CS3307A Course Outline

Object-Oriented Design & Analysis

Fall 2026

1. Course Information

1.1 Course Identification

Course	COMPSCI 3307A – Object-Oriented Design & Analysis
Term	Fall 2026
Delivery mode	In person
Tuesday meeting	9:30–10:30 AM, [redacted]
Thursday meeting	9:30–11:30 AM, [redacted]

1.2 Course Description

This course introduces students to the principles and practices of Object-Oriented Design and Analysis (OODA), with implementation in C++. It covers fundamental object-oriented concepts, UML modeling techniques, and SOLID design principles, followed by an in-depth exploration of design patterns across creational, structural, and behavioral categories.

Students will also learn software testing practices, including unit testing and mocking, to validate design and implementation. Through lectures, case studies, design exercises, and a comprehensive final project, the course emphasizes building scalable, maintainable, testable, and efficient software systems that reflect real-world development challenges.

1.3 Prerequisites and Antirequisites

Prerequisites: Either Computer Science 2212A/B/Y; or Computer Science 2210A/B, Computer Science 2211A/B, Electrical and Computer Engineering 3375A/B, and registration in the fourth year of a BESc program in Computer Engineering or Mechatronic Systems Engineering.

Antirequisite: Software Engineering 3350A/B.

> Unless students have either the prerequisites for this course or written special permission from the Department of Computer Science to enroll in it, they may be removed and withdrawn from this course in accordance with University policy. This may be done after the add/drop deadline of the academic term, and the course will be marked as withdrawn (WDN) on their academic record. This decision may not be appealed.

1.4 Course Goal

The primary goal of this course is to equip students with the knowledge and skills needed to design, model, implement, and critically evaluate robust object-oriented software systems. The course combines theoretical foundations with practical application so that students learn not merely *how* particular principles and patterns work, but *when*, *why*, and *whether* they should be used.

Students will analyze requirements, construct models, apply design principles, select appropriate patterns, implement

solutions in C++, test their systems, and justify their design decisions.

1.5 Learning Outcomes

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

1. Explain and apply the core concepts of object-oriented programming, including encapsulation, inheritance, polymorphism, and abstraction.
2. Analyze software requirements and translate them into effective UML models, including class, sequence, and use case diagrams.
3. Apply SOLID principles to improve code quality, reduce coupling, and increase maintainability.
4. Identify, implement, and justify appropriate creational, structural, and behavioral design patterns for recurring software-design problems.
5. Develop scalable and maintainable C++ systems that integrate object-oriented design principles.
6. Use unit-testing frameworks, including GoogleTest and Google Mock, to evaluate the correctness and robustness of object-oriented systems.
7. Critically evaluate software designs with respect to scalability, maintainability, extensibility, testability, and efficiency.
8. Work independently and collaboratively to develop a complete software project from requirements gathering through design, implementation, and testing.

2. Instructor Information and Communication

Instructor	Dr. Umair Rehman
Role	Course Coordinator
Email	[redacted]
Office	[redacted]
Phone	[redacted]
Office hours	[redacted], or [redacted]

Students must use their Western (`@uwo.ca`) email addresses when contacting the instructor.

2.1 Course Communication Guidelines

Brightspace. OWL Brightspace is the primary source for course announcements, lecture materials, project specifications, due-date information, and other course updates. Students are responsible for checking Brightspace regularly.

Email. Email should be used for formal or individual communication. Students should include the course number followed by a brief description of the purpose of the message in the subject line; for example, `CS3307 - Project Query` or `CS3307 - Request for Meeting`.

MS Teams. MS Teams may be used for office hours, brief questions, clarifications, and real-time course communication.

3. Fall 2026 Calendar and Course Schedule

3.1 Key Sessional Dates

Date	Day	University / Course Impact
September 9, 2026	Wednesday	Fall classes begin. CS3307A first meets on Thursday, September 10.
September 17, 2026	Thursday	Last day to add a Fall/Winter 24-week course or a Fall 12-week course. Regular CS3307A class day.
September 30, 2026	Wednesday	National Day for Truth and Reconciliation; no classes at Western. This does not cancel a regularly scheduled CS3307A meeting because the course meets Tuesday/Thursday.
October 10–18, 2026	Saturday–Sunday	Thanksgiving and Fall Reading Week. No CS3307A classes on Tuesday, October 13 or Thursday, October 15.
November 30, 2026	Monday	Last day to withdraw from a Fall 12-week course without academic penalty.
December 9, 2026	Wednesday	Last day of Fall-term classes. The final scheduled CS3307A meeting is Tuesday, December 8.
December 10, 2026	Thursday	Study Day; no classes.
December 11–22, 2026	Friday–Tuesday	December examination period. The final examination date and time will be assigned by the Registrar.

3.2 Tentative Course Schedule

The schedule below reflects the Fall 2026 calendar and the planned progression of course content. Minor adjustments may be made to pacing or sequencing as the term progresses; any substantive change will be communicated through Brightspace.

Week	Class Dates	Topics and Coverage
1	Thu. Sept. 10	**Introduction to Object-Oriented Design and Analysis.** Course overview; object-oriented thinking and software decomposition; comparison with functional approaches; cohesion and coupling as measures of design quality; maintainability, scalability, extensibility, and reuse.
2	Tue. Sept. 15 Thu. Sept. 17	**Core Object-Oriented Concepts.** Encapsulation, inheritance, polymorphism, and abstraction; abstract classes and interfaces; composition versus inheritance; modularity, flexibility, and reuse; common design pitfalls when object-oriented mechanisms are overused or misapplied.
3	Tue. Sept. 22 Thu. Sept. 24	**Requirements Gathering and UML Modeling.** Requirements elicitation; functional and non-functional requirements; use-case modeling; identifying candidate classes and relationships; UML class, sequence, and use case diagrams; moving from problem understanding to an implementable design.

Continued on next page

Week	Class Dates	Topics and Coverage
4	Tue. Sept. 29 Thu. Oct. 1	**SOLID Principles and Design Practices.** Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation, and Dependency Inversion principles; refactoring exercises; dependency management; reducing coupling and improving cohesion; recognizing when a design principle meaningfully improves a system.
5	Tue. Oct. 6 Thu. Oct. 8	**Creational Design Patterns I.** Why design patterns exist and how to evaluate when they are useful; Singleton, Factory Method, and Abstract Factory; object-creation responsibilities; flexibility and trade-offs in construction logic.
–	Oct. 10–18	**Thanksgiving and Fall Reading Week – No Classes.** No CS3307A meetings on Tuesday, October 13 or Thursday, October 15.
6	Tue. Oct. 20 Thu. Oct. 22	**Creational Design Patterns II.** Builder and Prototype; step-by-step construction and cloning; choosing among creational patterns; combining patterns; object lifecycle and creation strategies in larger systems.
7	Tue. Oct. 27 Thu. Oct. 29	**Structural Design Patterns I.** Adapter, Composite, and Decorator; reconciling incompatible interfaces; representing part-whole hierarchies; adding responsibilities dynamically; designing structures for reuse and extension.
8	Tue. Nov. 3 Thu. Nov. 5	**Structural Design Patterns II.** Façade, Proxy, Bridge, and Flyweight; simplifying complex subsystems; controlling or deferring access; separating abstractions from implementations; memory-efficient object sharing; combining structural patterns in realistic systems.
9	Tue. Nov. 10 Thu. Nov. 12	**Behavioral Design Patterns I.** Iterator and Command; traversing collections without exposing internal representation; encapsulating requests as objects; undo/redo, logging, queuing, and flexible command execution.
10	Tue. Nov. 17 Thu. Nov. 19	**Behavioral Design Patterns II.** Strategy, Observer, State, Chain of Responsibility, and Template Method; runtime behavior selection; publish–subscribe relationships; state-dependent behavior; request delegation; reusable algorithm structure.
11	Tue. Nov. 24 Thu. Nov. 26	**Software Testing for Object-Oriented Systems.** Unit testing with GoogleTest; assertions, fixtures, and parameterized tests; Google Mock, matchers, and dependency isolation; testing interactions among objects; the relationship between testability and sound software design.
12	Tue. Dec. 1 Thu. Dec. 3	**Integrating Object-Oriented Design.** Connecting requirements, UML, SOLID principles, design patterns, implementation, and testing; design critique and refactoring; evaluating alternative architectures; integrating the major course concepts in preparation for project completion and the final examination.

Continued on next page

Week	Class Dates	Topics and Coverage
13	Tue. Dec. 8	**Industry Perspective and Course Review.** Application of object-oriented design ideas in contemporary software practice; practical design trade-offs; synthesis of course concepts; final review and examination preparation. An industry-oriented discussion or guest contribution may be included subject to availability.

4. Course Materials and Technical Requirements

4.1 Required and Recommended Learning Materials

There is **no required textbook** for this course.

> **Required learning-material cost: $0.** All required lecture materials, project specifications, examples, and supplementary resources will be made available through Brightspace or through freely accessible resources identified by the instructor.

The following books are optional references. Students are **not required to purchase** them, and older/second-hand editions may be used where appropriate:

- Bjarne Stroustrup, *The C++ Programming Language*, 4th Edition.
- Bjarne Stroustrup, *Programming: Principles and Practice Using C++*, 2nd Edition.
- Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*.
- Martin Fowler, *UML Distilled: A Brief Guide to the Standard Object Modeling Language*, 3rd Edition.
- Grady Booch et al., *Object-Oriented Analysis and Design with Applications*, 3rd Edition.

Because these materials are optional, their purchase cost is not a required course expense. Students who wish to buy them should consult the Western Bookstore or another bookseller for current pricing.

4.2 Brightspace

All course material will be posted to OWL Brightspace: https://westernu.brightspace.com/.

Students are responsible for checking the course Brightspace site regularly for news and updates. This is the primary method by which information will be disseminated to the class.

Brightspace assistance is available at https://brightspacehelp.uwo.ca/. Students may also contact the Western Technology Services Helpdesk at [redacted] or [redacted].

4.3 Technical Requirements

Students require access to a computer capable of developing, compiling, and testing C++ software. The course will use C++, GoogleTest, and Google Mock. No paid programming software is required.

Students are responsible for maintaining appropriate backups of project work and ensuring that submitted software can be compiled and evaluated according to the instructions provided for each assessment.

5. Methods of Evaluation

5.1 Grading Scheme

Assessment	Weight
Project Proposal	20%
Intermediate Design and Partial Implementation	30%
Final Project Submission	30%
Final Examination	20%
Total	**100%**

5.2 Final Project – 80%

The major component of the course is a comprehensive software-design project. **The project must be completed in pairs of two students.**

Students will design, model, and implement a non-trivial software system in C++ demonstrating mastery of Object-Oriented Design and Analysis. The project will require students to:

- perform requirements gathering and analysis;
- develop appropriate UML models, including Class, Sequence, and Use Case diagrams;
- apply appropriate creational design patterns;
- apply at least one appropriate structural or behavioral design pattern;
- demonstrate appropriate use of encapsulation, inheritance, polymorphism, and abstraction;
- apply relevant SOLID principles;
- produce clean, modular, maintainable C++ code;
- develop a testing suite using GoogleTest and Google Mock; and
- explain and justify important design decisions.

The project is divided into three milestones so that students move progressively from requirements and conceptual design toward implementation, testing, and final integration. Detailed requirements and marking criteria will be posted on Brightspace.

Project Proposal – 20%

Due: Tuesday, October 6, 2026 at 11:59 PM.

Students will establish the problem, requirements, proposed functionality, preliminary architecture, and initial modeling strategy for their system.

Intermediate Design and Partial Implementation – 30%

Due: Tuesday, November 10, 2026 at 11:59 PM.

Students will submit a substantially developed software design, accompanying UML models, justification of design choices, and partial implementation demonstrating meaningful progress toward the complete system.

Final Project Submission – 30%

Due: Tuesday, December 8, 2026 at 11:59 PM.

Students will submit the completed system, final UML documentation, implementation, testing suite, and supporting design documentation.

5.3 Final Examination – 20%

The final examination will be scheduled by the Office of the Registrar during the official **December 11–22, 2026** examination period. The specific date, time, and location will be announced when the Registrar releases the examination schedule.

The examination focuses on students' ability to analyze and improve object-oriented software. Students may be asked to:

- analyze C++ code and identify design weaknesses;
- refactor code using object-oriented principles;
- apply SOLID principles;
- select or apply appropriate design patterns;
- explain and justify design decisions; and
- demonstrate understanding of core C++ object-oriented design and implementation concepts.

The final examination assesses each student's own independent understanding of the course material.

5.4 Assessment Calendar

Assessment	Listed Deadline	Weight	48-Hour Window
Project Proposal	Tue. Oct. 6, 11:59 PM	20%	Through Thu. Oct. 8, 11:59 PM
Intermediate Design & Partial Implementation	Tue. Nov. 10, 11:59 PM	30%	Through Thu. Nov. 12, 11:59 PM
Final Project Submission	Tue. Dec. 8, 11:59 PM	30%	Through Thu. Dec. 10, 11:59 PM
Final Examination	Registrar scheduled, Dec. 11–22	20%	Not applicable

The listed deadline is the expected submission deadline. The automatic 48-hour no-late-penalty period is described in Section 6.

6. Missed Coursework and Academic Consideration

Students must familiarize themselves with the University Policy on Academic Consideration – Undergraduate Students in First Entry Programs:

https://uwo.ca/univsec/pdf/academic_policies/appeals/academic_consideration_Sep24.pdf

Procedures for submitting Academic Consideration requests are posted by the Office of the Registrar:

https://registrar.uwo.ca/academics/academic_considerations/

All requests for Academic Consideration must be made **as soon as possible and no later than 48 hours after the missed assessment**. The Academic Consideration policy does not apply to requests for attempted or completed work. Students experiencing longer-term impacts on their academic responsibilities should consult Accessible Education.

6.1 Project Deadlines and the Automatic 48-Hour No-Late-Penalty Period

> Each project deliverable includes an automatic **48-hour no-late-penalty period** following the stated deadline. Students do not need to request Academic Consideration in order to use this period.

The listed deadline remains the expected submission deadline and students should plan their work around it. The additional 48 hours provide built-in flexibility for short-term disruptions and should not be treated as a replacement due date.

After the 48-hour period closes, submissions will not normally be accepted without an approved Academic Consideration or other University-approved accommodation. Because the three project milestones assess **distinct learning outcomes and different stages of the software design and development process**, a missed project component will **not automatically be re-weighted** to another project component or to the final examination.

Where Academic Consideration is granted after the 48-hour period, an appropriate accommodation will be determined in accordance with University and Faculty policies and the learning outcomes associated with the missed component. Depending on the circumstances, this may involve an extension, a rescheduled or alternative submission, or another academically appropriate arrangement rather than automatic re-weighting.

Because the project is completed in pairs, students experiencing circumstances that substantially affect their ability to participate in collaborative work should contact the instructor as early as reasonably possible so that any accommodation can be implemented without unfairly disadvantaging the project partner.

6.2 Essential Learning Requirements

The course project is the principal mechanism by which students demonstrate integrated competence in requirements analysis, UML modeling, object-oriented design, design-pattern selection, C++ implementation, testing, and design justification. Students are therefore expected to participate meaningfully in the project and to complete the required project work.

Where an approved accommodation or Academic Consideration affects a student's ability to satisfy an essential learning requirement, the appropriate academic process will be followed in consultation with the student and the relevant Faculty office.

6.3 Special Examination

Per University policy, Academic Consideration requests with supporting documentation are required for final examinations scheduled during official examination periods by the Registrar's Office.

If a student has been approved by the Academic Advising Office to write a Special Examination (SPC) and the final examination is their only outstanding course component, an SPC will be assigned and scheduled in accordance with University policy.

If the student misses the SPC exam and is approved for Academic Consideration, they will be withdrawn from the course without academic penalty. Where appropriate, the student may appeal to the Associate Dean (Academic), on a case-by-case basis, for the opportunity to write the final examination the next time the course is offered.

If the student misses the SPC exam and Academic Consideration is not approved, or was not requested, a grade of zero will be assigned for the examination.

7. Use of Generative AI Tools

Course position: AI-aware and AI-positive, with assessment-specific boundaries. Generative AI is permitted for designated project and learning activities when its use is transparent and the student remains responsible for the quality, correctness, and intellectual defensibility of the submitted work. Generative AI is **strictly prohibited on the final examination**.

This course recognizes that generative AI tools are increasingly part of contemporary software-development practice. Tools such as ChatGPT, Microsoft Copilot, Google Gemini, Claude, and comparable systems may be used for designated learning activities and project work subject to the conditions below.

7.1 Permitted Uses for Project Work

Students may use generative AI to support activities such as:

- brainstorming;
- exploring alternative designs;
- explaining unfamiliar concepts;
- generating or reviewing code;
- debugging;
- refactoring;
- developing test cases;
- summarizing technical material;
- evaluating design alternatives; and
- improving documentation.

Students remain responsible for **all material they submit**. Use of an AI system does not transfer responsibility for correctness, quality, security, attribution, or understanding to the system.

Students should be capable of explaining and defending the architecture, design patterns, code, UML models, testing decisions, and other substantive material included in their submissions.

7.2 Disclosure Requirement

Where generative AI materially contributes to assessed project work, students must identify:

- the tool used;
- the general purpose for which it was used; and
- the nature of the contribution made by the tool.

Individual project instructions may specify a more detailed disclosure format. Misrepresenting AI-generated content as entirely one's own work without the disclosure required by the course may constitute a scholastic offence.

7.3 Verification Responsibility

Students are responsible for checking AI-generated content before using or submitting it. Generative systems can produce incorrect code, fabricated APIs, insecure implementations, poor design decisions, incorrect explanations, and superficially plausible but inappropriate uses of design patterns.

A central learning objective in this course is therefore not simply producing software, but developing the judgment required to determine whether a proposed design or implementation is actually appropriate.

7.4 Final Examination Exception

> **Generative AI is strictly prohibited during the final examination.** Exam performance must reflect each student's own independent understanding and ability to analyze and apply course concepts. Unless explicitly authorized by the instructor, possession or use of a device capable of accessing generative AI during the examination is prohibited under the electronic-device rules below.

If students are uncertain about whether a particular use of AI is permitted for an assessment, they are responsible for seeking clarification in advance.

8. Additional Academic Policies

8.1 Religious Accommodation

When a recognized religious holiday either requires a student to be absent from the University, or requires or prohibits an activity that makes it impossible to complete an academic requirement scheduled that day, the student is entitled to accommodation. No student will be penalized for an absence for religious reasons, and an alternative means of satisfying the requirement will be arranged.

University policy: https://www.uwo.ca/univsec/pdf/academic_policies/appeals/accommodation_religious.pdf

Students should submit requests through the Student Absence Portal (SAP), identifying the conflict and the specific course component affected.

If a religious observance conflicts with the final examination, the student must submit the request no later than **two weeks before the examination**. An appropriate Special Examination will be arranged in accordance with University policy.

Students are responsible for finding out what was covered in any class missed due to a religious observance and for taking the appropriate steps to remain current with the course.

Recognized religious-observance calendars are available through Western's Equity, Diversity & Inclusion website: https://www.edi.uwo.ca/.

8.2 Academic Accommodation Policies

Students with disabilities are encouraged to contact Accessible Education, which provides recommendations for accommodation based on appropriate documentation and assessment.

Policy on Academic Accommodation for Students with Disabilities: https://www.uwo.ca/univsec/pdf/academic_policies/appeals/Academic%20Accommodation_disabilities.pdf

Students who require lecture or printed material in an alternate format may also contact the instructor.

8.3 General Academic Policies

Registrar Services: https://www.registrar.uwo.ca/

Use of `@uwo.ca` email. In accordance with University policy, the centrally administered email account provided to students will be considered the individual's official University email address. Students are responsible for ensuring that emails received at that address are attended to in a timely manner.

Requests for Relief.

- Policy: https://uwo.ca/univsec/pdf/academic_policies/appeals/requests_for_relief_from_academic_decisions.pdf
- Undergraduate procedures: https://uwo.ca/univsec/pdf/academic_policies/appeals/undergrad_requests_for_relief_procedure.pdf

8.4 Scholastic Offences

Policy on Scholastic Offences: https://uwo.ca/univsec/pdf/academic_policies/appeals/scholastic_offences.pdf

Procedures on Scholastic Offences (Undergraduate): https://uwo.ca/univsec/pdf/academic_policies/appeals/undergrad_scholastic_offence_procedure.pdf

8.5 Use of Electronic Devices During Assessments

In courses offered by the Faculty of Science, the **possession of unauthorized electronic devices during any in-person assessment** (such as tests, midterms, and final examinations) is strictly prohibited. This includes, but is not limited to, mobile phones, smart watches, smart glasses, hidden cameras, and wireless earbuds.
Unless explicitly stated otherwise in advance by the instructor, the presence of any such device at a student's desk, on their person, or within reach during an assessment will be treated as a scholastic offence, **even if the device is not in use**.
Effective September 2026, the typical penalty for this offence is a **failing grade in the course and a 12-month prohibition on retaking the course**. Only devices expressly permitted by the instructor may be brought into the assessment room.

8.6 Software Similarity and Assessment Tools

Course submissions may be subject to textual or source-code similarity analysis where appropriate. Written material may be reviewed using plagiarism-detection software licensed by the University. Programming submissions may also be evaluated using source-code similarity tools such as MOSS or comparable software to identify unusual similarities between submissions.

Similarity itself does not automatically establish misconduct; identified cases may be reviewed in accordance with University scholastic-offence procedures.

9. Support Services

9.1 Academic Advising

The Science & Basic Medical Sciences Academic Advising office provides information concerning adding/dropping courses, academic consideration, requests for relief, examination conflicts, and other academic matters: https://www.uwo.ca/sci/advising/

9.2 Mental Health Support

Students who are in emotional/mental distress should refer to Mental Health@Western (https://uwo.ca/health/) for a complete list of options about how to obtain help.

9.3 Gender-Based and Sexual Violence

Western is committed to reducing incidents of gender-based and sexual violence (GBSV) and providing compassionate support to anyone who has gone through these traumatic events. If a student has experienced GBSV (either recently or in the past), they can find information about support services for survivors, including emergency contacts, at:

https://www.uwo.ca/health/student_support/survivor_support/get-help.html

To connect with a case manager or set up an appointment, please contact [redacted].

9.4 Accessibility for Coursework

Students may contact the instructor if lecture or printed material is required in an alternate format or if other reasonable arrangements could make the course more accessible.

Accessible Education: http://academicsupport.uwo.ca/accessible_education/index.html

9.5 Learning Development and Success

Learning Development and Success provides support in areas such as time management, study strategies, examination preparation, textbook reading, and general learning skills: https://learning.uwo.ca/

9.6 Student Councils

Student-run support services are available through the University Students' Council (https://westernusc.ca/) and the Science Students' Council (https://www.westernssc.ca/).