor class receives a user from the user class in argument**
* **The instructor class should contain a method \_\_init\_\_ that receives a user\_id, name, email and password in arguments (AND ALSO, AS ALWAYS, IN ALL CLASSES, self) and defines an attribute equal to an empty list of taught courses**
* **The instructor class should further include a method (add taught course) that receives the name of a course in argument and appends it to the taught course list.**
* **The instructor class should further include a method (drop taught course) that receives the name of a course in argument and removes it from the taught course list**

4. Define a Course Class:

• Design the Course class with specified attributes and methods

• Implement methods to add, remove, and list learners

**Remarks:**

* **The course class defines a specific course.**
* **The course class should contain a method \_\_init\_\_ that receives a course\_id, and a course name in arguments and declare 3 attributes: a course name equal to the course name in argument, a course\_id equal to the course id in argument and an empty list of learners**
* **The course class should further include a method (add learner) that receives a learner in argument and appends it to the list of learners.**
* **The course class should further include a method (drop learner) that receives a learner in argument and removes it from the list of learners.**
* **The course class should further include a method (display learners) that receives no arguments are returns the NAMES of al the learners in the list of learners**

5. Create an Enrollment Class:

• Design the Enrollment class with specified attributes and methods

• Implement methods to manage course enrollments

**Remarks:**

* **The enrollment class defines a specific enrollment.**
* **The enrollment class should contain a method \_\_init\_\_ that receives an enrollment\_id, a learner and a course in arguments and declare 3 attributes: an enrollment\_id equal to the enrollment\_id in argument, a learner equal to the learner in argument and a course equal to the course in argument**
* **The enrollment class should further include a method that enrolls the learner to the course by performing the following:** 
  + **receive no arguments (ie: only receive “self”)**
  + **add the learner to the course using the method, from the course class, that adds a learner to a course**
  + **Add the course to the courses taken by the learner using the method, from the learner class, that adds a course to a learner**
* **The enrollment class should further include a method a method that drops the learner from the course by performing the following:** 
  + **receive no arguments (ie: only receive “self”)**
  + **remove the learner from the course using the method, from the course class, that removes a learner from a course**
  + **remove the course from the courses taken by the learner using the method, from the learner class, that removes a course from a learner**

6. Integrate All Modules into the Backend:

• Design the SLTech Backend class with specified attributes and methods

• Implement methods to manage users, courses, and enrollments

• Implement methods to retrieve enrolled learners and courses

**Remarks:**

* **Create a class called “backend”**
* **The backend class should contain a method \_\_init\_\_ that receives no input (ie: only receives “self”) and declares 3 attributes: an empty dictionary of users, an empty dictionary of courses and an empty dictionary of enrollments:  
  Remark: the command my\_dictionary = {} creates an empty dictionary called “my\_dictionary).**
* **The backend class should further include a method to add a user: the method receives a user (ie: defined by the user class) in argument and adds, to the dictionary of users, an element with a key equal to the user\_id of the user received and the value corresponding to the key is equal to the user received.**
* **The backend class should further include a method to add a course: the method receives a course (ie: defined by the course class) in argument and adds, to the dictionary of courses, an element with a key equal to the course\_id of the course received and the value corresponding to the key is equal to the course received.**
* **The backend class should further include a method to add an enrollment: the method receives an enrollment (ie: defined by the enrollment class) in argument and adds, to the dictionary of enrollments, an element with a key equal to the enrollment\_id of the enrollment received and the value corresponding to the key is equal to the enrollment received as argument.**
* **The backend class should further include a method that receives a course\_id in arguments and returns the names of learners, IF ANY, that are enrolled to the course corresponding to the course\_id received.**
* **The backend class should further include a method that receives a learner\_id in arguments and returns the names of courses, IF ANY, in which the learner corresponding to the learner\_id is enrolled.**

Example: a value of a dictionary may be a class:

add\_enrollement(self, e)

self.enrollments[e.enrollment\_id] = e

7. Add a User Input Method:

• Implement interactive functions to handle user input for adding users,

courses, and enrollments

• Print the output

**Remarks:**

* **This is the main code, that can be coded in a separate line in your jupyter notebook.**
* **The main code should perform, at least, the following:**
  + **Display a menu of options to the screen, the options including:**
    - **1- Add user**
    - **2- Add course**
    - **3- Enroll learner in course**
    - **4- View enrolled learners in a course**
    - **5- View courses in which a learner is enrolled**
    - **6- Exit**
  + **Prompt the user of the code to select one of the available options**
  + **Execute the user’s selected option**
  + **Remark: depending on the selected option, you may prompt the user of the code to input more information (for e.g., if the user selects option 1, you may prompt the user of the code to enter the user name, id, email and password; for option 1, you may also prompt the user of your code to enter whether the user is a learner or an instructor; then, in response to option 1, you will declare an instance of a learner or instructor class, accordingly).**
  + **You may also add options to the menu if you want to, such as, adding or removing a course from an instructor**