CIS*2750: Software Systems Development and Integration

Winter 2026

School of Computer Science
College of Computational, Mathematical, and Physical Sciences
University of Guelph

Lecture Section	01
Instructor	Dr. D. Nikitenko (cis2750@socs.uoguelph.ca)
Class Time / Location	Tue/Thu 4:00–5:20 pm | MACN 105
Office Hours	by appointment
Lecture Section	02
Instructor	Dr. J. McCuaig (cis2750@socs.uoguelph.ca)
Class Time / Location	Tue/Thu 8:30–9:50 am | MACN 105
Office Hours	by appointment
Course Site	https://moodle.socs.uoguelph.ca
Credit Weight	0.50
Delivery Method	In Person

Calendar Description

This course introduces techniques and tools used in the development of large interactive software systems. Students learn methods for constructing modules in different programming languages, developing software to specifications, organizing and constructing modular systems, and working with permanent data storage. Students also explore introductory database management tools.

The Academic Calendars are the source of information about the University of Guelph's procedures, policies and regulations which apply to undergraduate, graduate and diploma programs.

Requisites and Restrictions

Prerequisites: CIS*2500, CIS*2520, CIS*2430 or ENGG*1420
Co-requisites: None
Restrictions: None

Course Details

This course emphasizes the development of modular, testable, and maintainable software systems. Students will learn to design to specification, create clean module interfaces, and integrate components across programming languages using C and Python. Emphasis is placed on disciplined engineering practices, including data storage design, automated testing, version control, and deployment.

The course adopts a project-driven approach: students progressively build a larger system through a sequence of assignments. Each assignment requires both correct implementation and demonstrated understanding of design and integration principles. Continuous integration (CI) tools are used to automatically test submissions, encouraging iterative development and feedback-driven improvement. Working software is required early in the semester and must be maintained throughout, reflecting real-world practices of keeping systems functional and accessible.

Learning Outcomes

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

1. Design and develop modular software components that adhere to specifications and support integration into larger systems.
2. Integrate software components, including those written in different programming languages, into cohesive software systems.
3. Implement data storage solutions to support software functionality.
4. Apply quality assurance and systematic testing techniques using modern testing frameworks.
5. Leverage contemporary tools and practices to streamline software development, integration, and deployment processes.

Required and Recommended Textbooks

Required: None. cost: $0.
Recommended: Readings and documentation posted on the course site [1].

Schedule of Topics- subject to change

Lecture slots will be used to introduce and explain course concepts. Lab sessions will function as tutorials, providing guided problem solving, worked examples, and opportunities for discussion and collaboration. Lectures and labs will not be recorded. Presentation materials will be shared, but they are not a substitute for attending class. In-class discussions, slide annotations, and clarifications will not appear in the posted materials. Regular attendance is important to your success in this course.

Wk	Topics / Activities	Coursework
1	Course intro and overview of topics; unit testing in C with Check; testing principles; assertions; Docker/autograder lab.	A0 assigned (released Jan 6)
2	Compilation pipeline; Makefiles; version control; Git/Makefiles lab.	A0 due (Jan 16); A1 assigned
3	Debugging (valgrind; compile vs run-time; platform dependence); linters; APIs and interface contracts.	
4	Defensive programming; testing your own code; error handling; shared libraries and dynamic linking; debugging lab.	
5	A1 deep dive; code coverage and testing++.	A1 due (Feb 6); A2 assigned; Midterm One (Feb 7)
6	Scripting and Python intro; C–Python integration with ctypes; reading week begins.	
7	Post-reading-week catch-up; A2 deep dive; ctypes lab.	
8	Refactoring; OO Python.	A2 due (Mar 6); A3 assigned; 40th class day
9	A3 intro (Python/ctypes/refactoring review); MVC and curses.	Midterm Two (Mar 14)
10	More curses; serialization and persistence; curses lab.	
11	UI testing; code quality.	A3 due (Mar 27)
12	Project demos; exam review (or additional demos). *Final Exam: April 14, 11:30 am – 1:30 pm*	

[1] Unless otherwise noted, course handouts, slides, assignments, and exams are provided for your personal learning in this course. Redistribution or commercial use is not permitted.

Assessments and Grade Calculation

Exams

There will be two in-person midterms and an in-person final exam. Exams will be multiple choice format focusing on concepts covered in lectures, labs, and assignments. Makeup and deferred exams may be in a different format than the regular exam.

Exams:

Item	Weight	Due / Date
Midterm 1	15%	Saturday, Feb 7
Midterm 2	15%	Saturday, Mar 14
Final exam	30%	April 14, 11:30 am – 1:30 pm

Assignments

This course includes four programming assignments that build toward the development of an interactive software system. Students will begin by writing systematic tests for a provided software component, then gradually implement and extend a modular application that integrates C and Python code. Emphasis is placed on clean interfaces, methodical testing, and maintaining consistent program state across components.

Assignments:

Item	Weight	Due / Date
A0: Unit testing a provided C library	5%	Jan 16 @11:59 pm
A1: C library creation following API	10%	Feb 6 @11:59 pm
A2: Python CLI, Integration (C + Python)	10%	Mar 6 @11:59 pm
A3: Text-based UI, Persistence, Modularity, Demo	15%	Mar 27 @11:59 pm

Assignment Alignment with Learning Outcomes

	LO1	LO2	LO3	LO4	LO5
A0:				✓	✓
A1:	✓	✓		✓	✓
A2:	✓	✓		✓	✓
A3:	✓	✓	✓	✓	✓

Assignment Policies

- **Submission Method:** Assignments must be submitted via `git` to `gitlab.socs.uoguelph.ca` using the repository and organization specified in the assignment details. The assignment autograder runs immediately on each push and provides feedback in a separate branch of the student repository. Students may push multiple times before the due date, correcting issues as needed. The most recent correctness score assigned by the autograder is the one that will be used in calculating the assignment grade.
- **Submission Errors:** Students are responsible for ensuring that their repository is correctly configured. Incorrect submission structure (e.g., faulty or missing `Makefile`, missing files, misconfigured repository) will result in penalties up to and including a grade of zero for the assignment.

- **Compilation Requirement:** All submissions must compile and run *without errors or warnings* in the course CI environment as specified in the assignment details. Submissions that fail to compile will receive a grade of zero.
- **Late Submission Period and Penalties:** Each assignment has a 2 day late submission period following the deadline. A penalty of 1% per hour (or part thereof) during the late submission period applies to work submitted during this period. Submissions made after the late-submission deadline will not be graded unless formal academic consideration has been approved.
- **Extensions:** Extensions are granted only in cases of serious disruption (e.g., system outages or officially declared closures). Extensions are not granted for workload conflicts, travel, or similar reasons.
- **Accommodations:** Academic consideration for illness, compassionate grounds, religious obligations, or registered SAS accommodations must follow official University procedures. Documentation may be required. Students seeking such consideration must notify the instructor before the assessment deadline, and in all cases no later than 5:00 pm on the day the assessment is due.
- **Regrades:** If you believe an error has been made in the grading of your work you may submit a request for a regrade **in writing to the Regrades dropbox on the course website within 2 calendar days of grade release**. No other regrade request format will be accepted. The entire work will be re-evaluated, and the grade may increase or decrease. **The original submission is regraded**. A regrade is not a redo. If a submission has been regraded, it will not be regraded again.

Grading

Each programming assignment is graded on two dimensions: Correctness and Concept Mastery. A low mastery score can substantially reduce your grade. Failure to complete the concept mastery assessment results in 0 for that component. Unless specified otherwise in the assignment description, assignments are graded according to formula noted below.

Assignment Grading

1. **Correctness** Determined by the autograder, which runs your submission through automated tests. Score: 0–100.
2. **Concept Mastery** Assesses your understanding of principles, design choices, and implementation details. Usually an interview (Zoom), but may be written. Score: 0.0–1.0.

$$\text{Assignment Grade} = \text{Correctness} \times \text{Concept Mastery}$$

Example: Correctness $= 85$, Mastery $= 0.8 \Rightarrow$ Final Score $= 68$.

The final course grade is the sum of the weighted exam and assignment components.

Final Grade Calculation

Your final course grade is normally the sum of the **assignment component** (40%) and the **exam component** (60%).
However, you must pass *both* components in order to pass the course:

- If you pass both components, your final grade is the sum of the two.
- If you fail either component, your final course grade will be the grade of the failed component.
- If you fail both components, your final course grade will be the lower of the two failed component grades.
- The maximum recorded course grade cannot exceed 100%.

Examples:

- Assignments = 25/40 (62.5%), Exams = 42/60 (70%) ⇒ You pass both. Final grade = 25+42 = 67
- Assignments = 15/40 (37.5%), Exams = 30/60 (50%) ⇒ You fail the assignment component. Final grade = 37.5.
- Assignments = 19/40 (48%), Exams = 10/60 (16.7%) ⇒ You fail both. Final grade = 16.7.

Course Policies

If you require accommodations, please connect with the appropriate campus office and also let me know how I can support your learning. Information: https://www.uoguelph.ca/accessibility/.

- **Course Communication:** Use lectures, lab sessions, and the website discussion forum as your main opportunities to ask questions about the course. Questions that are specific to your particular situation may be emailed to cis2750@socs.uoguelph.ca and will be answered by one of the instructional team. Technical questions related to the programming assignments should be sent to the discussion forum on the course website. Extremely private communication should be conducted by making an appointment with the course instructor.

  Please note that the in-class discussions, course email, course forums, and the instructorTA office hours are the only means of communications for this course. Any other attempts at communication will not get a response.

  Major announcements will be posted to the course website and the discussion forums. It is your responsibility to check the course website regularly. As per university regulations, all students are required to check their <mail.uoguelph.ca> e-mail account regularly: e-mail is the official route of communication between the University and its students.

- **Reweighting of Grades:** Considered only in exceptional circumstances and on a case-by-case basis when other forms of academic consideration are insufficient. Reweighting will **not** be applied to accommodate individual preferences or to offset poor performance. Reweighting, when it occurs, is always **within the same assessment component** (e.g., across assignments, or across exams), and will not shift weight from one component to another. For example, marks will not be moved from assignments to exams or vice versa.

- **Sources of Information:** The only reliable sources of information about course deliverables are the teaching team, the official course website including the discussion forums on that website, and messages sent to your `uoguelph.ca` email address. While platforms like Discord, Reddit, Instagram and other social media can be useful for discussion, you must verify any assumptions about deliverables or requirements with the teaching team via official communication channels. Unverified information from social media or peers will not be accepted as grounds for regrades, extensions, or reweighting of assessments.

- **Academic Integrity:** You are responsible for understanding and following university policies on academic integrity (https://guides.lib.uoguelph.ca/AcademicIntegrity/). Proof of completion of the SoCS Academic Integrity Unit is required before any grades will be returned.
- **Oral Competency Exams:** You may be evaluated through an in-person oral exam to verify that submitted work is your own and that you understand the material. Results may affect the grade for the associated work. All graded work must be done independently unless collaboration is explicitly permitted. Any work that appears to be primarily created by AI or in unauthorized collaboration will not be graded. Cases of unauthorized collaboration or AI use will be forwarded to the Associate Director.
- **Grades:** All grades in this course are reported as percentages. The official interpretation of percentage grades (e.g., letter equivalents, standing) is governed by the University's grading policy: https://calendar.uoguelph.ca/undergraduate-calendar/undergraduate-degree-regulations-procedures/grades/
- **Institutional Policies:** Standard institutional statements and policies (e.g., academic consideration, religious accommodations, recording of lectures, copyright, email communication, and others) are available online.These policies form part of this course outline. https://uoguelphca.sharepoint.com/sites/Syllabi/SitePages/Standard-Statements---Undergraduate-Courses.aspx.