AI-Driven Trustworthy Software Development
ITCS 6010/8010 – Spring 2026

Welcome to *ITCS 6010/8010 – AI-Driven Trustworthy Software Development*. This project-based graduate course explores how software engineering practices must evolve when Artificial Intelligence (particularly Large Language Models) becomes an active participant in the software development process. In this course, we will build a real software system using AI tools across the software lifecycle, including requirements engineering, design, coding, testing, deployment, and monitoring. The focus is not only on learning how to use AI tools in software development, but also on understanding how trust in software must be engineered, evaluated, and defended when development artifacts are generated by probabilistic and fallible AI systems.

This syllabus describes the policies, expectations, and structure of the course. Please read it carefully before continuing. These policies are designed to support a rigorous, evidence-driven learning environment for all students. Participation in the course assumes that you are willing to abide by these expectations. Changes to the syllabus will be communicated via class announcements and/or Canvas.

Course Description

This course provides a project-centered exploration of software development in the presence of Artificial Intelligence (AI), with a particular focus on trustworthiness and evidence-based engineering. As AI systems, especially Large Language Models (LLMs), are increasingly being used for requirements engineering, design, coding, testing, and deployment, traditional assumptions about correctness, reliability, and developer intent no longer hold. This course examines how software engineering practices must adapt when development artifacts are probabilistic, fallible, and partially autonomous.

The course guides students through the full software development lifecycle using a single evolving project. Students work in teams to design, implement, and evaluate a realistic software system in which AI tools actively generate or influence development artifacts. Rather than treating AI as a productivity aid, the course treats AI as a first-class development actor whose outputs must be questioned, constrained, and evaluated. Central topics include AI-assisted requirements elicitation, AI-influenced architectural design, AI-generated code and tests, and the risks introduced by AI-generated configurations and deployment artifacts.

A central principle is the use of a DevOps pipeline as an enforcement and evaluation mechanism. The relevant trustworthiness properties, such as reliability, security, safety, and robustness, are evaluated continuously through explicit trust gates integrated into the pipeline. Students explore how to design systems so that trust claims are observable, testable, and defensible, and to document failures through incident analysis rather than hiding or avoiding them.

The course emphasizes exploration, hands-on, and collaborative learning in an AI-driven software development setting. Lectures are deliberately short and are paired with studio-style sessions in which students use generative AI tools to build, break, and refine software systems, and to examine the trust implications of AI-generated artifacts. There are no traditional exams. Assessment is based on the continuous analysis of project artifacts and a final defense in which students must demonstrate what their system can and cannot be trusted to do.

This is a super-interesting course! It is intended for MSc and PhD students with a strong interest in modern software engineering, DevOps, and the challenges of building and evaluating trustworthy systems when generative AI is integral to the development process. Success in the course requires sustained engagement with a semester-long project, active participation in studio-style classes, a willingness to experiment and learn from failure, and careful, evidence-based reasoning rather than reliance on intuition or tool-generated explanations.

Credit hours: 3

Prerequisites

Students are expected to have strong programming skills and a solid background in computer science. Familiarity with at least one modern programming language (e.g., Java, Python, or similar), basic software engineering concepts, version control (e.g., Git), and a Unix/Linux development environment is required. Prior exposure to data structures and algorithms, testing, and basic systems concepts is assumed.

Familiarity with machine learning concepts and experience using AI-based programming tools are helpful but not required. Students must be comfortable working on open-ended projects and reading technical documentation.

Learning Outcomes

Upon successful completion of this course, students will be able to:

- Understand and explain how the use of Artificial Intelligence, particularly Large Language Models, fundamentally changes software development practices and challenges traditional assumptions about correctness, reliability, and developer intent.
- Apply AI-driven techniques across multiple phases of the software development lifecycle, including requirements engineering, design, coding, testing, deployment, and monitoring, while recognizing the risks and limitations introduced by probabilistic AI-generated artifacts.
- Design and implement a DevOps pipeline that integrates AI-assisted development tools and enforces trustworthiness properties through explicit, observable trust gates.
- Evaluate software trustworthiness using evidence-based methods, including metrics, testing, security analysis, monitoring, and incident analysis, rather than relying on intuition or tool-provided explanations.
- Identify, analyze, and document failures arising from AI-generated artifacts, including logic errors, evaluation blind spots, security vulnerabilities, and pipeline enforcement gaps, and use these failures to refine trust assumptions and system design.
- Critically assess and justify trade-offs between productivity, safety, reliability, and robustness when delegating development tasks to AI systems, and articulate and defend scoped trust claims by clearly stating what a software system can and cannot be trusted to do, the evidence supporting those claims, and the limitations that remain.

Location and Time

Tuesday, 5:30 pm-6:30 pm, Atkins 126

Tuesday, 5:30 pm-6:30 pm, Atkins 126

Instructor

Marco Vieira
Email: marco.vieira@charlotte.edu
Office: Woodward 205C
Office Hours: Tuesday and Thursday, 4:30 pm to 5:30 am

Textbook(s)

Selected readings on generative AI and LLMs in software engineering.

(optional) Pressman, Roger S., 2019, "Software Engineering: A Practitioners Approach", Ninth Edition, McGraw-Hill Higher Education, ISBN-13: 978-1260548006.

(optional) Raschka, Sebastian, 2024, "Build a Large Language Model (From Scratch)", Manning, ISBN-13: 978-1633437166.

Course Topics

Getting Started: Artificial Intelligence, particularly Large Language Models, as active participants in software development, and the implications of probabilistic, fallible AI-generated artifacts for responsibility, correctness, and engineering practice.

Trustworthiness as an Engineered Property: Foundations of software trustworthiness, including reliability, safety, security, robustness, and the distinction between trust, confidence, and correctness.

AI-Assisted Requirements and Design: Use of AI for requirements elicitation and architectural design, identification of ambiguity and inferred behavior, and design principles that enable evaluation and enforcement of trust.

DevOps Pipelines for Continuous Trust Evaluation: Continuous integration and delivery pipelines as mechanisms for enforcing trust, generating evidence, and making AI-related risks observable throughout the software lifecycle.

Code, Testing, Deployment, and Runtime: Delegation of coding and testing tasks to AI systems, limitations of testing and oracles, security and deployment risks, and runtime monitoring for detecting silent or evolving failures.

Adversarial Analysis and Evidence-Based Trust Claims: Adversarial testing, trust-gate refinement, peer auditing, incident analysis, and communication of defensible trust claims using a claim-evidence-limitation structure.

Format of (most) Classes

Most classes in this course follow a consistent structure designed to support sustained project progress, critical thinking, and hands-on experimentation with AI-driven software development.

Meaningful progress in the course requires substantial work outside of scheduled class time; in-class sessions are intended to support, not replace, independent and team-based project work.

- **Before class**: Students are expected to review any assigned readings, slides, short videos, or artifacts provided in advance. Preparation may also include reviewing the current state of the team's project, recent pipeline results, incident reports, or open questions identified in the previous session. This preparation is mandatory. In-class time assumes familiarity with the context and is not used for first exposure to material.
- **During class**: Each session begins with a brief framing by the instructor to introduce the day's focus, connect it to the overall course trajectory, and clarify expectations. This is followed by discussion, often grounded in the team's ongoing project and recent observations. Most of the class time is devoted to studio-style work. The instructor circulates to ask probing questions, challenge assumptions, and provide feedback, rather than delivering extended lectures.
- **After class**: Students are expected to continue working on their projects outside of class. This includes implementing changes, running experiments, committing updates to the repository, refining documentation, incident reports, or trust artifacts, and recording decisions or open issues identified during the session. These activities are core to the course and essential for maintaining continuous progress and evaluation rather than last-minute work.

Not all classes follow this format exactly (e.g., audits or final presentations), but this structure represents the default rhythm of the course.

Grading

Students will be evaluated through a combination of continuous project-based work and a final project synthesis and defense. There are no traditional quizzes, assignments, or exams. Evaluation emphasizes sustained engagement, evidence-based reasoning, and responsible use of AI throughout the software development lifecycle.

System Proposal and Infrastructure Setup (5 points): Early in the semester, each team will propose a software system and establish the initial project infrastructure. This component evaluates whether the chosen system is appropriate for AI-driven trust evaluation, whether anticipated AI-related trust risks are clearly identified, and whether a shared repository and minimal CI/CD pipeline are set up in a timely manner. The emphasis is on readiness and evaluability rather than ambition or feature completeness.

Requirements and Trust Goals (10 points): Teams will use AI-assisted techniques to elicit, refine, and reason about system requirements with explicit attention to trust. This component evaluates the identification of ambiguities, unsafe assumptions, and trust-critical requirements, as well as the articulation of trust goals that distinguish between what must hold, what may fail, and what cannot be guaranteed. Credit is based on clarity and critical reasoning rather than stability or finality of requirements.

Design for Trust and Evaluation (10 points): This component focuses on evaluating architectural and design decisions made with and against AI input. Teams are assessed on their ability to

critically engage with AI-generated design suggestions, document rejected or modified proposals, and make design choices that enable trust evaluation and enforcement. The focus is on design rationale and evaluability rather than architectural elegance.

From Design to Execution (5 points): As teams move from design artifacts to a working system, this component evaluates early integration and pipeline readiness. It rewards teams for implementing a minimal yet complete execution path, integrating early trust gates, and exposing failures rather than delaying integration in pursuit of completeness.

Coding and Logic Trust (5 points): This component evaluates how teams delegate coding and refactoring tasks to AI systems and reason about the trustworthiness of AI-generated logic. Assessment focuses on identifying logic errors, misleading behavior, and mismatches between AI explanations and observable outcomes, and on treating AI-generated code as a first-class, fallible project artifact.

Testing, Oracles, and False Confidence (5 points): This evaluates testing practices in the presence of AI. Teams are assessed on their understanding of test coverage limitations, oracle problems, and the risks of AI-generated tests, as well as their ability to recognize false positives and false negatives. Strong work uses testing outcomes to refine trust assumptions rather than to claim correctness.

Deployment and Runtime Trust (5 points): This component evaluates trust after deployment, focusing on deployment decisions, AI-generated configuration risks, runtime observability, and the detection of silent or gradual failures. Credit is given for making runtime behavior observable rather than assuming deployment success implies trust.

Breaking the System (5 points): This component evaluates adversarial thinking. Teams are assessed on deliberate attempts to break their systems and pipelines through adversarial commits, prompt perturbations, configuration drift, or trust-gate bypass attempts. The goal is to expose trust weaknesses and enforcement limits, not to demonstrate robustness.

Final Project, Evidence Consolidation, and Defense (50 points): The final evaluation is based on the system's final state, consolidated documentation, and a live project defense. Teams are evaluated using the same criteria applied throughout the semester, but assessed holistically and in their final, integrated form. The emphasis is on the team’s ability to synthesize all evidence accumulated during the semester into clear, defensible trust claims. This includes an explicit description of AI usage across the software lifecycle, trust gates and evaluation results, analysis of failures and residual risks, and the ability to articulate what the system can and cannot be trusted to do using a claim-evidence-limitation structure. Honesty about limitations and uncertainty is required and rewarded.

Standard grading -

100%-90%: A
<90%-80%: B
<80%-70%: C
<70%: F

Policies

I. Course Materials

All lectures and course material will be available in *Canvas*. Lectures and course materials, including presentations, assignments, exams, outlines, and similar materials, are protected by copyright. You are encouraged to take notes and make copies of course materials for your educational use. However, you may not, nor may you knowingly allow others to reproduce or distribute lecture notes and course materials publicly without my express written consent. This includes providing materials to commercial course material suppliers. Students who publicly distribute or display or help others publicly distribute or display copies or modified copies of an instructor's course materials may violate University Policy 406, The Code of Student Responsibility, or University Policy 407, Code of Student Academic Integrity. Similarly, you own copyright in your original papers and exam essays.

II. Classroom Conduct

This course is conducted in an atmosphere of mutual respect. Active participation in discussions is greatly encouraged. Each of us may have strongly differing opinions on the various topics of class discussions. The conflict of ideas is encouraged and welcome. The orderly questioning of others' ideas, including mine, is similarly welcome. However, I will exercise my responsibility to manage the discussions so that ideas and arguments can proceed in an orderly fashion. You should expect that if your conduct during class discussions seriously disrupts the atmosphere of mutual respect I expect in this class, you will not be permitted to participate further.

III. Attendance and Absences

Students are expected to attend every class and remain in class for the duration of the session. Failure to attend class or arrive late may affect your ability to achieve course objectives, which could impact your course grade. An absence, excused or unexcused, does not relieve a student of any course requirement. Regular class attendance is a student's obligation, as is the responsibility for all the work of class meetings, including tests and written tasks.

The instructor has the authority to excuse a student's class absence(s) and to grant a student an academic accommodation (turn in a late assignment, provide extra time on an assignment, reschedule an exam, etc.). However, under the Academic Affairs Policy on Course Attendance and Participation, University-sanctioned events or activities are considered excused absences. A University-sanctioned event or activity is one in which a student formally represents the University to external constituencies in athletic or academic activities. This policy does not supersede individual program attendance and/or participation requirements that are aligned with accreditation or licensure. For more information and student responsibilities to account for such an absence, see provost.charlotte.edu/policies-procedures/academic-policies-and-procedures/course-attendance-and-participation.

IV. Instructor's Absence or Tardiness

If the instructor is late to class, students must wait a full 20 minutes after the start of class before leaving without being counted absent, or they must follow any written instructions I may give them regarding my anticipated tardiness.

V. Non-Discrimination

All students and the instructor are expected to engage with each other respectfully. Unwelcome conduct directed toward another person based upon that person's actual or perceived race; color; religion (including belief and non-belief); sex; sexual orientation; gender identity; age; national origin; physical or mental disability; veteran status; genetic information; or for any other reason, may constitute a violation of University Policy 501, Nondiscrimination. Any student suspected of engaging in such conduct will be referred to the Office of Civil Rights & Title IX.

VI. University Policy on Withdrawals

Students are expected to complete all courses for which they are registered at the close of the add/drop period. If you are concerned about your ability to succeed in this course, it is important to make an appointment to speak with me as soon as possible. The University policy on withdrawal allows students only a limited number of opportunities available to withdraw from courses. It is important for you to understand the financial and academic consequences that may result from course withdrawal. See: provost.charlotte.edu/policies-procedures/academic-policies-and-procedures/withdrawal-and-cancellation-enrollment-policy

VII. Computer, Cell Phones and Other Mobile Devices in the Classroom

Students are permitted to use computers during class for note-taking and other class-related work only. Those using computers during class for work not related to that class must leave the classroom for the remainder of the class period.

The use of cell phones, smartphones, or other mobile communication devices is disruptive and is therefore prohibited during class. Except in emergencies, those using such devices must leave the classroom for the remainder of the class period.

VIII. Use of Generative Artificial Intelligence (AI)

This course explicitly assumes and requires the use of generative artificial intelligence (AI) tools, such as ChatGPT and similar systems, in learning activities and project work. Generative AI is treated as a first-class development tool in this course rather than as an optional aid. To maintain academic integrity, students must disclose any AI-generated material they use and properly attribute it, including in-text citations, quotations, and references (see, for example, https://apastyle.apa.org/blog/how-to-cite-chatgpt). Students remain fully responsible for the content they submit, including any errors, omissions, or misrepresentations (e.g., hallucinations) produced by generative AI tools.

Students should also include the following statement in their assignments to indicate use of a generative AI tool: "*The author(s) acknowledges the use of [generative AI tool Name] in the preparation or completion of this assignment. The [generative AI tool Name] was used in the following way(s) in this assignment: [e.g., brainstorming, grammatical correction, citation, which portion of the assignment].*"

Important Note on Data Protection and Privacy: When using generative AI tools, be aware that the data you supply may be used to train AI models or for other purposes. Consequently, there is no guarantee that the information you provide will remain confidential. You should exercise

caution and avoid sharing any sensitive or private information when using these tools. Examples of such information include personally identifiable information, protected health information (PHI), financial data, intellectual property, original research, and any other data that might otherwise be legally protected.

IX. Syllabus Policies, Academic Integrity, Plagiarism

All students are required to read and abide by the Code of Student Academic Integrity. Violations of the Code of Student Academic Integrity, including plagiarism, will result in disciplinary action as provided in the Code. Definitions and examples of plagiarism are set forth in the Code and on the Student Accountability & Conflict Resolution website. The Code is available from the Dean of Students Office or online at legal.charlotte.edu/policies/up-407. Additional resources are available on the Student Accountability & Conflict Resolution website.

Violation of these syllabus policies may result in appropriate academic penalties, including a reduction of grade in the relevant assignment, project, or exam. If violation of these syllabus policies also implicates the Code of Student Academic Integrity because of alleged academic misconduct, I will follow the process outlined in the Code to address such cases.

X. Reporting Expectations

UNC Charlotte is committed to maintaining an environment conducive to learning for all students and a professional workplace for all employees. The University takes active measures to create or restore a respectful, safe, and inclusive environment for community members that is free from discrimination, discriminatory harassment, and interpersonal violence. If you (or someone you know) has experienced any of these incidents, know that you are not alone. UNC Charlotte has staff members trained to support you in navigating campus life, accessing health and counseling services, providing academic and housing accommodations, helping with civil protective orders, and more.

Please be aware that all UNC Charlotte employees, including faculty members, are expected to report any information or incidents of discrimination, discriminatory harassment, or sexual and interpersonal misconduct to the Office of Civil Rights and Title IX. This means that if you tell me about a situation involving these matters, I am expected to report the information. Although I am expected to report the problem, you will still have options regarding how your case will be handled, including whether you wish to pursue a formal complaint. Our goal is to make sure you are aware of the range of options available to you and have access to the resources you need.

If you wish to speak to someone confidentially, you can contact the following on-campus resources, who are not required to report the incident to the Office of Civil Rights and Title IX: (1) Center for Counseling and Psychological Services (CAPS) (caps.charlotte.edu, 7-0311); or (2) Student Health Center (studenthealth.charlotte.edu, 7-7400). Additional information about your options is also available at civilrights.charlotte.edu under the “Students” tab.