COSC2101 Software Engineering Process & Tools

Assignment 2 (iteration 2)

Deadline: 11.59 pm Sun of week 8 (Apr 14, 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

The main objective of iteration 2 is to fine-tune the program not only to meet all the requirements in iteration 1 but also to make it extremely easy-and-convenient for the user to use. It is expected that you may have to perform major rework for your GUI in order to achieve this objective. In addition, some of the previous requirements will be changed and a few new requirements are added in this iteration.

- Username: program shall not generate username automatically. Instead, a manager will input username manually to add or edit an account. Username contains alphanumeric characters only and must be unique.
- Phone number: to avoid user from making mistakes when entering phone numbers, user must select an area code or mobile network code from a drop-down list and then type 8 or 7 digits after that accordingly. The digits are separated by a dash (-) as follow: (08) 1234 – 5678, (0126) 123 – 4567. You must provide appropriate input boxes to allow user to type the digits without typing the dash. For now there is only one area code (08) but many

mobile network codes such as (090), (091), (092), (0126), etc. You have to research and get all the mobile network codes into your list.

- Address: address should be given enough space to be entered. A space of 3 or 4 lines is expected.
- Contact person: each student must have a contact person with the following information: name, phone, email, address, relationship (e.g. father, mother, uncle, grandmother, etc.). Phone and email should be validated the same way as for users, teachers and students.
- Teacher: timetable for a teacher is showed in the GUI of a teacher.
- Student: timetable for a student is showed in the GUI of a student.
- Room: timetable for a room is showed when a room is selected.
- Enrollment: enrollment now is only performed from the GUI for a class. User can select students to enroll or un-enroll from a class and see the class list updated accordingly. Since the number of student can be large, a search function to search student by name should be provided so that user can select students to enroll into a class easily.
- Invoice: user (both staff and manager) can generate invoices for one or more students. However, only manager has the right to delete invoices. Once generated, an invoice will be saved automatically and is associated with a student. User can view the list of all invoices (newest invoice is listed first) and each invoice in detail for each student from the GUI of a student record. Once an invoice is paid, user can mark the invoice as 'paid' with a paid date, a paid method (cash, bank transfer, or credit card), and a paid note to store any note that user may want to remember. In addition, manager can change an invoice from 'paid' to 'unpaid' but staff can't.
- Report: manager can generate a monthly report for any month from the past to the current month. A monthly report hast two parts. The first part is about students who had paid the tuition fees. It contains a list of student with their paid amounts, paid dates, paid methods, paid notes for the selected month, and the total paid amount. The second part is similar to the first one but it's for the students who haven't paid. Once generated, a report can be printed out to a printer. However unlike invoices, reports are not saved after generated. Manager has to re-generate a report if he/she wants to view it again in the future.
- Write unit test for all classes in the Model package and its sub-packages, as well as the classes that read and write to files.
- Provide an executable jar file to allow users to run the program directly without the need to include additional libraries or to compile the code. You can't assume that the client's computer has the software or libraries that you have in your computer.

3 Submission

3.1 Deadline & Submission

The deadline of this assignment is **11.59 pm Sun of week 8 (Apr 14, 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 code**: must be loaded, compiled and run in NetBeans 7.2. The source codes submitted must not contain any metadata from SVN.
5. **Executable jar file**: the program can be run using this executable jar file. There should be no need to load the source code into NetBeans to compile and run.
6. **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)	5
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)	15
Features (all requested requirements meet client's expectation)	40
Unit test	15
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.)	5
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.