CSCE 3444.002 – Software Engineering

Fall 2026 – 3 Credit Hours

University of North Texas – Department of Computer Science and Engineering

Course Information

Class Timing:	**Monday and Wednesday 12:15 pm – 1:35 pm**
Location:	**NTDP E265**

Instructor:	Beilei Jiang
Office Hours:	Monday 2:00pm – 3:00pm, Tuesday 12:30pm – 3:30pm
Email:	Beilei.Jiang@unt.edu
Location:	NTDP E245J

TA:	Paulson Elsa
Email:	Elsa.Paulson@my.unt.edu
TA Office Hours:	Monday & Tuesday 2:00pm - 3:00pm
Zoom:	Else's Zoom Meeting

Course Documents:	All lecture notes, assignments, and announcements will be available on Canvas
Class Textbook:	Software Engineering: A Practitioner's Approach 9th Edition By Roger Pressman and Bruce Maxim. (supplementary)

Course Description

This course introduces students to the principles and practices of software engineering through a semester-long team project. Students move from user-centered project ideation to project planning, requirements engineering, software design, implementation, quality evaluation, testing, and final delivery. The course emphasizes software development processes, SRS and use cases, UML and design, collaborative Git/GitHub workflow, usability and accessibility, software testing, documentation, and professional project presentations.

Why This Course Matters

Software engineering is more than writing code. It is the organized process of understanding a real user problem, planning the work, defining what the system must do, designing and building the solution, and evaluating whether the result is reliable, usable, accessible, and maintainable. The semester project connects these activities into one complete development experience.

Learning Objectives

By the end of this course, students will be able to:

1. Explain the software development life cycle (SDLC) and compare major process models, selecting an appropriate approach based on project conditions and tradeoffs.
2. Identify users and stakeholders, elicit and refine functional and non-functional requirements, and create a software requirements specification (SRS) with use cases.
3. Apply project planning practices to define scope, team roles, communication, tools, milestones, schedules, and major project risks.
4. Create software design artifacts that translate requirements into an implementable solution using appropriate architecture, UML, interface, workflow, and data design.
5. Use team workflow and configuration-management practices, including Git/GitHub, branches, pull requests, issue tracking, and project boards.
6. Implement and integrate functional software features and apply usability, accessibility, code review, debugging, and iterative refinement to improve software quality.
7. Design and execute appropriate software tests, maintain project documentation, and communicate project progress and results through reports, presentations, and live demonstrations.

Prerequisite

CSCE 2110 with a grade of C or better.

Grading Criteria

Team Project & Presentation	**35%**	Semester-long group project with midterm and final presentations
Group Assignments	**30%**	Collaborative tasks and project deliverables
Individual Assignments	**15%**	Individual work applying SDLC, requirements, design, testing, and related course concepts
Peer Evaluations	**15%**	Teammate feedback on contributions
Attendance	**5%**	Regular attendance and active participation

Grade Assignments are made based on the following:

A = 90-100%
B = 80-89.9%
C = 70-79.9%
D = 60-69.9%
F = less than 60%

Course Schedule: (subject to change; updates will be posted on Canvas)

Week	Class Topic	Expected Team Work	Deliverables
1	Intro to Software Engineering & Generic Process	Form teams; review project expectations; connect user problems to the software engineering process.	**Team formation**
2	User-Centered Project Ideation	Define target user/problem; brainstorm and refine a meaningful interactive project idea beyond basic CRUD.	**Project Proposal**
3	Team Workflow, GitHub & Configuration Management	Set up and organize the team GitHub repository; use branches, commits, pull requests, issues/project boards, and define a clear team workflow for integration and collaboration.	**GitHub workflow checkpoint**
4	SDLC & Process Models / Mon: Labor Day - No Class	Wed lecture: review SDLC phases and compare major process models, including Waterfall, V-Model, Lean, Spiral, Iterative, Incremental, Agile, and DevOps.	
5	Process Model Selection & Project Planning	Practice choosing a process model from project clues and tradeoffs; define project scope, roles, communication, tools, milestones, schedule, and risks.	**Individual Assignment 1 / Team Project Plan**
6	Requirements Engineering & SRS	Elicit, analyze, prioritize, and validate stakeholder needs; write clear functional and non-functional requirements; develop use cases and an SRS draft.	**SRS Document**
7	Software Design & UML	Translate requirements into a software blueprint using architecture, UI/workflow design, UML diagrams, and relevant data/API design.	**Design Document**
8	Implementation I	Implement meaningful functional features; integrate work; review code and resolve conflicts.	**Project Status 1**
9	Implementation II & Prototype Integration	Continue implementation, debugging, integration, and midterm prototype preparation.	**Midterm Demo**
10	Software Quality: Usability	introduce NFRs and usability evaluation.	**NFR + LLM Evaluation Assignment**
11	Accessibility	Evaluate and improve accessibility, usability, feedback, navigation, and error handling.	**Accessibility testing**
12	Software Testing	Create/run unit, integration, system, regression, usability, and other appropriate tests.	**Peer Evaluation 2, Individual Assignment 2**
13	Stabilization & Final Presentation Prep	Complete required features, stabilize software, finish slides, and practice the final demo.	**Project Status 2** **Final Project presentation slides + demo**
14	Final Project Presentations I	Approximately 20-25 minute team presentations with live software demonstrations.	**Final presentation**
Break	Thanksgiving Break - No Class		
15	Final Project Presentations II & Wrap-Up	Remaining presentations; submit final project materials and code.	**Final code & slides**

Project Deliverables:

1. **Project Plan** - A practical team plan covering the project overview and scope, roles, communication, development environment/tools, milestones, schedule, major risks, and the GitHub repository link.
2. **Software Requirements Specification (SRS)** - A structured document describing WHAT the software must do, including stakeholders, system overview, functional and non-functional requirements, use case diagram/descriptions, and individual contributions.
3. **Design Document** - A technical blueprint describing HOW the system will be organized, using appropriate architecture, UML/design models, interface/workflow decisions, and data/API/database design when relevant.
4. **Test Plan** - A testing document with specific test cases for important functionality and quality requirements. Include expected results, actual results, identified defects, and appropriate testing methods.
5. **Final Presentation & Demo** - An approximately 20-25 minute team presentation including a live demonstration of the working software. Submit slides and final code, explain major technical decisions and testing results, and be ready for questions. All team members must participate.
6. **Project Status Reports** - Team updates submitted at designated checkpoints describing completed work, current work, blockers, project status, and each member's substantive contributions.
7. **Peer Evaluations** - Confidential evaluations submitted at designated checkpoints to assess each teammate's contribution, effort, communication, and collaboration. Project grades may be adjusted based on peer feedback.

Attendance:

- Attendance is required and will be taken periodically. [check attendance 7 times, and drop the lowest two scores.]
- Notify me or the TA in advance if you must miss class.
- Participation in group work and presentations requires your presence.

AI Usage Policy:

AI tools (GitHub Copilot, Claude, ChatGPT) are encouraged as development aids and must be cited using code comments. You must understand and be able to explain all submitted code. Include AI usage in your status reports. AI cannot be used for peer evaluations or reflection writing. The goal is to become an AI-augmented engineer, not AI-dependent. Uncited AI use or submitting AI-generated code without understanding constitutes academic dishonesty.

Project Guidelines:

THE PROJECT CAN:

- Be done in a language of your choosing (approval required if other than C/C++, C#, Python, Java, JavaScript)
- Be web-based or mobile app (you supply your own hardware)

THE PROJECT MUST:

- Have appropriate scope for 2-3 team members over the term
- Have a Graphical User Interface (GUI)
- Have a meaningful interactive core beyond basic create/read/update/delete (CRUD) operations
- Have at least one customer/user group you can gather requirements from

THE PROJECT MUST NOT:

- Be trivial or primarily a simple CRUD application (e.g., a basic calculator or to-do list)
- Be static web pages or screens
- Be a game (Serious/educational games are OK with approval)
- Already exist (no cloning existing repositories)
- Harm the safety, security, or privacy of users
- Be primarily used by minors

Communication Expectations:

- Email is the primary way to contact me or the TA.
- Expect a reply within 24 hours on weekdays.
- Keep messages short and clear. No need for overly formal AI-generated text.
- For grade questions, contact the TA first; follow up with me if needed.

Late Submission:

- Assignments are due on the posted deadline.
- Late work submitted within 24 hours after the deadline will receive a 50% penalty. Work submitted more than 24 hours late will not be accepted.
- Technical issues (e.g., internet outage, computer crash) are not valid excuses; plan ahead.

Project Grade Scaling:

- Group project grades will be scaled based on peer evaluations
- Speak up and volunteer – "my team didn't give me work" is not an excuse
- Every member must make substantive implementation contributions to the working software
- Setup tasks, documentation-only work, or very small UI changes do not replace substantive functional contributions
- Contact the instructor or TA early if team issues arise

Academic Dishonesty/Plagiarism:

Students must follow the UNT Policy on Student Academic Integrity (Policy 06.003) and the CSE Department Cheating Policy. Academic dishonesty — including cheating, plagiarism, unauthorized collaboration, or using prohibited materials — will result in a failing grade for the course and a report to the Office of Academic Integrity.

Student Perceptions of Teaching (SPOT)

SPOT is your opportunity to provide feedback on this course and instructor. It will be available near the end of the semester. Your responses help improve teaching and learning at UNT.

Syllabus Revisions

This syllabus may be modified as the course progresses should the instructor deem it necessary. Notice of changes to the syllabus shall be made through Canvas and/or in-class announcements.

Disability Accommodation

The University of North Texas complies with Section 504 of the 1973 Rehabilitation Act and with the Americans with Disabilities Act of 1990. The University of North Texas provides academic adjustments and auxiliary aids to individuals with disabilities, as defined under the law. Among other things, this legislation requires that all students with disabilities be guaranteed a learning environment that provides for reasonable accommodation of their disabilities. If you believe you have a disability requiring accommodation, please see the instructor and/or contact the Office of Disability Accommodation at 940-565-4323 during the first week of class.