The Student Learning Access Portal

[*Please note that, as before, the available description is deliberately intended to be sketchy and imprecise.*

The necessary details are to be supplied by the USER upon request from DEVELOPER(S).

The USER is free to add any detail she or he wants; but should keep in mind that iterations typically last for two weeks only, hence the amount of work that can be reasonably expected to be finished is necessarily somewhat limited.]

The Student Learning Access Portal (SLAP) Interface software is intended to enhance administration of courses at the Church Street University by facilitating communication, coordination, and assignment submission among students, and between students and their instructors. Students and instructors need a flexible project management system that will allow administrators to create courses and manage the assigned instructors and students for each course; instructors to create assignments, perform evaluations, and communicate with students; and students to view assignments and grades, submit work, and communicate with other students as well as with instructors. Also, an announcement page is displayed for all users upon login which will display any SLAPs they have been sent since their last login. A SLAP is a message sent from one user to one or more other users.

Your job is to design, implement, and test a software application to help the users (students, instructors, and administrators) meet these goals. You are more than welcome to rely on your student experience in making this application as useful as possible, for both students and professors.

As always, any specific requirements might change along the way – in fact, at any time during the development process.

Instructions for XP stream

A. Review Requirements

All team members should review the synopsis of the chosen project and get a clear idea of what the software is supposed to do.

B. Prepare Project Team

XP User

Choose a team member to play the role of the USER; she or he will represent the other users of the proposed station and is NOT allowed to participate in code development. Instead, she or he will give the development team specific information about what the software should look like and how it should work. The USER will prepare the stories, assign priorities, and decide which ones to develop.

The development team should ask the USER all questions related to what the proposed system needs: what the software screens should look like, what specific rules to follow in processing data, what are some examples of songs they use, and so on.

During development, the USER should be available (in person, preferably, or through email or chat) to provide guidance to developers should they need it.

Team Leader

Choose a team member to play the role of the software development TEAM LEAD.

The TEAM LEADER assists other developers with the technical and coding details of software development. Other developers should ask the TEAM LEADER for help with IDE setup, development language use, code deployment, and so on.

Developers should not ask the TEAM LEADER how to implement a specific user requirement, however, as this is the job of the individuals working on the task.

The TEAM LEADER is also a developer who takes part in a programming pair.

Programming Pairs

Team up with another team member in a programming pair. Work with your partner to complete the development tasks that produce the desired software. Team lead should participate in a pair, perhaps not all the time.

XP Team

This procedure will give you the XP team consisting of one USER and three to five developers, one of which is the TEAM LEADER. The developers can form two programming pairs.
The roles of USER and TEAM LEADER should, ideally, be rotated from one iteration to another; but this is up to the team to decide.

C. Set Up Development Environment

You must set up a development environment that all developers can use to produce the software. The TEAM LEADER should assist team members with this. Developers will produce different parts of the software. You must integrate all software parts so that they compile and work together. You must then build an executable which the USER can test to ensure it meets the requirements.

For the project, you may use any development platform. Suggested choices include

- Java on Eclipse IDE, using JUnit for unit tests and Ant (or equivalent) for the build engine; or
- C#, C++, or Visual Basic (.NET) on Visual Studio IDE, using VS Test Project for unit tests and VS .Release Build as a build engine.

Other choices are also possible, depending on the team's preferences. Nonetheless, it is to your benefit to use an IDE that provides you with drag-and-drop capabilities for building screens, as well as code generators for classes and other parts of an application.

As CPS406 is a software engineering course, rather than a programming course, you are free to use any development tools you prefer in order to speed up the software development process. However, I would STRONGLY recommend that you stay within the confines of the tools (programming language, first and foremost) taught in Computer Science courses at Ryerson.

D. The XP Development Process, step by step

1. The user should:
   - Write user stories from the requirements.
   - Assign a priority (high, medium, low) to each story.

2. Then, the development team should:
   - Break down stories into programming tasks. A story may become many tasks. On the other hand, if stories are very small, these stories can be merged into a single task.
   - Estimate the time needed to implement each task (e.g. 10 hours).
   - Estimate the risk or uncertainty associated with the time estimates for each task (e.g., 10 hrs ± 2 hours).

3. Then, the user should:
   - Review time estimates and risks for each task.
   - Choose the stories to be completed in the first iteration.

4. Then, the programming pairs should:
   - Pick the stories to implement. Note that all stories for the first iteration must be distributed among the two programming pairs.
   - Review the tasks associated with the story.
   - Start implementing those tasks using test-driven development (i.e. write the test that the code must meet, and only then code the software that meets the test).

5. Then, the user and development team should:
   - Review results after the iteration time is over.
   - Deal with stories as defined by XP rules for unfinished stories, story re-prioritization, and so on.
   - Perform new iterations (step 4) until all stories are implemented or you run out of time.

E. Deliverable and Marks

Please see the XP Process Deliverable template below for an example of what to hand in. Use landscape mode in your Word document to ensure there is sufficient space for all necessary information.

To submit the Iteration deliverables, you should do the following:

1. The USER will prepare the delivery package with the following contents:
   - A 'ReadMe.txt' file that describes the delivery package.
   - The XP Process Deliverable document, using the template below.
   - Source code of the part of the project that has been implemented and WORKING at the time of submission.
   - Test suite used for unit tests.
   - The executable (which is supposed to be actually capable of being executed!) obtained from the source code.

   The package should be submitted through Blackboard.

3. This process is to be repeated for iteration 3.

4. Similar process is to be repeated for iteration 4, with the following differences:
   – It is expected that the ultimate result of iteration 4 is a functionally complete and thoroughly tested executable. However, you may want to check the slides entitled p30twenty.pdf for a few hints.
   – Test suite should include acceptance tests as well.

5. Immediately after handing in the iteration delivery package, each team member should individually submit a self/peer assessment survey through Blackboard. This holds for each iteration.

6. Your project deliverable for an iteration is worth 10 percentage points of your final mark. All team members receive the same mark.

F. A Word on Iterations 3 and 4

Team composition is expected to remain fairly stable throughout iterations 2, 3, and 4, although some minor shuffling is still possible.

F. Template for XP Process Deliverable (repeat for each iteration)

Requirement	Story	Initial priority	Task	Estimate	Risk	Iteration details
<description of requirement 1>	<description of story 1>	<Low, Med, High>	<description of task 1>	<time to complete>	<± time>	<initial iteration; results after iteration; what happened next to the story>
			…	…	…	
	…	<Low, Med, High>	…	…	…	…
			…	…	…	
<description of requirement 2>	<description of story 1>	<Low, Med, High>	…	…	…	…
			…	…	…	
	…	<Low, Med, High>	…	…	…	…
			…	…	…	
…	…	…	…	…	…	…