**School Management System**

*[Strictly adhere to the object oriented programming specifications given in the problem statement. Template code is provided to ease the input output process. Template code will not work. You need to fill in the missing code.]*

**Business Requirement:**

Your task is to create a basic School Management System where students can register to the system and courses, and also view the courses assigned to them.

**Work-Flow:**

Only students with right credentials can login. Otherwise, a message is displayed: “Wrong Credentials!”

1. Valid students are able to see the courses they are registered.
2. Valid students are able to register to any course in the system.

**Database:**

**CSV Files**

Three Comma Separated Values (csv) files that contain columns specified in the tables below are provided. The tables are in the following format:

**Format:**

|  |  |
| --- | --- |
| **Name** | **Description** |
| The name of the column | The description of what this column contains. |

**File 1 – students.csv:**

|  |  |
| --- | --- |
| **Name** | **Description** |
| email | *Student*’s current school email |
| name | The full name of the student |
| pass | *Student*’s password in order to login |

|  |  |
| --- | --- |
| **Name** | **Description** |
| course\_id | Unique *Course* Identifier |
| name | The name of the *Course* |
| instructor | The name of the instructor |

**File 2 – courses.csv:**

**File 3 – attending.csv:**

|  |  |
| --- | --- |
| **Name** | **Description** |
| course\_id | Unique *Course* Identifier |
| email | *Student*’s current school email |

**Requirement 1:**

**Model Class: Models** module contains every model class associated with each file.

Every Model class must contain the following requirement:

- \_\_init\_\_ method that initializes every attributes with a parameter provided.

The attributes for the *Student* class are specified in **TABLE 1**. These attributes have **GETTER** methods. Each instance of the *Student* class must carry data related to only a single *Student*.

**TABLE 1:**

|  |  |  |
| --- | --- | --- |
| **Datatype** | **Name** | **Description** |
| string | email | *Student*’s current school email |
| string | name | The full name of the *Student* |
| string | pass | *Student*’s password in order to login |

The attributes for the *Course* class are specified in **TABLE 2**. These attributes have **GETTER** methods. Each instance of the *Course* class must carry data related to only a single *Course*.

**TABLE 2:**

|  |  |  |
| --- | --- | --- |
| **Datatype** | **Name** | **Description** |
| string | course\_id | Unique *Course* Identifier (ex: CS101) |
| string | course\_name | The name of the *Course* |
| string | instructor | The name of the instructor |

The attributes for the *Attending* class are specified in **TABLE 3**. These attributes have **GETTER** methods. Each instance of the *Attending* class must carry data related to only a single assignment of a *Student* to a *Course*.

**TABLE 3:**

|  |  |  |
| --- | --- | --- |
| **Datatype** | **Name** | **Description** |
| string | course\_id | Unique *Course* Identifier (ex: CS101) |
| string | student\_email | *Student*’s school email |

**Requirement 2:**

**Data Access Objects: DAO** module contains three classes, **StudentDAO, CourseDAO,** and **AttendingDAO.**

Every DAO class must contain the following requirement:

- \_\_init\_\_ method that initializes the instance attributes. Every DAO class must have an instance attribute which is a list of objects associated with each Model class. \_\_init\_\_ method should read the associated csv file, parse the data and store it into each list.

*StudentDAO* class is going to be used to search the csv files for student’s information only.

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Return Type | Method Name | Input Parameters |
| 1 | list | get\_students :  This method returns a list of Student Objects. | None |
| 2 | Student | get\_student\_by\_email :  This method takes a Student’s email as a String searches the List of Students for a Student with that email and returns a Student Object. | email |
| 3 | boolean | validate\_user :  This method takes an email and a password from the user input. Return whether or not a Student with the given information is found. | email,  password |

*CourseDAO* class is going to be used to query the database for course’s information only.

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Return Type | Method Name | Input Parameters |
| 1 | list | get\_courses:  This method returns a list of Course Objects. | None |

*AttendingDAO* class is going to be used to query the database for Attending’s information.

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Return Type | Method Name | Input Parameters |
| 1 | list | get\_attending:  This method returns a list of Attending Objects. | None |
| 2 | boolean | register\_student\_to\_course:  This method takes a Student’s email, a Course ID, and a Course List. It checks if a Student with that Email is currently attending a Course with that ID.  If the Student is not attending that Course and the Course ID is valid, then add a new Attending object with the Student’s Email and Course ID to the List and return True.  Otherwise, return False. | student\_email,  course\_id,  course\_list |
| 3 | list | get\_student\_course:  This method takes a Student’s Email and a Course List as parameters and searches the Attending List for all the courses a student is registered to.  Then the course objects that correspond to each of these are added to a new List of courses. This list of courses the Student is attending is returned. | student\_email,  course\_list |
| 4 | None | save\_attending:  This method overwrites the original Attending.csv file with the new data. | None |

**Requirement 3:**

**Main Entry:** In the module named, **MainEntryPoint**, there is a function named, **main**. When your code is completed, this function will be used to run the School Management System.

**Example Workflow:**

Welcome!

1. Student

2. Quit

Please, Enter 1 or 2: 1

Enter Your Email: J@pysc.edu

Enter Your Password: 333

Wrong Credentials!

1. Student

2. Quit

Please, Enter 1 or 2: 1

Enter Your Email: joe@pysc.edu

Enter Your Password: jc2142

My Courses:

# COURSE NAME INSTRUCTOR NAME

1 Python Basic Young

2 Jazz History Ryan

3 Number Theory Mike

What Would You Like To Do?

1. Register To Course

2. Logout

Please, Enter 1 or 2: 1

All Courses:

ID COURSE NAME INSTRUCTOR NAME

CS101 Intro Prog Mark

CS102 Python Basic Young

CS103 Java Basic James

SC105 Intro Bio Lisa

SC205 Organic Chem Jake

MS200 Jazz History Ryan

EN120 Writing Jill

MT680 Number Theory Mike

HT150 US History Matt

Select Course By ID Number: CS102

Attempting To Register...

You Are Already Registered In The Course.

1. Register To Course

2. Logout

Please, Enter 1 or 2: 1

All Courses:

ID COURSE NAME INSTRUCTOR NAME

CS101 Intro Prog Mark

CS102 Python Basic Young

CS103 Java Basic James

SC105 Intro Bio Lisa

SC205 Organic Chem Jake

MS200 Jazz History Ryan

EN120 Intro Writing Jill

MT680 Number Theory Mike

HT150 US History Matt

Select Course By ID Number: HT150

Attempting To Register...

Registration Successful!

My Courses:

# COURSE NAME INSTRUCTOR NAME

1 Python Basic Young

2 Jazz History Ryan

3 Number Theory Mike

4 US History Matt

1. Register to Course

2. Logout

Please, enter 1 or 2: 2

You Have Been Logged Out.

Closing Program. Goodbye.