COSC2101 Software Engineering Process & Tools

Assignment 1 (iteration 1)

Deadline: 11.59 pm Tue of week 5 (Mar 19, 2013)

1 Overview

In this assignment, you are to develop **QuickManage** - a Java desktop program that manage the daily operation of a local music and art school in HCMC. This document describes a subset of the requirements of the complete program. Additional requirements will be added in later assignments.

Note that the requirements given here are user requirements (high-level) that are intentionally vague, incomplete and may contain conflicts. The system requirements are not yet available. Part of your responsibilities is to figure out the system requirements by analyzing the user requirements carefully, perform relevant research, and asking the client representative (acted by your lecturer) appropriate questions to discover and specify the system requirements.

Despite consisting of only partial requirements in this assignment, you are expected to apply Rational Unified Process (RUP) and other relevant practices to complete a proper release by the end of the iteration. This means that at the end of this iteration, your program must:

1. Be able to run and met the given set of requirements.
2. Be easy to use (i.e. have an intuitive user interface).
3. Have an appropriate level of acceptance testing. This means you should have checked if your system satisfies the given requirements. Any defects discovered should be reported as a ticket in unfuddle.com and get fixed.
4. Have all of the required documents.

2 User Requirements

2.1 Functional Requirements

There are two types of users for this program: manager and staff. Users must login in order to use the program. By default, there is one manager account the program initially with the default username/password as "manager/manager" but additional manager accounts can be created later. A manager can use all features of the program but a staff can only use a subset of those features.

Features for manager:

- Add an account
- Edit an account
- Deactivate/activate/delete one or many accounts. However, a manager can't deactivate/activate/delete his own account.

Features for staff:

- Add a teacher
- Edit a teacher
- Deactivate/activate/delete one or many teachers
- Add a class (which is taught by a teacher)
- Edit a class
- Copy a class
- Delete a class
- Add a student
- Edit a student
- Deactivate/activate one or many students
- Enroll students into one or many classes

2.2 Non-Functional Requirements

- Persistent storage: program data shall be automatically saved as binary-format files in a default data folder right after each operation that changes the data is completed to avoid losing data. The data folder should be stored inside the folder containing your source codes. When the program starts, data shall be automatically loaded from the data folder to the program.
- User guide: The program shall display a User Guide in HTML format in the default browser of the computer (be it Internet Explorer, Firefox, Chrome, or Safari) when the user presses F1 or click at an appropriate item in the menu and tool bar. The User Guide contains acknowledgements and detailed instructions on how to install and how to use each function of your program. Relevant screenshots should be included to help the users learning to use the program easily
- Safe handling: For each of the above feature, the program shall be able to detect invalid inputs or unexpected sequence of actions and handle them in a user-friendly fashion. The program should not stop working or crash because of invalid inputs or any other reasons.
- The program works on the following hardware and OS: minimum Pentium 3, 1 GB RAM, 1 GB free hard disk space, Windows XP/7 or Linux Fedora 17/18 or Linux Ubuntu 12.04/12.10.
- Source codes are written in Java and able to load, compile and run in NetBeans 7.2.
- The GUI shall be intuitive and easy-to-use with a standard menu bar and a tool bar. The GUI is optimized for a resolution of 1024 x 768 pixels.
- The program applies the MVC pattern correctly and makes good use of OO concepts such as abstract class, inheritance, polymorphism, interface, etc. where it's appropriate to achieve the best results. In addition, the program design will be robust, high cohesion, and low coupling.
- Use RUP, SVN, Unfuddle and other relevant tools.

3 Submission

3.1 Deadline & Submission

The deadline of this assignment is **11.59 pm Tue of week 5 (Mar 19, 2013)**. By the deadline, the Team Leader must submit to Blackboard the following items in a zip file:

1. **README**: contains acknowledgments, references, descriptions and locations of the required documents and source codes in the submitted folder. In addition, it must contain the list of team members, their roles, responsibilities and percentage of effort (e.g. Minh Nguyen 25%) in the assignment as well as other relevant information. The sum of all efforts must be 100%. Because each member has a different level of knowledge and skills, effort in this context should be roughly understood as the number of working hours rather than the amount of work accomplished by each member. What we want in this course is that each member should put in a similar number of working hours into the assignment rather than letting a few members doing most of the work. The individual mark received by each member may be adjusted from the team mark base on his/her effort percentage.
2. **Software Requirements Specification (SRS)**: a PDF file containing the Use Case Diagram, all Use Case Specifications, and all non-functional requirements.
3. **Software Architecture Document (SAD)**: a PDF file containing the package and class diagrams of your design model and one page explanation of your design. Here you should explain issues such as your choice of one data structure over another, the reason for selecting a particular type of relationship between two classes, the reason for using an inheritance hierarchy, an interface, an abstract class, etc. In addition, explain how well your design follows the MVC patterns and the two design principles: low coupling (between classes) and high cohesion (between methods in a class)
4. **Source codes**: must be loaded, compiled and run in NetBeans 7.2. The source codes submitted must not contain any metadata from SVN.
5. **User Guide**: must be written in HTML but it should be formatted to allow pretty printout without breaking the pages. The User Guide contains acknowledgements and detailed instructions on how to install and use the program. Include relevant screenshots for the usage instructions to help the end-user following your instruction easily.

For your convenience, please use the simplified templates of SRS and SAD provided for you in Blackboard. More information on RUP and its templates can be found at:

- http://www.ts.mah.se/RUP/RationalUnifiedProcess/
- http://www.ts.mah.se/RUP/RationalUnifiedProcess/wordtmpl/index.htm

For User Guide, please refer to a sample user guide from Google Earth:

- http://earth.google.com/support/bin/static.py?page=guide_toc.cs

You should use StarUML or other suitable UML tools to create the original UML diagrams for your documents. However, the diagrams must be eventually exported and embedded into the documents in JPEG, GIF or PNG to ensure that your work can be assessed. Make sure that your diagrams are clear and large enough for the marker to read.

Put your submitted files into a folder and compress it into a single **.zip** file before uploading it to Blackboard. **DO NOT** use any other compression formats. Use of other formats (e.g. RAR, 7zip, etc.) may lead to delays in marking and/or a deduction of assignment marks. There should be only one submission per team.

In addition, a demo session will be scheduled with each team. The Statement of Authorship is to be submitted at that time. During the demo, the lecturer may ask any team members about your program and its development.

3.2 Marking Guide

Criteria	**Marks**
SRS (SMART requirements)	10
SAD (apply MVC correctly, good package and class diagrams for the design model, sound design justifications)	10
User Guide (complete, good layout and structure, clear and easy to follow instructions, etc.)	5
GUI (intuitive and easy to use) and Features (all requested requirements meet client's expectation) &	60
Code Quality (following strictly to a naming conventions and coding standard, adequate inline comments, code robustness)	5
Tools Usage (make good use of SVN, Ticket, Notebook, NetBeans, and other relevant tools in the project. Note: if you don't use SVN extensively in this assignment, you will receive zero for the whole assignment.)	10
Total	100

3.3 Plagiarism Notice

This is a teamwork assignment. While discussion and assistance between teams is encouraged, your attention is draw to the RMIT policies that outline the requirements and penalties for submission of work that is not your own. Any assistance received with the assignment must be acknowledged in the README file and in the code if relevant. Discussion about program features, design, implementation and quality issues is encouraged but exchange of files and/or copying of designs are regarded as plagiarism and will be penalized to the maximum extent.

Students should be aware that excessive collaboration can be a problem and can lead to disciplinary action being taken. Excessive collaboration can occur when students work on separate pieces of the assignment and share the resultant design/code to each to submit as a whole assignment. Consult your lecturer for advices on this issue when you are in doubt.