Software Development Project:
Iteration 1, Starting the Implementation

This is worth 20% of your final project mark.

Due: Monday Feb 15, 2016 by 4:00 PM

In iteration 1, you will begin implementing the project you planned in the previous iteration. This will include the business objects, a stub database, unit tests, and the GUI for at least one big story.

You need to start using GIT for version control and set up a public GIT repository for your team (all the team members may have their own account and fork this official repo). By the deadline, your project on GitHub should include all the recent files for the iteration 1.

Each group will also keep a log, which describes their progress and meeting summaries. The log may be updated by the individual team members. You should keep the log in a text file log.txt alongside your code (the log file also goes through a proper versioning).

Process requirements:

Divide your higher-priority user stories into developer tasks and allocate time to those tasks. Do not (gratuitously) exceed the time available for this iteration. Task assignments, estimates, and actual time spent should be detailed in your log. Also in the log, include information about team meetings and the rationale behind any major design decisions. Use the log to detail who worked on each task, and approximately how long was spent on each. Also include the developer tasks from your plan (regardless of whether you completed them all).

Revise your plan from iteration 0 accordingly. Include both old and the new version of your plan. You will be discussing these changes during the project seminars at the end of the course.

You need to commit frequently to the repository. There may be penalties for big commits and only commits close to the deadline. Each team member is supposed to contribute directly into the team's GIT repository. The contributions will be visible by commits and also in the issue tracking feature of GitHub. Low contribution of a team member will affect the individual's mark not the team mark.

Implementation requirements:

For this iteration, you must implement all classes that represent the domain specific components of your system; for example, the student records system requires objects for students, courses, and registrations. It must specify an interface to the database, plus a

stub implementation that uses non-persistent storage (say, an ArrayList) with a set of initial contents. It will also need additional classes to connect components in the system and for calculations or logic.

You must implement automated unit tests for your domain-specific and business logic classes using the JUnit test framework. Construct a very thorough set of unit test cases as described in class (introduction to testing week). Ensure that your code passes all tests. TDD is recommended but not required.

You must also implement a GUI. It must satisfy the requirements of at least one of your big user stories. You can (and should) implement additional stories as time permits.

Use appropriate coding standards and simple design style (e.g., no unnecessary generalization). Ensure everyone on the project uses the same coding style. There should be no dead code or TODOs. Divide the project into packages that match your architecture. Remember, all code must be local, there is no network connection.

Documentation requirements:

Create a Wiki for your Github repository to describe the content of your submission. Identify the packages and major source code files. Sketch out the overall architecture of your system (internal and external components with their dependencies). Include a copy of the sketch with your submission.

Evaluation:

Your work for this iteration will be marked out of 20. Marks will be allocated roughly equally to four areas: a) functionality, b) implementation quality, c) the quality and thoroughness of your unit tests, and d) process/release/documentation details.

There will also be additional substantial penalties for missing elements (log files, developer task lists, etc.), broken releases (that do not compile), or compile/run-time errors or meaningful compiler warnings.

Hand-in:

The hand-in is through Github (i.e., no UMLearn submission, or email, etc.). Your project history up until Itr1 should be accessible at Github. **The only thing you need to submit to UMLearn is the location of your team's Github repository**, which should be accessible to TAs and me (as **contributor**). Our Github accounts:

- Me: hemmati
- TA1 (Moein): moeinalmasi
- TA2 (Taha): trsiddiqui

Your repository should contain all the files necessary to run/test your program and the corresponding documentation by the deadline. These include:

- Source code

    - all source files in a folder called src (each layer's classes in separate package)
    - all test files in a folder called test with the same organization as source files
    - a lib folder that contains any necessary jar files
- Documents
    - Wiki
    - Architecture sketch
    - Log file (including meetings minutes, rationale behind changes on plan and big design decisions, any concern with the project or group members, task assignments, the development tasks per user story and the times originally allocated per development task and the actual time spent on each task)
    - Old planning document (from Itr0) and the modified version of it (containing any change to the plan that happened during Itr1).

You may develop your code on any platform. The project must contain all the files it needs to execute and run on a "fresh" development computer (running Windows 10, with Java and the Android SDK and tools installed and a Nexus 7 emulator running), it should take nothing more than selecting "Build" and "Run As..." to execute it on the emulator. Similarly, it must deploy and execute on a real Nexus 7 tablet. All tests need to run and pass as well.

A snapshot of your repository will be taken by 4:00 PM on the due date, or shortly after, for marking. So change after that time will be considered as Itr 2 work and will not be considered as Itr 1 contribution.