QUINSIGAMOND COMMUNITY COLLEGE
(This syllabus is subject to change)
If you sign up late for class, you are still subject to late homework/exam policy.
February 3 is the last day to drop the class

COURSE DESCRIPTION:

CSC 212 Intro. Software Engineering 4 credits

This is the fifth course of a five-course program that gives students a foundation in computer science. The progression of software engineering topics from the previous two courses concludes in CSC 212, where students are asked to step beyond the programmer role and take a broader view of software development; to consider its lifecycle from problem description to maintenance. Students first practice with analysis and design of medium-sized systems. Standard modeling tools are introduced and the students complete the phases of analysis, design, implementation, and testing of a medium-sized team project that includes documents such as UML diagrams or CRC cards in addition to test plans. Students consider design patterns and write applications using data structures and templates. The software engineering topics are integrated with professionalism and ethics, as well as software and information assurance topics, such as security concerns and liabilities of computer-based systems. This course meets for 4 hours of lecture weekly.
Prerequisite: CSC 109. F/S

Office hours:
Tuesday, Thursday: 2 pm – 3 pm Office S136
Monday, Wednesday: 12 pm – 1 pm
via the Zoom link below
https://zoom.us/j/99669938161?pwd=Y2pwRWt2TUExdWhhcUh6c3o2SzV2Zz09

Instructor: Hao Loi
Associate Professor
Computer Science Dept.
Quinsigamond Community College
670 West Boylston Street
Worcester, MA 01606
Office: Suprenant #136, Mailbox #431
hloi@qcc.mass.edu
(508) 854-2702

- **You are required to participate at UMASS Undergraduate Research conference on 4/18/2025, Friday, and present your project. Failure to do so will result in a failing grade for this class.**

- **If you fail to submit an approved abstract by the deadline, you will very likely fail this class also.**

- This course aims to guide students through a semester-long research project, culminating in a presentation at the UMASS Undergraduate Conference. Students will develop skills in research, implementation, documentation, and presentation, while also learning to articulate their individual contributions to the team effort.

- Students are required to work in teams of two to complete a semester-long research project. Teamwork is a crucial component, as it allows students to share responsibilities, collaborate on tasks, and leverage each other's strengths.

Mandatory Attendance:

- **Attendance Requirement:** Attendance for this class is mandatory, not optional, and plays a critical role in your academic success and effective teamwork. Please note that accumulating four absences will result in a failing grade. This policy is designed to promote consistent participation and engagement throughout the semester.
- **Punctuality Matters:** Being on time is just as important. Arriving promptly demonstrates your commitment and respect for the learning environment, allowing you to fully benefit from each session without causing disruptions. Late arrivals can be distracting and may result in missing important material presented at the start of class. If you are more than 15 minutes late, it will be recorded as an absence.

Attendance Policy:

- Each unexcused absence will result in a 10% reduction to your final grade. This policy is designed to discourage unnecessary absences and encourage consistent attendance. I strongly encourage you to prioritize attending class and managing your schedule effectively to avoid penalties. Please be reminded that accumulating four absences will result in a failing grade.

Individual Progress Reports:

- Additionally, it is mandatory for each student to submit individual progress reports. These reports are a critical component of your assessment, allowing you to reflect on your learning and progress in the course. It is also an opportunity for you to receive personalized feedback to improve your performance.

Term Project Requirements:

For the term project, it is essential that each student clearly outlines their contribution in the following areas:

Implementation: Detail your specific roles and responsibilities in the development and execution of the project.
Documentation: Describe your involvement in creating and maintaining comprehensive documentation throughout the project.
Final Research Paper: Highlight your individual contributions to the research, analysis, and writing of the final paper.

The clarity and depth of your involvement in these areas are crucial for your assessment and the overall success of the project.

A. Course Objectives

Student Learning Outcomes (SLOs):
Upon successful completion of this course, the student will be able to:

1. Ethical and Professional Behavior in Software Development
   - Understand and apply ethical principles and professional standards in software engineering.
   - Recognize and respond to professional responsibilities and liabilities in software development.

2. Analysis, Design, and Implementation of Secure
   - Apply standard analysis and design methodologies in the creation of a team-developed software project.
   - Develop and implement a medium-sized, secure software application, incorporating formal testing procedures.

3. Algorithm Analysis
- Evaluate and differentiate various searching and sorting algorithms.
- Analyze and compare the time and space efficiencies of different algorithms.

4. Recursive Programming Solutions
- Design and build software solutions employing various recursive techniques.
- Demonstrate the ability to solve complex problems using recursion.

5. Reusable Software Development
- Design and develop reusable software components utilizing data structures and templates.
- Emphasize modularity and reusability in software design.

6. Creating Effective and Secure Software
- Apply principles of software engineering to create efficient, effective, and secure software.
- Integrate software assurance best practices to ensure software reliability and security.

7. Project Management
- Plan, organize, and manage software development projects effectively.
- Utilize project management tools and methodologies to guide the software development lifecycle.

8. Deployment and Monitoring
- Execute software deployment strategies and monitor software performance post-deployment.
- Implement continuous monitoring practices to ensure software stability and performance.

9. Performance Optimization and Security
- Optimize software for maximum performance and efficiency.
- Incorporate advanced security measures to safeguard software against vulnerabilities.

B. Course Academic Requirements and Teaching Approach

Teaching Approach: Project-Based Learning (PBL)

1. Course Introduction and Framework

Orientation: Introduce students to the course structure, emphasizing the project-based learning model. Explain how this approach ties into the outlined learning outcomes.
Expectation Setting: Clearly define expectations for participation, collaboration, and project completion.

2. Team Formation and Role Assignment
Team Building: Organize students into diverse teams, considering a mix of skills and experience.
Role Assignment: Assign or allow teams to choose specific roles (e.g., project manager, lead developer, quality assurance) to mimic real-world software development environments.

3. Project Selection and Proposal

Idea Generation: Encourage teams to brainstorm project ideas, possibly linked to current research or industry trends.
Proposal Submission: Have teams submit project proposals, including objectives, timelines, and methodologies.

4. Phased Project Development

Milestone Planning: Break the project into phases (analysis, design, implementation, testing, deployment).
Regular Check-ins: Schedule periodic check-ins or 'sprints' to monitor progress, provide feedback, and adjust project scope as necessary.

5. Peer Learning and Collaboration

Cross-Team Reviews: Arrange sessions where teams present their progress to each other, fostering peer learning and constructive feedback.
Collaboration Tools: Teach and encourage the use of collaboration tools like version control systems, project management software, and code sharing platforms.

6. Continuous Assessment and Reflection

Progress Tracking: Use rubrics to assess team and individual progress in each phase.
Reflective Journals: Encourage students to maintain journals documenting their challenges, learnings, and reflections throughout the project.

7. Final Presentation and Evaluation

Project Demonstration: Conclude with a final presentation where teams demonstrate their completed project and discuss learnings.
Peer and Instructor Evaluation: Include both peer assessment and instructor evaluation, focusing on the project outcome, teamwork, problem-solving, and application of course concepts.

9. Feedback and Iteration

Feedback Session: Post-project, conduct a feedback session to discuss what went well and what could be improved.
Iterative Learning: Encourage students to view the project as an iterative process, emphasizing continuous learning and improvement.

C. Required Textbooks:

Not required

D. Course Outline

Week	Topics
01-Feb SAT	Introduction
08-Feb SAT	Project Proposal
15-Feb SAT	Abstract Due
22-Feb SAT	Project Update 1
01-Mar SAT	Project Update 2
08-Mar SAT	Project Update 3
15-Mar SAT	Project Update 4
22-Mar SAT	Project Update 5
29-Mar SAT	Spring break
05-Apr SAT	Poster Prep.
12-Apr SAT	Final Porster
18-Apr FRI	URC Presentation

26-Apr SAT	Project Update 6
03-May SAT	Final Implementation
10-May SAT	Final Report Due

E. Course Evaluation Rubrics (ACM Cap Space Course Details)

Assessment Rubric:

Course Learning Outcome	Below Expectations	Meets Expectation	Exceeds Expectation
Compare and contrast a range of searching and sorting algorithms and analyze time and space efficiencies.	Uses various searching and sorting algorithms, and investigates time and space tradeoffs.	Compares and contrasts a range of searching and sorting algorithms for time and space efficiencies.	Critiques searching and sorting algorithms, including recursive solutions, for various algorithmic efficiencies and evaluates them in terms of Big-O.
Create effective, efficient and secure software, reflecting standard principles of software engineering and software assurance.	Calculates the risks and liabilities of a computer-based solution using standard software assurance and engineering principles.	Creates an effective, efficient and secure solution, utilizing principles of software assurance and software engineering.	Judges the safety and security of a software solution.
Design and construct programming solutions using a variety of recursive techniques.	Converts a simple recursive algorithm into a working recursive method.	With guidance develops recursive programming solutions for applications that use data structures such as trees and lists.	Independently designs and develops recursive programming solutions for applications that use backtracking and data structures such as trees and lists.
Design and develop reusable software using appropriate data structures and templates.	Differentiates among the classic data structures and selects a suitable data structure for use in an application.	With some guidance designs and develops applications using appropriate data structures for a given problem.	Independently designs and develops applications using appropriate data structures and incorporates reusable software components in the solution.

Practice the tenets of ethics and professional behavior promoted by computing societies; accept the professional responsibilities and liabilities associated with software development.	Studies the tenets of ethics and professional behavior promoted by international computing societies, such as ACM and IEEE-CS.	Practices the tenets of ethics and professional behavior promoted by international computing societies, and recognizes the liabilities associated with software development.	Displays ethical and professional behavior associated with the responsibilities of software development.
Use standard analysis and design techniques to produce a team- developed, medium-sized, secure software application that is fully implemented and formally tested.	As part of a team, produces an executable, medium-sized software application that meets some program requirements and includes design documentation and some evidence of testing.	As part of a team, produces a working, medium-sized software application on time that meets many program requirements including design and some test plan documentation.	As part of a team, successfully develops a medium-sized, secure software application on time that meets all program requirements including design and formal test plan documentation.

F. Method of Evaluation

Final project grades for this course will be based on the following:

Progress report	20%
Documentation	20%
Research paper	20%
Presentation	20%
Project	20%
Total:	100%

You must present your project at the UMASS undergraduate research conference to pass this course.

Grades:

Academic	Grade	Quality	Points
A	95-100	Outstanding	4
A-	90–94		3.7
B+	87–89	High Quality	3.3
B	83–86		3
B-	80–82		2.7
C+	77–79	Average	2.3
C	73–76		2
C-	70–72		1.7
D+	67–69	Below Average	1.3
D	63–66		1
D-	60–62		0.7
F	Below 60	Failed	0

Plagiarism Policy for Research-Based Learning Course

Overview

This policy outlines the expectations and consequences regarding plagiarism in our research-based learning course. Plagiarism, the act of presenting someone else's work or ideas as your own without proper acknowledgment, undermines the integrity of the educational process and the value of the course.

Definitions

Direct Plagiarism: Copying text, ideas, or data directly from a source without citation.
Paraphrasing Plagiarism: Rephrasing someone else's work without citing the original source.
Mosaic Plagiarism: Blending copied material from multiple sources into new work without proper acknowledgment.

Self-Plagiarism: Submitting your previous work as if it were new and original for this course.

Accidental Plagiarism: Unintentionally failing to cite sources or inadequately paraphrasing the source material.

Expectations

All submitted work must be your original creation specifically for this course.
Proper citations and references must be used when using someone else’s ideas, data, or words.
Collaboration on assignments must be within the guidelines provided by the instructor.
Procedures

Consequences

In response to these dishonest actions, this program takes a firm stance by implementing retroactive penalties. This means that once cheating is discovered, not only the direct participants but also the collaborators are held accountable and face the punishment. Consequently, regardless of the extent of involvement, all individuals who engage in plagiarism risk failing the class or facing severe academic penalties, such as suspension or expulsion. By imposing retroactive punishment, this program aims to deter plagiarism at its roots and maintain the integrity and credibility of the education system, ensuring that students are evaluated fairly based on their own merits and efforts.

Absenteeism

Any student absences from the class for four consecutive weeks or 33% of attendance during the semester will receive a "F" grade automatically.

Final course grade will follow the QCC grading system.
The above syllabus is designed in according with ACM and IEEE guidelines for Associate-Degree Transfer Curriculum in Computer Science. Some portions of this syllabus materials are extracted from http://www.acmccecc.org/course_inventory/coursedetail.aspx?cID=43