**CS 591 Final Project – Grading system**

* Overview (Manan)

High level overview of the document and what I should expect to get out of reading this document

* Scope (Manan)

Whats the problem and reason for you making this piece of software?

* Functionality (Manan,Zhelin)

What where you expecting the program to do?

* Goals (Zhelin)

What goals did you have which ones did you meet or don’t meet?

* Object Diagram (Dennis)
* Object Justification (Dennis, Zhelin)

**Core Class**

**Main**: The entrance for the Grading System. Main will create a singleton instance of class *GradingSystem* and create the very first UI for *GradingSystem* which is the login page.

**GradingSystem**: We use the singleton model for *GradingSystem* because there will only be a *GradingSystem* instance at one time. Also, it’s easy to access the singleton by using the static method *GradingSystem.getInstance().*

*GradingSystem* is responsible for the Account operation, the *LoginCheck()* and*signUp()* methods have to be within the *GradingSystem* class, it ensures the security and the data Confidentiality of the system.

*GradingSystem* class maintain the important data like all the information of courses and store all of them in the memory. When the instance is first created, it will interact with database and load the data into memory. In this way, if the database disconnects during the runtime of GradingSystem, the system won’t be affected, the data won’t be discarded.

**GradingType**: An Enum class which defines the type of grading. For now, we have two different grading type, one is *Absolute Grade* and another one is *Deduction Grade*.

**Assignment:**

**Course:**

**CourseTemplate:**

**Curve:** An abstract class, which represent a basic data structure for curve. It only holds attributes like *Amount* which represent the value of a curve. It also has two abstract methods like *getCurveString()* which output the type of curve and *ConvertRawToCurved()* which calculate the score after curved.Use a abstract super class of curve, we can make the future extension easier and reuse the code when add more types of curve.

**FlatCurve:** The class extends *Curve* and overrides the *getCurveString()* method and *ConvertRawToCurved()* method according to the definition ofa flat curve.

**PercentageCurve:** The class extends *Curve* and overrides the *getCurveString()* method and *ConvertRawToCurved()* method according to the definition ofa percentage curve.

**Grade:**

**Person**: An abstract class, which represent a basic data structure for a person. It only holds attributes like *Name* and *ID*.

**Student**: The class extends *Person*, it gives a general idea of a student. For now it only has attributes like *Name* and *ID*. The reason we keep it is for future extension. We might need to add more attributes like *BirthDate* or *Sex* in the future.

**EnrollStudent**:

**Supplementary Class**

**CustomizedTable**: This class extends *AbstractTableModel*. It overrides a method *isCellEditable()*, to create a customized Table model, that allows the table in our system can be editable in different scenarios.

**ReadExcel**: This class has a method *getSheet()* which can parse .xls file from a certain path, enable the system to load student data from Excel.

**SelectFile**: This class has a method *select(),* which allow user to choose a file from operation system and pass the path of the selected file to our grading system.

**SettingChangeListener**: An listener interface which has one method called *updatePage().* Because we separate the setting of Curve, EnrolledStudent and Assignment in different frames. These Frames need to implement this listener interface, so that when the data in other frames change, this frame will change accordingly. For example, when we add new assignment in Assignment setting page and confirm it, the grading page will need to be updated and add more columns for the new assignment to be graded.

**UUIDGenerator**: We use this class to assign a unique ID for each instance we create, so that we can easily track these instance in memory as well as in database.

We need these supplementary classes to improve the convenience and efficiency of our system. Although it’s not pivotal but it enable our system to have a better performance and usability.

**UI Frames:**

* Database (Tian)