CAS CS411 Software Engineering

Syllabus (Sydney/Australia)

Instructor Name: Dr Mahendra Samarawickrama
Contact Information: msamara@bu.edu
Contact Hours: 40
Course Credits: 4 BU credits

Course Description

This course introduces students to the core principles and methodologies of software engineering, focusing on the design, development, and management of complex software systems. Students will learn about software development life cycles, project management in software contexts, requirements analysis, design principles, coding best practices, testing, and maintenance. By integrating theory with practical applications, the course prepares students to tackle real-world software engineering challenges and equips them with skills essential for success in the tech industry. Emphasis is placed on collaboration, problem-solving, and the ethical implications of software engineering.

Course Learning Outcomes

Upon successfully completing this subject, students should be able to demonstrate achievement of the following learning.

1	**Understand and apply** software engineering methodologies and life cycle models.
2	**Perform requirements analysis** and translate requirements into functional software designs.
3	**Develop software solutions** using industry-standard practices in coding, debugging, and testing.
4	**Implement project management skills** to plan, track, and deliver software projects on time.
5	**Evaluate and maintain** software systems focusing on scalability, reliability, and performance.
6	**Examine the ethical considerations** in software engineering and their impact on society and the industry.

Hub Learning Outcomes

BU Hub Units: This course fulfills a single unit in each of Quantitative Reasoning II. and Critical

Thinking.

Quantitative Reasoning II:

- **Complex Problem-Solving**: Students will frame and address complex problems using quantitative tools such as analytical, statistical, or computational methods. This course, a core part of computer science, provides an in-depth examination of problem-solving techniques following foundational programming and theory courses.
- **Application of Quantitative Tools**: Students will apply quantitative tools across various settings to answer discipline-specific questions and participate in broader societal debates.
- **Evidence-Based Argumentation**: Students will formulate and test arguments by organising and analyzing quantitative evidence. They will gather empirical data to explore and validate algorithmic properties.
- **Quantitative Communication**: Students will effectively communicate quantitative information symbolically, visually, numerically, or verbally, with numerous opportunities for symbolic communication throughout the course.
- **Understanding Quantitative Limitations**: Students will recognise and articulate the strengths and limitations of quantitative methods and understand the risks of misuse, with topics including the inherent challenges of NP-Hard problems.

Critical Thinking:

- Developing Critical Thinking Skills: Students will develop essential critical thinking skills, including identifying components of critical thinking appropriate to their discipline. Skills include differentiating between deductive and inductive reasoning, resolving disputes, identifying logical fallacies and cognitive biases, translating everyday language into formal arguments, distinguishing between empirical and normative statements, and understanding how cultural factors influence reasoning. This course, central to the CS major, integrates these elements to foster a rigorous approach to software engineering, utilising proof techniques and empirical analysis to avoid biases and logically test assumptions.

- Evaluating and Generating Arguments: Leveraging skills acquired in class, students will critically evaluate, analyse, and construct arguments, supporting evidence, and claims, including their own. The course emphasises various software engineering techniques, focusing on privacy and security, enabling students to analyse, evaluate, and critique software programs and applications effectively.

Program Learning Outcomes

Study Abroad Sydney Program Outcome:

The student will “demonstrate knowledge of Australian culture and society with respect to a combination of the following areas: Australian politics, industry, science and technology, economics, social policy, environmental policy, literature and the arts, film, marketing, advertising, and mass media”.

Study Abroad Sydney Program Outcome

The student will "demonstrate knowledge of Australian culture and society with respect to a combination of the following areas: Australian politics, industry, science and technology, economics, social policy, environmental policy, literature and the arts, film, marketing, advertising, and mass media".

Programming Environment

We will use Python and Jupyter Notebooks to demonstrate and test the algorithms. Students can utilise Google Colab for their work.

Books and Other Course Materials

Preferred Software Engineering Textbook:

- ***Braude, E., & Bernstein, M. E.*** *Software Engineering: Modern Approaches (2nd ed.). Waveland Press, Inc. ISBN: 9781478632306. Available for purchase at Barnes & Noble, Boston University.*

Other Recommended Software Engineering Textbooks:

- ***Martin, R. C.*** *Agile Software Development, Principles, Patterns, and Practices.*
- ***Bruegge, B., & Dutoit, A. H.*** *Object-Oriented Software Engineering: Using UML, Patterns, and Java.*
- ***Pfleeger, S. L., & Atlee, J. M.*** *Software Engineering: Theory and Practice.*
- ***Pressman, R. S.*** *Software Engineering: A Practitioner's Approach.*
- ***Van Vliet, H.*** *Software Engineering: Principles and Practice.*
- ***Fox, A., & Patterson, D.*** *Engineering Long-Lasting Software: An Agile Approach Using SaaS and Cloud Computing.*

Other Essential Books for Software Engineers:

- ***Brooks, F. P., Jr.*** *The Mythical Man Month.*
- ***Freeman, E. [Elisabeth], Freeman, E. [Eric], Bates, B., & Sierra, K.*** *Head First Design Patterns.*
- ***Fowler, M., Beck, K., & Roberts, D.*** *Refactoring: Improving the Design of Existing Code.*
- ***McConnell, S.*** *Code Complete: A Practical Handbook of Software Construction.*

Assignments and Grading

Final Grades

	All Grades out of 100 Points		
Grade	**Max**	**Avg**	**Min**
F	59.4	50	0.0
D	69.4	65	59.5
C-	72.4	72	69.5
C	76.4	75	72.5
C+	79.4	78	76.5
B-	82.4	82	79.5
B	86.4	85	82.5
B+	89.4	88	86.5
A-	93.4	92	89.5
A	100	96	93.5

Assessment and Marking Weighting and due date

In this course, students will participate in an online quiz, complete three assignments, take a final examination, and go on a field visit.

- **Class Participation based on topics – 5%**
- **Online Quiz – 15%**
- **Final Examination – 20%**

All submissions are due by Wednesday at 8 PM with the BU Sydney Assignment coversheet during the designated submission week. Students are welcome to submit their assignments earlier if needed.

Assessment	Weighting	Submission week	Details
Assessment 1	20%	Week 3	**Objective** In this assignment, students will develop a real-time hand gesture recognition application using a laptop webcam. The goal is to introduce students to machine learning concepts in computer vision, focusing on gesture recognition. This assignment also emphasises hands-on experience with Python and related libraries for building machine-learning models in real-time applications. **Learning Outcomes** By completing this assignment, students will be able to: • Understand the fundamentals of gesture recognition and computer vision. • Work with Python libraries for image processing and machine learning. • Develop a real-time application using Python. • Integrate webcam functionality and apply machine learning models for live data. • Understand sustainability and DEI in engineering design and development. **Task Description** Students are required to build a real-time hand gesture recognition application. The application should detect and classify specific hand gestures (e.g., thumbs up, number sign) using the laptop's webcam. It should then display a label or output indicating the detected gesture. Students should use Python for this task, leveraging various machine learning and image processing libraries. **Students must share their code, documentation, and an MP4 video demonstrating their implementation on a public GitHub repository. This GitHub repository will be used for grading and evaluation purposes.**
Assessment 2	20%	Week 5	**Objective** In this assignment, students will develop a comprehensive business proposal advocating for adopting private blockchain technology in business environments. This proposal aims to enhance students' understanding of blockchain technology, specifically the distinctions between public and private blockchains and the governance advantages that private blockchains offer. Additionally, the assignment will focus on applying private blockchain to strengthen security and privacy within the Metaverse, Web3, and NFT use cases. By completing this proposal, students will engage with recent academic research (within the last three years) to inform their recommendations. **Learning Outcomes** By completing this assignment, students will be able

			to: • Understand the key differences between public and private blockchains and their respective roles in governance. • Recognize the strengths of HyperLedger for private blockchain implementations in a commercial context. • Analyze how private blockchain technology can enhance privacy and security within the Metaverse. • Explore the adoption of private blockchain within Web3 and NFT ecosystems. • Conduct research and synthesise findings from contemporary academic papers to support a business proposal. **Task Description** Students are required to create a 3000-word business proposal advocating for the adoption of private blockchain for businesses. This proposal should include the following components: 1. **Comparison of Public and Private Blockchains**: Explain the structural and functional differences between public and private blockchains, emphasising why private blockchain is more suitable for business governance. This section should justify the preference for private blockchain, particularly in maintaining control and compliance with business regulations. 2. **HyperLedger for Commercial Private Blockchain Implementation**: Discuss the strengths of HyperLedger as a framework for implementing private blockchain solutions in business. Highlight its features, such as modularity, interoperability, and governance capabilities, and include recent case studies or academic research that showcases successful HyperLedger implementations. 3. **Privacy and Security in the Metaverse**: Examine how private blockchain can enhance privacy and security in the Metaverse by safeguarding digital identities, ensuring data privacy, and controlling access. Use recent research to support your analysis of these enhancements. 4. **Adoption of Private Blockchain in Web3 and NFT Use Cases**: Explore how private blockchain can be effectively integrated into Web3 and NFT platforms within the Metaverse. Discuss its potential for securing NFT transactions, supporting interoperability, and maintaining authenticity. Utilise current research to demonstrate how private blockchains can meet the demands of these emerging digital ecosystems.

			5. **Research and References**: Ensure that your proposal is supported by contemporary academic research within the last three years. Properly cite all sources using a consistent referencing style. **Submission Requirements**: • Students must submit their proposal as a single document in either Word or PDF format. • All references should be appropriately cited in a reference section at the end of the document. • The assignment should be uploaded to the blackboard for grading and evaluation.
Assessment 3	20%	Week 7	**Objective** To critically analyse the importance of cybersecurity in software engineering, focusing on applying the NIST Cybersecurity Framework within the software development lifecycle (SDLC) to enhance security practices. **Learning Outcomes** By completing this assignment, students will be able to: • Understand the role of cybersecurity in software engineering. • Explore the NIST Cybersecurity Framework and its relevance to secure software development. • Identify common security vulnerabilities and strategies to mitigate them. • Analyze the integration of cybersecurity measures within the SDLC. • Apply contemporary academic research to support secure software engineering practices. **Task Description** Students must write a 3000-word academic report discussing the importance of cybersecurity in software engineering. The report should: 1. Define cybersecurity's role in software engineering. 2. Provide an overview of the NIST Cybersecurity Framework and its five core functions. 3. Discuss the application of the NIST Framework within each stage of the SDLC. 4. Address common software vulnerabilities and how the NIST Framework mitigates these. 5. Identify challenges in implementing cybersecurity and the NIST Framework in software projects. 6. Incorporate recent academic research and case studies to support the analysis. **Submission**: The report should be uploaded to Blackboard in Word or PDF format.

Community of Learning: Class and University Policies

Course atmosphere, diversity and inclusion:

We intend to provide a positive and inclusive atmosphere in class and on the associated virtual platforms. Students from a wide range of backgrounds and with diverse perspectives are welcome. We ask that students treat each other with thoughtfulness and respect and do their part to make all their peers feel welcome. Your suggestions are encouraged and appreciated. Please let us know ways to improve the effectiveness of the course for you personally or other students or student groups.

Course Matters

Attendance at all classes is mandatory.
Any absence for medical reasons or other misadventure must be supported by a medical certificate or a letter offering a satisfactory explanation. Strict penalties apply, on a pro-rata basis, for any unapproved absence. Missing one class without reason would attract as much as a 10% penalty.

Statement on Plagiarism
All students are responsible for having read the Boston University statement on plagiarism, which is available in the Academic Conduct Code. Students are advised that the penalty against students on a Boston University program for cheating on examinations or for plagiarism may be "… expulsion from the program or the University or such other penalty as may be recommended by the Committee on Student Academic Conduct, subject to approval by the Dean".

Academic Conduct
Academic standards and the code of academic conduct are taken very seriously by the University, the College of Arts and Sciences, and the Department of Computer Science. Course participants must adhere to the CAS Academic Conduct Code. Please take the time to review this document if you are unfamiliar with its contents.
Any case of academic misconduct, including but not limited to plagiarism and submission of work that was not solved by you – be that a tutor, friend, web resource, AI, etc., will be reported to the College and will also carry a grading penalty.

Late Work
In general, there will be no extensions granted for any coursework. The exception is where there are clear and acceptable reasons for late submission. In this case a written statement outlining any serious illness or misadventure together with supporting documentation (e.g. medical certificates) must be provided or a strict penalty of 5% per day will apply.
Tentative schedule

This schedule is for informational purposes only and is subject and is likely to change as we progress through the semester.

Week	Topic
1	Software Development Lyfe Cycle
2	Object Oriented Programming and Design Patterns
3	Machine Learning and Neural Networks
4	Cyber security, Data Privacy and Software Engineering
4	Field Trip
5	Quantum Computing
6	Blockchain and Web3
7	Emerging tech and Software Engineering (Industry Perspective)

Appendix

GitHub: Essential for version control, collaborative coding, and hosting open-source projects.
W3Schools: Tutorials and references on web development languages.
GeeksforGeeks: Coding problems, algorithm explanations, and tutorials for various programming languages and computer science concepts.
Stack Overflow: Q&A site for software developers; very helpful for problem-solving and troubleshooting.
NIST (National Institute of Standards and Technology): Access to cybersecurity standards, frameworks (like the NIST Cybersecurity Framework), and technical resources.
LeetCode: Coding challenges and interview prep for data structures, algorithms, and problem-solving.
HackerRank: Practice coding problems and participate in coding contests.
CodeSignal: Coding practice platform with a focus on real-world software engineering challenges.
Codewars: Practice and improve coding skills through challenges and community-driven problems.
AWS Educate: Free resources and cloud computing courses for students interested in cloud-based software engineering.
Azure for Students: Free access to Microsoft Azure services and student credits.
Docker: Containerization platform that is useful for deploying and managing software in isolated environments.