SE 461 Software Testing and Quality (3 units)

Course Description
Theory and practice of determining whether a product conforms to its specification and intended use. Methods of testing, types of testing, verification and validation, software quality assurance methods, test plans and strategies, unit level and system level testing, software reliability, peer review methods, security-related quality assurance processes and techniques, and configuration control responsibilities in quality assurance.
Prerequisite: SE 451

Course Materials
Software Testing and Quality Assurance: Theory and Practice
by Kshirasagar Naik and Priyadarshi Tripathy

The course can be taught with assigned readings in the textbook as a series of lecturers and activities in:
1. Software Craftsmanship
2. GIT Basics
3. Unit Testing
4. Test Driven Development
5. Code Coverage
6. Mocking
7. Integration Testing
8. Performance Testing
9. Localization
10. Metrics
11. Test Plans
12. Behavior Driven Development

Learning Objectives
Students that successfully complete the course will be able to:
1. Identify, analyze, and apply skills and professional standards in software verification and validation.
    - Verification and validation terminology and foundations for software (e.g., planning the verification and validation effort, documenting verification and validation strategy, including tests and other artifacts, metrics and measurement, verification and validation involvement at different points in the life cycle, etc.)
    - Reviews and static analysis for software (e.g., peer reviews, static analysis, etc.)
    - Testing for software (e.g., coverage analysis and structure-based testing, black-box functional testing techniques, integration testing, quality analysis, developing test cases based on use cases and/or user stories, testing based on operational profiles, system and acceptance testing, testing across quality attributes, regression testing, testing tools and automation, user interface testing, usability testing, performance testing, etc.)
    - Problem analysis and reporting for software (e.g., analyzing failure reports, debugging and fault isolation techniques, defect analysis, problem tracking, etc.)
2. Identify, analyze, and apply skills and professional standards in software quality.

- Software quality concepts and culture (e.g., definitions of quality, society's concern for quality, costs and impacts of bad quality, cost of quality model, quality attributes for software, roles of people, processes, methods, tools, and technology, etc.)
- Process assurance (e.g., the nature of process assurance, quality planning, process assurance techniques, etc.)
- Product assurance (e.g., the nature of product assurance, distinctions between assurance and verification and validation, quality product models, root cause analysis and defect prevention, quality product models, root cause analysis and defect prevention, quality product metrics and measurement, assessment of product quality attributes, etc.)

3. Identify, analyze, and apply skills and professional standards in security issues.
    - Developing secure software (e.g., security-related verification and validation, etc.)

Course Structure
The course is primarily about testing, versus creating quality by processes preceding testing.

What is the format of the course?
The course involves three hours of discussion/work activity a week, along with intensive after-class project activities. The goal of every class session is for individual students and teams to be able to apply requirements-related and other skills as soon as possible, with growing independence, and to learn from that application.

How are students assessed?
Labs - A series of labs in which the students learn to plan and conduct testing of software.

Project - A team of students will use the principles of Test Driven Development on a five-week software development exercise.

Presentation - A team of students will choose from one of the several domain tracks and describe Quality Assurance practices.

Exams - Two tests and a final exam.

Grade Distribution

Labs	Project	Presentation	Exams	Final Exam
20%	20%	15%	15%	30%

SCALE:
A, 93 or more; A-, at least 90 but less than 93;
B+, at least 87 but less than 90; B, at least 83 but less than 87; B-, at least 80 but less than 83;
C+, at least 77 but less than 80; C, at least 70 but less than 77;
D, at least 60 but less than 70
F, 0 to less than 60

Campus Writing Requirement
The University Writing Requirement of 2,500 words in this course will be exceeded by the lab projects and documentation for the presentation.

Inform Your Instructor of Any Accommodations Needed
Students with disabilities who require reasonable accommodations must be approved for services by providing appropriate and recent documentation to the Office of Disabled Student Services (DSS). This office is located in Craven Hall 4300, and can be contacted by phone at (760) 750-4905, or TTY (760) 750-4909. Students authorized by DSS to receive reasonable accommodations should meet with me to ensure accommodations are met.

Cougar Care Network (CCN)
Our campus has implemented a new early alert system to support and promote students' academic and personal success. I may refer you through the Cougar Care Network (CCN) to get connected to campus support and resources to assist you.

Civility Campaign
The Civility Campaign, an effort led by the Dean of Students Office, defines civility to reflect the community values of CSUSM. The university strives to be a community demonstrating respect for oneself and for others, treatment of others with dignity, and behaviors which promote a physically and psychologically safe, secure and supportive climate enabling all community members to engage as full and active participants where the free flow of ideas are encouraged and affirmed.

COURSE TOPICS for SE 461

- Software testing fundamentals (basic definitions in the field of software testing, the basic terminology and key issues, and software testing's relationship with other activities)
- Test levels in which the testing of large software is traditionally subdivided (target of the test, unit testing, integration testing, system testing)
- Objectives of testing (specifications are correctly implemented, correctness testing, performance testing, reliability testing, user interface testing, and usability testing, among many others)
- Test adequacy criteria (how much testing is enough for achieving the stated objective)
- Selection criteria (which test cases should be selected for achieving the stated objective)
- Testing for software (e.g., coverage analysis and structure-based testing, black-box functional testing techniques, integration testing, quality analysis, developing test cases based on use cases and/or user stories, testing based on operational profiles, system and acceptance testing, testing across quality attributes, regression testing, testing tools and automation, , usability testing, performance testing, etc.)
- Verification and validation terminology and foundations for software (e.g., planning the verification and validation effort, documenting verification and validation strategy, including tests and other artifacts, metrics and measurement, verification and validation involvement at different points in the life cycle, etc.)

- Test techniques (e.g., input domain-based techniques, code-based techniques, fault-based techniques, ad hoc testing, random testing, exploratory testing, …)
- Problem analysis and reporting for software (e.g., analyzing failure reports, debugging and fault isolation techniques, defect analysis, problem tracking, etc.)
- Software testing tools (Test harnesses, test generators, capture/replay tools, assertion checking, coverage analyzers and instrumenters, tracers, reliability evaluation tools)
- Process assurance (e.g., the nature of process assurance, quality planning, process assurance techniques, etc.)
- Product assurance (e.g., the nature of product assurance, distinctions between assurance and verification and validation, quality product models, root cause analysis and defect prevention, quality product models, root cause analysis and defect prevention, quality product metrics and measurement, assessment of product quality attributes, etc.)
- Developing secure software (e.g., security-related verification and validation, etc.)