Approval: 9th Senate Meeting

Course Name:	Large Applications Practicum
Course Number:	CS308
Credits:	0-0-3-2
Prerequisites:	IC150
Intended for:	UG
Distribution:	Compulsory for CSE; CS elective for EE and ME
Semester:	6th

Preamble:

The new curriculum calls for a sequence of 3 Practicum courses for CSE, viz. CS207 Applied Databases Practicum, CS307 Systems Practicum and CS308 Large Applications Practicum. Now, the networks material is shifted to CS307 and the database material to CS207 with a focus on developing database applications. The main focus of CS 308 is to learn the mechanics of building, testing, versioning, and reversing of large software applications. Furthermore, this course will expose students to object-oriented design using the UML notation.

Tentative sequencing:

Semester 3 – CS207 Applied DB Practicum, followed by CS204 Introduction to Databases

Semester 3 – CS307 Systems Practicum, followed by CS3xx Introduction to Distributed Communicating Processes electives on networks and OS

Semester 6 – CS308 Large Applications Practicum, in conjunction with courses like compiler construction, etc.

Course Outline:

The students will learn the mechanics of building large software applications using object-orient languages. Topics covered in this course include: Writing Makefiles and use of Make to compile large programs; source code revision control; documentation generation from code; systematic and organized approaches to software testing; and. introduction to software testing tools. Also, this course covers certain software utilities that help write very fast parsers for almost arbitrary file formats: Flex and Bison. Furthermore, this course exposes students to use of UML notation for object-oriented design. The course concludes with an assignment on reverse engineering of a large open-source software application.

Modules:

A few lab lectures (8 hours spread over the semester):

- Overview of the Make utility (Makefiles, writing rules, use of variables, Conditionals, Functions, running make)
- Source code revision control (version control basics, introduction to some basic version control systems like CVS, SVN, and Git)
- Introduction to document generation from annotated source code (with specific focus on Doxygen, TwinText, and Natural Docs)
- Software testing (introduction, need for software testing, types of tests, test case design)

- Software testing tools (introduction to some basic testing tools for object-oriented languages – e.g., Jester for JUnit)
- Introduction to conventions of fast parsers for context-free grammars: Flex and Bison
- Overview of diagrams in the UML notation (also, how UML translates to programming structures in certain object-oriented languages like C++/Java)
- Introduction to software reverse engineering

Lab assignments (listed below) require 3 hours in the lab, preceded by at least 3 hours at home. The weekly assignments would be targeted at mastering the concepts covered weekly in the course:

- Week 1-2 Use of Make and Makefiles for object-oriented programming languages.
- Week 3-4 Use of a versioning system (e.g., Git, SVN).
- Week 5-6 Document generation from annotated source code using one of the open-source software (e.g., Doxygen, Natural Docs).
- Week 7 – 8 Software testing and test-case design; use of open-source software testing tools to test object-oriented code (e.g., Jester).
- Week 9-10 Use of parsers for parsing context-free grammars (using Flex/Bison)
- Week 11-12 Application of UML notation and diagrams for object-oriented design.
- Week 13 Reverse engineering of an existing open-source application (using certain reverse engineering tools)

References:

1.GNU 'make' pages (http://www.gnu.org/software/make/manual/make.html)

2.Boston University's Make Tutorial (http://www.cs.bu.edu/teaching/cpp/writing-makefiles/)

3.Emory University's Make Tutorial (http://www.mathcs.emory.edu/~cheung/Courses/255/Syllabus/1-C-intro/make.html)

4. Introduction to CVS, SVN, and Git (http://www.linuxdevcenter.com/pub/a/linux/2002/01/03/cvs_intro.html; http://subversion.apache.org/; http://git-scm.com/)

5.Manual pages for Doxygen and Natural Docs (http://www.stack.nl/~dimitri/doxygen/; http://www.naturaldocs.org/)

6.Introduction to Software Testing by Paul Ammann and Jeff Offutt. Cambridge University Press, February 2008, ISBN-13: 9780521880381

7.Carnegie Mellon University resources on Software Testing Tools (https://www.ece.cmu.edu/~koopman/des_s99/sw_testing/; http://mcahelpline.com/tutorials/testing/testing.pdf)

8.Software Testing Foundations: A Study Guide for the Certified Tester Exam (Rockynook Computing). Andreas Spillner, Tilo Linz, Hans Schaefer. Rocky Nook. 2011.

9.Flex & Bison: Text Processing Tools. John Levine. O'Reilly Media; 1 edition (August 21, 2009)

10.GNU resources on Flex and Bison (http://www.gnu.org/software/bison/; http://flex.sourceforge.net/)

11.UML and object-oriented design (http://www.agiledata.org/essays/objectOrientation101.html)

12.C++ Unleashed by Jesse Liberty, Vishwajit Aklecha. 1998.

13.Reversing: Secrets of Reverse Engineering. Eldad Eilam. Wiley. 2005.