# Project 5: Course Management System

Harshit Mehta (2020A7PS0057P)

Rachit Agrawal (2020A7PS0033P)

TA – Sai Teja Akalankam

This document gives a brief idea about the classes that have been implemented and their attributes and methods. It also provides a small idea about how the input and output works. There are also a few short comings of our code discussed. Along with it there is a note on hoe the code can be enhanced upon. There is also a section where we have discussed how the usage of different types of design patterns would make the code better. GitHub and GDrive Links at the end.

## 1. Description of Classes

### 1.1 **Test**

This class holds the data of a test. It has attributes such as testName , testDuration etc. which store the details of the test. There is a string variable that tells if the test is available for the student to give. This variable is under the control of the teacher. It has a constructor that assigns values to the variable when an object is instantiated. There is a ‘setKey’ attribute using which the private variable answer key is assigned its value. There is an ‘evaluatingTest’ method using which the marks are calculated by comparing the entered answers of the user and the answer key for the test.

### 1.2 **CourseDetails**

This class is a place to store the specific details of a course that is present on the Management System. It has variable to store the name of the course, units of the course the classroom where the course class will happen. Along with that there are multiple Lists one of which stores the name of all the test components that will take place in the course. There are lists that store the name and id numbers of all the students that are registered for the course.

### 1.3 **Course**

The class inherits from the CourseDetails class. It was simple to use inheritance here as the details stored in the CourseDetails can be common for multiple courses also. This has a variable to keep count of the number of students registered for the particular course. There is List of type test. This list has elements where every element stores individual details of the test that are to ake place in the course. This provides an efficient way to hold data about tests without having to create multiple redundant variables. Another one which holds all the topics that are to be taught in the course.

### 1.4 **Student**

This happens to be probably one the most the important class of the entire project as the student is the heart of any course management system. The class has variables that store the personal details of a student like the name, id, email etc. There is a Hashmap that stores the marks that the student secured in the test that takes place. Its matches the name of the quiz along with the marks of that test. It shows null on printing if the student failed to take the test in the stipulated time else shows the marks. There is a List that keeps count if a particular test has been attempted by the student or not. The constructor initializes the personal details of the students along with which it stores them in a text file. There are load of different methods I.e., functions that a student can do the system. There are methods that can print the student's personal details and the courses in which the student is registered in. There is a 'marksofTest' method that can tell a student the marks that he/she got in the any test. There is a method that can tell the student the tests that are available that he/she can give. There is a static method available that gives the list of all students that have registered for a course. There is a 'takeTest' method using which a student can take any available test. The method first gives the list of the available tests. It then gives details of test that the student has decided to take. Later it takes the input from the user about the answers that he/she wishes to enter for a test and then evaluates and prints the marks for the test.

1.5 **Teacher**

There are variables to store the name and email id of the teacher. There is an ArrayList that stores all the courses that the teacher has already taught before or is teaching at the moment. There is a method that gives the details of the course that the teacher is teaching. The teacher can see the marks of the students that are registered in the course. A teacher can create a new test in the course I.e., add details of a new test and make it available for all registered students can take.

## 2. Working of Code and Input/Output

On executing the code objects of the Course, Teacher and the Student are created. There are certain functions that are done automatically like adding the details of the students in the Course page and saving the personal details of the student in a text fil to maintain the database of all the registered students of the course. The personal details of the students that have already registered for the course are printed. The teacher creates a few tests and the user is asked to enter the answers for the questions following which the marks of the student are reflected. Then the user is asked is any new student has registered for the course. If the input is “Yes” then the user is asked to give details of the new registration after which there is a menu given to user from which multiple functions on the newly registered student can be done. If “No” then we move ahead. The teacher then creates the Comprehensive test which can be given by all the users and then the running ends by providing the final marks of all the students. Detailed explanation with visual aids is provided in video.

## 3. Known Limitations

There is no code which is completely sound proof. Even our code has a few limitations. Some of them are known to us which we can tell you about. The code has a few repetitive redundant lines of code that makes it inefficient. The code is limited to returning marks only for Quiz, Midsemester_Examination and Comprehensive Examination. It won't be able to handle a component division of

other types for now. The project works as a common point to handle the existing students and the newly registered student if any. We have capacity to handle only 1 new student for now. Another flaw is that we are unable to give the user any opportunity to work from the teacher login as the activities of the teacher are performed automatically by the system at the time of execution of the code. In spite of all of this the good part of the code is that there are extensive functions that a student can perform and the teacher can do multiple things just that user can't control the latter. However, there is always scope for improvement and the following section talks about it.

## 4. Future scope of the project

The recent crisis of COVID-19 showed the importance of developing online education. It is very important to develop a course management system which has a number of facilities. We have an in-house example of Nalanda-aws which has number of functions like taking a test, teacher can share study material on it, there can be links to various videos and material. There is also a feature to store marks of the test that a student takes. However, there is always some scope for improvement. The project we have made can be enhanced to the already existing level. We can design a feature using which we can attend an online lecture session on the page itself instead of resorting to apps such as G-Meet or Microsoft team and others. There can also be an option for the student to view the previous papers of the course from earlier semesters and view the grading statistics from the previous time the course was taken.

## 5. Analysis on the basis of Design Patterns

### 5.1 **Favour Composition over Inheritance**

In the code the CourseDetails class holds the data of the various details of the course. The Course class inherits the CourseDetails class. Here very clearly, we should have used composition I.e., make an object of the CourseDetails class in the Course class to store all the details of the class instead of inheriting the class. It's easier to use composition as we can have multiple objects of different classes in another class however, we can't have multiple inheritance. It is difficult to find

two classes with maximum matching features and much easier to have multiple objects of small classes to store the common details between different classes.

### 5.2 **Strive for Loose Coupling between objects that interact**

The teacher class has details of all students that are registered in a course that the teacher is teaching. The teacher also has all the information and control over the contents of the course class object. There is a need to add loose coupling between the teacher and course class and the teacher and the student class in order to make the code more efficient and to make it easier to sort out any maintenance and troubleshooting issues. It can also help to avoid crashing of code in case there is any unexpected change in one element.

### 5.3 **Program to an Interface not implementation**

A simple example of this is the method that all three Student, Teacher and the Course class possess I.e., the viewDetails method using which the details of the object of that class can be obtained. This method could have been put in an abstract interface which will be implemented by the classes. This would make the program easier to change and along with-it easy maintenance follows.

### 5.4 **Classes should be open for extension and closed for modification**

This pattern says that classes should be made in a way that they can be extended by any class and more attributes and methods in the child class to enhance the features however there should not any code that supports the changing of data structures of the parent class. This shows that the class that has been designed are stable. We have implemented the same in our code. All classes have stable structure and along with it all the classes are available for extension.

GitHub Link to code file - https://github.com/Rachit-0107/Course_Management_System

GDrive link to all resources-
https://drive.google.com/drive/folders/19PJdAJ8DBcBsw2E2nrXogZA7jPCUFwFH?usp=sharing