THE UNIVERSITY OF WESTERN ONTARIO

DEPARTMENT OF COMPUTER SCIENCE
LONDON, ONTARIO, CANADA

Computer Science 2212b

Introduction to Software Engineering

Project Specification - Winter 2014

Version 1.0 – January 17, 2014

1 Project Overview

The software to be developed is a gradebook application that a teacher or professor might use to enter and track the grades of his/her students. The application should allow a user to create courses, add students, enter grades, and generate reports.

2 Explanation of Terms

Active Course
The course with which the user is currently working. At any given time in the application, there will be one active course.

Deliverables
Graded items that are completed by a student in a course. Deliverables is an umbrella term which includes assignments, exams, labs, etc.

Gradebook
A spreadsheet used to keep track of students' grades, in which each row represents a particular student in a course, and each column represents his/her grade for a given assignment or exam.

Weighted Average
Similar to the arithmetic mean (our usual definition of an *average*), except that some data points contribute more to the final average than others.

Mathematically, a weighted average is defined as:

$$\bar{x} = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i}$$

where $\{x_1, x_2, \ldots, x_n\}$ is a non-empty set of data points, and $\{w_1, w_2, \ldots, w_n\}$ is a set of non-negative weights.

For example, if we have a 20% assignment, a 35% project, and a 45% final exam, and we receive

grades of 85%, 90%, and 75% on them, respectively, then our weighted final average is:

$$\begin{aligned}\frac{(0.2 \times 0.85) + (0.35 \times 0.9) + (0.45 \times 0.75)}{0.2 + 0.35 + 0.45} &= \frac{0.17 + 0.315 + 0.3375}{0.2 + 0.35 + 0.45} \\ &= \frac{0.8225}{1.0} \\ &= 0.8225 \\ &= 82.25\%\end{aligned}$$

Weighted Assignment Average
The weighted average of only one's assignment grades.

Weighted Exam Average
The weighted average of only one's exam grades.

3 Non-Functional Requirements

1. The application must run on the Linux systems in MC 10.
   (a) A virtual machine identical to the MC 10 systems will be provided to you for convenience.
   (b) You can, however, develop outside of the virtual machine on any system, but you must ensure that the final product will run within the virtual machine (test early and test often).
2. The application must be developed in Java.
3. The application must have a Swing graphical user interface.
4. It must be possible to both compile and package the application into an executable JAR file using the Maven command `mvn package`.
5. It must be possible to run the application's unit tests using the Maven command `mvn test`.
6. Reporting functionality must use JasperReports.
7. Data must be stored using object serialization.
8. Email must be sent using the JavaMail API.
9. Quality:
   (a) User Friendliness
      i. The system must be easy to use. For example, a user must be able to navigate easily between screens, and must have the option of quitting or going back to a previous screen at all times.
      ii. Messages, as well as errors, must be reported correctly and consistently.
      iii. Keep in mind that popup messages are jarring to a user and interrupt his/her workflow. They should typically be used only in exceptional circumstances. When considering using a popup message, ask yourself whether or not there is another way to convey the same information.
   (b) Maintainability
      i. You must make sure that your code is commented (at least a description of each class and each method must be given) so that it can be modified and reused by others.

ii. You must ensure that your code is unit-tested using JUnit.

(c) Reusability

i. Where possible, you must endeavour to make your code reusable.

4 Base Functional Requirements

The requirements in this section, if properly implemented, will earn your group approximately 60 - 70% on acceptance testing.

4.1 System Startup and Exit

1. The application must be called `teamN-gradebook.jar`, where `N` is your team number.
2. It must be possible to run the application by executing the command `java -jar teamN-gradebook.jar`.
   (a) Hence, all dependencies must be contained within your JAR file.
3. The user must be able to quit the application at any time.
4. If the user adds, edits, or deletes data within the application, these changes must be stored prior to exiting the program. The data must then be retrieved and restored when the system is started once again. In other words, **there must be data persistence** in the application.

4.2 Courses

1. The user must be able to add a new course. A course should have
   (a) a course code (e.g. `CS 2212b`).
   (b) a course title (e.g. `Introduction to Software Engineering`).
   (c) a term (e.g. `Winter 2014`).
2. The user must be able to edit a course and change its code, title, or term.
3. The user must be able to delete a course.
4. The user must be able to change the active course.

4.3 Students

1. The user must be able to add a student to a course. A student should have
   (a) a first name.
   (b) a last name.
   (c) a *unique* student number.
   (d) a *unique* email address.
2. The user must be able to edit a student and change his/her first name, last name, student number, or email address.
3. The user must be able to delete a student.
4. The user must be able to import a list of students into the active course from a CSV file having the following format:

```
"09","A","1119","2","UGRD","COMPSCI","2212B","001","250000001","Smith","John","","SBS4","
    fake-jsmith@uwo.ca"
"09","A","1119","2","UGRD","COMPSCI","2212B","001","250000002","Doe","Jane","","SSBA3","fake
    -jdoe@uwo.ca"
```

- Most fields in the CSV will be unused (this is the format used by Western for class lists).

4.4 Deliverables

1. The user should be able to add a deliverable to a course. A deliverable should have
   (a) a name (e.g. `Assignment 1`).
   (b) a weight (e.g. `50%`).
   (c) a type (`Assignment` or `Exam`).
2. The user should be able to edit a deliverable and change its name, weight, or type.
3. The user should be able to delete a deliverable.

4.5 Grades

The user should be able to work with a spreadsheet in which each row represents a student, and each column represents a student's grade for a particular deliverable.

1. The spreadsheet should represent each student in the active course as a row.
2. The spreadsheet should represent each deliverable in the active course as a column.
3. Each row in the spreadsheet should display
   (a) the student's full name.
   (b) the student's student number.
   (c) the student's overall weighted average for the course.
   (d) the student's weighted assignment average for the course.
   (e) the student's weighted exam average for the course.
   (f) the grade assigned to each student for each deliverable.
4. The spreadsheet should allow the user to assign grades to a student for a particular deliverable by clicking on a cell in the spreadsheet and typing in the grade.

Note: when calculating weighted averages, if, for any given student, a grade has not been entered for a particular deliverable, then that deliverable should be excluded from the grade calculation for that student. That is, if a student received 90%, 80%, and 70% on each of three assignments worth 20%, but has no grade entered for a 30% assignment, then that student's current weighted average is

$$\begin{aligned}\frac{(0.2 \times 0.9) + (0.2 \times 0.8) + (0.2 \times 0.7)}{(0.2 + 0.2 + 0.2)} &= \frac{0.18 + 0.16 + 0.14}{0.6} \\ &= \frac{0.48}{0.6} \\ &= 0.8 \\ &= 80\%\end{aligned}$$

Notice that we do not include the 30% assignment in the grade calculation since no mark has yet been entered for it.

5 Additional Functional Requirements

The requirements in this section will earn your team more marks on top of the base grade indicated in the **Base Functional Requirements** section. You should try to complete all of these, but at least complete one or two.

We will not provide individual breakdowns of the marks available for each feature so that you can pick and choose which ones to complete. It is up to your team to prioritize features. Requirements are often vague and the customer does not always know exactly what he/she wants, so, when we are strapped for time, we often have to prioritize work and determine which features will best meet the customer's needs.

5.1 Reporting

1. The user should be able to generate a PDF report for a student containing
   (a) the course code, title, and term.
   (b) the student's full name, email address, and student number.
   (c) the grades received by the student for each of the course deliverables.
   (d) the class average for each of the course deliverables.
   (e) a bar chart (or other type of chart *clearly* representing the same information) with
      i. the course deliverables across the X axis.
      ii. a scale from 0 - 100% across the Y axis.
      iii. a bar representing the student's grade for each deliverable.
      iv. a bar representing the class average for each deliverable (in a different colour).

5.2 Email

1. The user should be able to select one or more students and send them an email containing
   (a) each student's PDF report as an attachment (if your team completed the Reporting section); or,
   (b) all of the information in the PDF report, except the bar chart (if your team did not complete the Reporting section).

5.3 Grade Import/Export

1. The user should be able to import grades into the course from a CSV file.
   (a) The CSV file will contain a header row listing the columns, followed by rows listing student grades. For example:

```
"Student Number", "Final Exam", "Assignment 2", "Assignment 1"
250000000,0.85,0.90,0.98
250000001,0.92,0.87,1
```

   (b) The student number column must be present, as it uniquely identifies a student.
   (c) Deliverables will be referenced by name, but may not necessarily be in the same order in which they are displayed in your spreadsheet.
   (d) The CSV file may contain a strict subset of the course deliverables (i.e. not all deliverables may be present as columns in the CSV file).
   (e) All other columns may be ignored.

(f) An error message should be displayed if the student number column does not exist.

(g) If a student is present in the CSV file, but does not exist in the course, an error message should be displayed, but the rest of the grades should still be imported for valid students.

2. The user should be able to export grades from the course to a CSV file.

   (a) The CSV file will contain a header row listing the columns, followed by rows listing student grades.

   (b) The format of the CSV file should be as follows:

```
"First Name", "Last Name", "Student Number", "Email", "Deliverable 1", "Deliverable 2",
    ...
"John", "Smith", "250000000", "fake-jsmith@uwo.ca", 0.9, 0.98, ...
"Jane", "Doe", "250000001", "fake-jdoe@uwo.ca", 0.92, 1, ...
```

6 Other Notes

- This application is data-driven. It is **very** important to have persistent data. Otherwise, the acceptance testing will not run smoothly. **Applications without persistent data will not receive more than 50% on acceptance testing** (and will likely receive considerably less, since many things will not work properly without persistent data).

7 Revision History

Version	Date	Description
Version 1.0	January 17, 2014	Initial release.