LAB 4: Implementation, Testing, and Preparation for Demo

1. OBJECTIVES

1.1 Implement your application
1.2 Design test cases using black box and white box testing techniques
1.3 Plan your live demo
1.4 Prepare SDLC work products for delivery

2. INTRODUCTION

2.1 The design model from Lab#3 must now be implemented in a working application.

2.2 The initial application skeleton needs to be further refined (i.e. some more code to be written) to accommodate additional behavior such as error handling and optimization.

2.3 The class' implementation in code should be traceable to its design and requirements.

2.4 As the coding gets underway, the test cases are developed in parallel. The inter-mingling of code-test activities helps build confidence as working code snippets are systematically built into a sizable piece of code.

2.5 The implemented web application should be deployed in a cloud platform.

2.6 The implementation and testing phase of the Software Development Life Cycle should be completed by your Lab#5 in which you will demonstrate your working application. Note that if you do not have time to implement all functionality of your design you can just implement and demonstrate the main functionality in the demo. Your design should be complete but there is no need to implement all the functionality if your schedule becomes tight.

2.7 It is now time to prepare the product for delivery and presentation to the client.

As the time allocated to the live demo is limited (15 minutes per team), prepare the material and sequence of the demo carefully, so that the salient features of your application may be highlighted to the client, amongst many competing products.

As an indication of the style the demo should take, it is good to imagine that you are presenting a prototype of a product to a customer who will choose one of the prototypes and invest in its further development for full implementation. Among several companies presenting to the customer, the customer will select the one that they consider meets their requirements and can be extended to meet their full system requirements. Meanwhile, the technical capability of the company is also an important factor to determine the winner.

3. PROCEDURE

3.1 *Implement your application*

3.1.1 From the design class diagram, generate the skeleton code. Implement the behavior as documented in the sequence diagrams and dialog map.

3.1.2 You may wish to organize the code into packages or folders, following the stereotypes in the design class model.

3.1.3 Document the classes and their **key public methods** to convey design intent and usage.

3.1.4 Use the SVN repository to facilitate team collaboration.

3.2 Deploy the implemented application to cloud.

3.3 *Design test cases using black box and white box testing techniques*

3.3.1 Design test cases using equivalence class and boundary value testing techniques to test **one important control class** that implements some important application logic according to the requirements specification.

3.3.2 Design test cases using basis path testing techniques to test **two methods** that implement complex application logic.

3.3.3 Minimize the number of test cases through the careful selection of test input using proper black box and white box testing techniques learnt in the course.

3.3.4 Execute your test cases and document the testing results.
Note that you can develop automatic test cases using proper automatic testing framework, for example, http://developer.android.com/training/activity-testing/activity-unit-testing.html, or https://phpunit.de/.

Or you can perform manual testing guided by your test cases.

Test Input	Expected Output	Actual Output

3.4 *Plan for live demo*

3.4.1 Craft the demo script and sequence so that you maximize the opportunity to present your product to the client.

3.4.2 The client is eager to see how well the various specified functionalities can be carried out in your application. Demonstrate the various usage scenarios by the different end-users.

3.4.3 In the presentation, highlight innovative solutions in the application, as well as, elements of good software engineering practices and system design.

3.4.4 Apart from the live demo, use a screen capture tool to prepare a five-minute screen video of the application. You can provide voice-over to explain the functionality. Remember that the live demo cannot be repeated and could go wrong. The video can be seen as a second (and different) opportunity to demonstrate or demo your software which is under your control and will not go wrong.

3.5 *Prepare SDLC work products for delivery*

3.5.1 Re-visit the documentation from previous lab sessions to ensure that it is complete and sufficient for delivery to the client. In particular, document the traceability from requirements to design, then to implementation and test cases.

3.5.2 The author(s) for each work product must be clearly indicated.

4. **DELIVERABLES**

- Working application prototype
- Source code
- Test Cases and Testing Results
- Demo script

5. **REFERENCES**

- Object-Oriented Software Engineering: Using UML, Patterns and Java – B. Bruegge and A. Dutoit
  - Chapter 10 – Mapping Models to Code
  - Chapter 11 - Testing
- Javadoc
  - http://en.wikipedia.org/wiki/Javadoc#Structure_of_a_Javadoc_comment
  - A more technical treatment on the Oracle Javadoc Home Page - http://www.oracle.com/technetwork/java/javase/documentation/index-jsp-135444.html
- JUnit
  - See section 4 of http://supportweb.cs.bham.ac.uk/documentation/tutorials/docsystem/build/tutorials/junit/junit.pdf
  - A more technical treatment at http://junit.sourceforge.net/junit3.8.1/doc/cookbook/cookbook.htm