Educate. Innovate. Inspire.

CS-551/CS-351 Software Testing – Graduate/Undergraduate Parent-Child Course

Professor: Zane Harvey Email: zwharvey@captechu.edu Phone: (412) 709-2184	Office Hours: Thursdays 6:30-7 PM or by appt. Dates/Times: Thursdays 7- 9:40 PM Number of Meetings: 16 All Meetings are Live online in Canvas.

Spring 2020

Course Description:

The abstract goal of software testing is to discover all errors in a software system, without exhaustively testing the software and ideally with as little testing as possible. All good programmers can debug their own code. However, this course will cover the modern approach to software testing and software quality assurance in an organizational/corporate setting. We will also investigate advancements in automated testing and testing methodologies which have been invented since the course textbooks were written. A focus will be on software testing concepts which can be applied in practice in various workplace settings. Theory will guide us in constructing test plans with adequate domain coverage.

This course covers the concepts and methodologies required for quality software testing and product deployment. The course will cover determining realistic test coverage and creating real world test plans. The course will focus on applied techniques and much of the work will be hands on testing – designing a test plan, setting up a test environment, and executing the test plan. An emphasis will be placed on contemporary tools that are available to aid in and increase the efficiency of testing. The course will also spend a reasonable time developing the theoretical toolkit necessary to plan efficient tests and investigating the theoretical foundations of software testing. Using the appropriate

theories will allow us to setup a proper test plan with the right amount of coverage. A firm grasping of the underlying theories will lay the foundation for successful and efficient real-world testing. The course will also cover other engineering principles and paradigms necessary for quality.

Prerequisites: CS-531 or exemption. Calculus, Statistics, Sequences, Basic Combinatorics, Mathematical Proof Techniques, Basic Set Theory, Basic Graph Theory

A review of the mathematical concepts necessary for this course will be provided during the lectures. The main programming languages that will be used are Java and Python.

Course Objectives and Expectations

Upon completion of the course, the student should be prepared for the ISTQB Certified Tester Foundation Level exam. The student will also have acquired practical software testing skills which will be applicable at small or large companies.

Upon Completion of the course, the student will be knowledgeable in the following topics and concepts (in no respective order):

Quality and History of Quality	Testing at Module Interfaces
Role of Testing in Quality	Verification
Software Life Cycle	Validation
Code Reviews and Walk throughs	Code Reviews
Code Review Metrics	Change Requests
Test Cases	Categories of Errors
Expected Outcomes	State oriented vs. Stateless Systems
Test Levels [Unit (static and dynamic) , Integration, System, Acceptance]	Input and Output Domains and the Concept of Complete Testing
Unit Testing Environments	Test Tools and Test Automation
Control Flow Testing, Flow Modeling, Flow and Path Analysis	Organizing a Test Team
Control Flow Graphs	Test Metrics
Statement Coverage, Branch Coverage, and Predicate Coverage Criteria	Economics of Testing and Test Automation
Domain Determination and Domain Analysis	System Integration Testing
Boundary Analysis	Agile, Waterfall, and Extreme Methods
Data Flow Anomalies	Test First Methodology
Mutation Testing	SVN, Git, and other version control tools (Ant, Maven, Gradle)
Debugging Methods	Problem Tracking Systems
V model of Software Testing	Race Conditions and Time Dependencies
Regression Testing	Characteristics of a Good Test
Glass or White Box (Structural) and Black Box (Functional) Testing	Hardware Tests

Theoretical Foundations of Software Testing	Cause Elimination
Theorems of Program Testing	Backtracking
Unit Testing	Junit and xUnit
Static Unit Testing – non-execution based testing	Unit Testing Frameworks
Dynamic Unit Testing – execution based testing	Control Flow Testing
Mutation Testing	Control Flow Graphs
Debugging – Playing Detective	Paths in a Control Flow Graph
Brute Force Testing	Path Selection Criteria
Sandwich and Big Bang Integration Testing	Generating Test Input
Software and Hardware Integration	Data Flow Testing
System Tests and Types of System Tests	Data Flow Graphs
Regression Tests	Domain testing
Documentation Tests	Domain Error
Regulatory Tests	Program Paths
Equivalence Class Partitioning	Input Domain
Boundary Value Analysis	Boundary Analysis
Decision Tables	Test Selection Criterion
Pairwise Testing	Top Down and bottom up integration testing
Random Testing and the Stochastic Generation of Test Inputs	Development Teams
Numerical Coding Testing	Testing Economics
Systematic Testing	Prototyping
Test Series	Side Effects
Test Reports	Acceptance Testing
Relationship between Tester and Developers	IV&V
Problem Reports	Integrity and Release Testing
First Customer Shipment	Certification
Acceptance Testing and Acceptance Criteria	Port Testing
Reliability and Frequency of System Failure	Error Categories
Fault, Failure, Time	Reproducing Errors
Reliability Models	Finite State Machine (FSM) Models
Test Team Organization	Transition Tours
ISO and IEEE Standards for Software Testing and Quality	Test Architectures
Maturity Models	Test Driven Development
Capability Maturity Model	Docker
Key areas of Maturity	Kubernetes
Metrics for Tracking Tests	Cloud execution
Defect Causal Analysis	Cybersecurity vulnerability assessment
Beta Testing	Secure Coding
State Transition Models	Software Maintenance Phase Planning/Execution
Test Planning	Build Managers

Time and Locations

Synchronous lessons will be online on Thursday evenings from 7- 9:40 PM.

Required Software

Java with Eclipse IDE

Python Anaconda Distribution (www.anaconda.com)

Texts:

The course content will draw from multiple sources. Assigned readings may come from any of the following resources:

- Testing Computer Software: Kaner, Falk, Nguyen (Available online)
- Introduction to Software Testing 2nd Edition by Paul Ammann (Author), Jeff Offutt (Author) ISBN-13: 978-1107172012 ISBN-10: 9781107172012
- Software Testing and Quality Assurance, Theory and Practice: Naik, Tripathy
- International Software Testing Qualifications Board – Certified Tester Foundation Level Syllabus (Available online)
- IEEE Std. 1012-1016 – IEEE Standard for System, Software, and Hardware Verification and Validation

Course Schedule

The schedule will follow Naik on a weekly basis for course reading assignments. Other readings (from Kaner, Ammann, etc.) will be assigned in class. The assigned chapters should be read before the class session.

Week	Date	Main Topic/Event	Other Notes
1	Jan 9	Chapter 1 Naik, Syllabus Review, Course Outline	Introductions, Backgrounds, Personal Testing Philosophies will be discussed
2	Jan 16	Chapter 2 Naik and Review Mathematical Concepts & Proof Techniques	
3	Jan 23	Chapter 3 Naik	
4	Jan 30	Chapter 4 Naik	
5	Feb 6	Chapter 5 Naik	
6	Feb 13	Chapter 6 Naik	
7	Feb 20	Chapter 7 Naik and Exam 1	Take Home Exam 1
8	Feb 27	Chapter 8 Naik	
9	Mar 5	Chapter 9 Naik	

10	Mar 12	Chapter 10 Naik (Spring Reading Week)	
11	Mar 19	Chapter 11 Naik and Exam 2	Proctored Exam 2
12	Mar 26	Chapter 12 Naik	
13	Apr 2	Chapter 14 Naik	
14	Apr 9	Chapter 15 Naik	
15	Apr 16	Chapters 16, 17 Naik	
16	Apr 23	Chapter 18 Naik, Final Exam Take Home	Take Home Final

Grading

Grading Components:

Projects: 50% (Various reports will be requested throughout the semester)

Exam 1: 10%

Exam 2: 10%

Final: 15%

HW: 10%

Attendance: 5%

Course Requirements

Prerequisites: CS-531 or exemption. Calculus, Basic Combinatorics, Basic Mathematical Proof Techniques (Set theory fundamentals)

Participation

Participation is required and your attendance will be tracked in Canvas.

Homework/Project Late Submissions

After the due date, assignments are eligible for 50% credit. After one week from the due date, they are not eligible for any credit.

Examination

Exam 1 and the Final will be take home exams with stipulated time windows. Exam 2 is proctored.

Communication

Emails, phone calls, text. Canvas Appointments are suggested.

Academic Integrity

Every Student is expected to be familiar with Capitol Technology University's Code of Academic Conduct including (but not limited to) the issues of cheating, plagiarism, etc. All cases of suspected academic dishonesty will be reported to the appropriate school officials, and disciplinary action may result, following investigation by a judiciary committee. Some of the core concepts are given here:

DEFINITION AND EXPECTATIONS OF ACADEMIC INTEGRITY:

Cheating – intentionally using or attempting to use unauthorized materials, information or study aids in any academic exercise. Examples include, but are not limited to, submitting another student's work as your own, using books or notes during closed book tests.

Fabrication – intentional and unauthorized falsification or invention of any information or citation in an academic exercise. Examples include, but are not limited to, changing collected data to meet the hypothesis, listing a research source that does not exist, listing a quote that does not exist.

Facilitating academic dishonesty – intentionally or knowingly helping or attempting to help another to violate any provision of this code. Examples include, but are not limited to, giving any individual other than the professor your completed assignment, suggesting ways to cheat or plagiarize.

Plagiarism – The Technology University plagiarism policy may be found online at http://www.captechu.edu/resources/lib/writingguide/plagarism.html

Self-Plagiarism – submitting the same paper or assignment for more than one class for a grade without the professor's knowledge or permission.

Complicity – failing to report the incidents of academic dishonesty to the professor, department chair, Dean of Academic Affairs, or the Vice President for Academic Affairs.

Code of Conduct – the academic integrity code is incorporated into the Capitol Technology University's Code of Conduct Standards.

Judicial Process

Any incidents should be reported to the appropriate Department Chair with written documentation. The Department Chair will forward academic integrity cases to the Academic Affairs Council for review and all other incidents to the Dean of Students.

Once the case is reviewed, the Judicial Facilitator, Dean of Students or designee, will meet with the student to discuss the allegations. The student will have the opportunity to accept responsibility and sanctions or to have the case heard by a Conduct Review Panel (CRP). If a CRP is needed, the student and all other faculty, staff or students who have direct knowledge of the incident will be asked to participate in a hearing. The CRP is composed of three members who are selected by the Judicial Facilitator from a pool of faculty, staff, or students. In cases of potential violations of the Academic Integrity Code, the CRP is generally composed of faculty members. The CRP will determine if it is more likely than not that the campus policies have been violated. If the CRP finds that the policies have been violated, they will recommend sanctions. The Judicial Facilitator will notify the student in writing of the CRP's findings. The student has the opportunity to appeal to the VP for Academic Affairs.

To learn more about the official policies of the university on this issue, please read "Code of Academic Integrity" beginning on page 18 and "Sanctions for Violations of Regulations" beginning on page 63 of the Student Handbook. The Student Handbook can be downloaded from:
http://www.captechu.edu/current-students/undergraduate/academic-resources

The contents of this syllabus or the scheduled contained herein can be modified at any time without notice.by the Professor.