CS3307A Course Outline - Fall 2024

1. Course Information

Course Information
COMPSCI 3307A - OBJECT-ORIENTED DESIGN & ANALYSIS

Time/Place
9:30 - 10:30 AM, Tuesdays [redacted]
9:30 - 11:30 AM, Thursdays [redacted]

2. Instructor Information

Instructors	Email	Office	Phone	Office Hours
Dr. Umair Rehman (Course Coordinator)	urehman6@uwo.ca	[redacted]	[redacted]	MS Teams, Tuesday 2:00 PM – 3:00 PM

Course Communication Guidelines

Primary Channels for Communication

- Email: Formal communication, such as course updates and assignment submissions, should be conducted via email. Students are required to use their Western (@uwo.ca) email addresses when contacting instructors. This is essential for maintaining a formal and secure line of communication.
- MS Teams: For real-time interactions, queries about course material, and brief updates, MS Teams will be utilized. This platform facilitates more immediate and interactive communication.

Email Etiquette

- Subject Line: When sending an email, students must include the course number followed by a brief description of the email's purpose in the subject line. For example, "CS3307 – Project Query" or "CS3307 – Request for Meeting".
- Instructor Messaging: The instructor's email is integrated into MS Teams, allowing students to message the instructor directly through the platform. This should be used for quick questions or clarifications.

3. Course Description

In today's software development landscape, Object-Oriented Design and Analysis (OODA) is an essential approach for building complex, high-quality systems. This course delves into the key aspects of OODA, focusing on the process of analyzing system requirements and translating them into effective, scalable software designs. You will explore the use of Unified Modeling Language (UML) to model system interactions, structures, and behaviors, and learn how to leverage established design patterns to optimize the development process.

Through this course, you will gain a deep understanding of how to design software systems that meet user requirements and operate efficiently within their intended environments. By working with C++, you will implement these designs and understand how object-oriented principles are applied in real-world programming. The course includes a combination of theoretical learning, practical assignments, and a significant project where you will design and implement a large-scale system using C++.

Course Goal

The goal of this course is to equip students with the expertise to perform rigorous system analysis and create robust, maintainable software designs using Object-Oriented methodologies. By the end of the course, students will be proficient in using UML for modeling, applying design patterns to streamline the design process, and implementing their designs in C++. This will prepare them to tackle complex software development challenges and deliver high-quality solutions in professional settings.

Learning Outcomes

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

1. Clearly articulate the fundamental concepts of object-oriented design, including encapsulation, inheritance, polymorphism, and abstraction, and effectively apply these principles in designing software systems.
2. Demonstrate proficiency in identifying and applying appropriate design patterns to address common software design challenges, ensuring that systems are flexible, reusable, and maintainable.
3. Create detailed and accurate UML diagrams, including class diagrams, sequence diagrams, and use case diagrams, to visually represent and communicate complex software designs.
4. Critically evaluate existing software codebases, identify areas for improvement, and implement refactoring strategies to enhance the structure, performance, and maintainability of the code.
5. Develop software systems that adhere to key quality attributes such as scalability, security, and performance, using best practices and design principles to ensure the system meets both current and future requirements.
6. Convert software requirements and use cases into effective and efficient design specifications, ensuring alignment with business needs and technical constraints.
7. Reflect on the evolution of software systems, manage technical debt, and apply refactoring and design evolution techniques to maintain and enhance software quality over time.

4. Course Schedule

Week	Topics
Week 1: Introduction to Object-Oriented Design and Analysis Sept 5, 2024	1. Objects and Classes: Introduction to classes, objects, attributes, methods, access specifiers, and encapsulation in C++. 2. Object-Oriented Design and Analysis (OODA): Focus on designing systems with key objects and relationships, using use cases and step-by-step OOD. 3. OODA vs. Functional Programming: Comparison of core philosophies, state management, modularity, and error handling between OODA and functional programming. 4. Cohesion and Coupling: Importance of high cohesion and low coupling in system design, with examples of modular, maintainable software architectures.
Week 2: Core Object-Oriented Concepts Sept 10 and Sept 12, 2024	1. Encapsulation: Definition, importance, implementation, and examples. 2. Inheritance: Concept, types (single, multiple, hierarchical), benefits, and pitfalls. 3. Polymorphism: Types (compile-time, runtime), role in maintainable code, and examples. 4. Abstraction: Definition, comparison with encapsulation, use of abstract classes/interfaces. 5. Real-World Case Studies: Analysis of software systems to illustrate OOD concepts and their impact on software quality.
Week 3: SOLID Principles and Design Practices Sept 17 and Sept 19, 2024	1. Introduction to SOLID Principles: Overview and significance in software design. 2. Single Responsibility Principle (SRP): Definition, importance, examples, and refactoring strategies. 3. Open/Closed Principle (OCP): Understanding OCP, implementation, and case studies. 4. Liskov Substitution Principle (LSP): Explanation, impact on inheritance, and examples. 5. Interface Segregation Principle (ISP): Importance, design of fine-grained interfaces. 6. Dependency Inversion Principle (DIP): High-level vs. low-level modules, reducing coupling, and dependency injection.
Week 4: Requirements Gathering and Use Case Modeling Sept 24 and Sept 26, 2024	1. Introduction to Requirements Gathering: Importance in the software lifecycle, functional vs. non-functional requirements. 2. Techniques for Gathering Requirements: Interviews, surveys, observation, workshops. 3. Use Case Modeling: Purpose, elements (actors, scenarios), writing effective descriptions. 4. Identifying Use Cases: Techniques, prioritization, creating diagrams. 5. Practical Application: Hands-on exercises and case studies.

Week 5: Class and Object Modeling & UML Techniques Oct 1 and Oct 3, 2024	1. Identifying Classes and Relationships: Extract objects from use cases and define relationships (association, inheritance, composition). 2. Class Diagram Notation: Use UML to represent classes, attributes, methods, and relationships with clarity. 3. Refining Class Diagrams: Iteratively improve diagrams for consistency and simplicity. 4. Sequence and Use Case Diagrams: Model object interactions and user-system behavior using lifelines and scenarios. 5. Best Practices in UML: Tips to create effective, clear, and consistent UML diagrams, avoiding common pitfalls.
Week 6: Design Patterns (Part 1) - Creational Patterns Oct 8 and Oct 10, 2024	1. Introduction to Design Patterns: Importance and categories (Creational, Structural, Behavioral). 2. Singleton Pattern: Ensuring a single instance, implementation, and use cases. 3. Factory Method Pattern: Object creation, design variations, and real-world examples. 4. Abstract Factory Pattern: Creating families of related objects, comparison with Factory Method. 5. Builder Pattern: Step-by-step construction, differences from Factory patterns, examples.
Reading Week	
Week 7: Design Patterns (Part 2) - Structural and Behavioral Patterns Oct 22 and Oct 24, 2024	1. Adapter Pattern: Adapting interfaces, implementation, and case studies. 2. Composite Pattern: Treating objects and compositions uniformly, GUI/file system examples. 3. Decorator Pattern: Adding responsibilities dynamically, comparison with other patterns. 4. Observer Pattern: One-to-many dependency, implementation, and common applications. 5. Strategy Pattern: Family of algorithms, interchangeability, and real-world examples. 6. Command Pattern: Encapsulating requests, implementing undo/redo, use cases.
Week 9: Software Architecture and Design Patterns Oct 29 and Oct 31, 2024	1. Introduction to Software Architecture: Definition, significance, and distinction between software architecture and design. 2. Common Architectural Styles: Overview of styles such as layered, client-server, microservices, and event-driven architectures. 3. Integrating Design Patterns with Architecture: How design patterns align with architectural concerns to solve specific problems. 4. Examples of Architectural and Design Patterns: Includes MVC, Repository, SOA, and structural/behavioral patterns like Facade, Proxy, and Chain of Responsibility.

	5. Case Studies and Best Practices: Analysis of systems using a combination of architectural styles and design patterns, along with criteria for selecting patterns.
Week 8: Design for Maintainability and Reusability Nov 5 and Nov 7, 2024	1. Introduction to Software Quality Attributes: Overview of key attributes like maintainability, reusability, performance, scalability, and security. 2. Design for Maintainability and Reusability: Principles of simplicity, modularity, refactoring, creating reusable components, and using version control. 3. Performance and Scalability: Metrics, bottlenecks, optimization techniques, and strategies for horizontal/vertical scaling. 4. Security Considerations: Secure design principles, common threats, and best practices to safeguard systems. 5. Case Studies and Practical Applications: Analysis of systems focusing on maintainability, reusability, performance optimization, scalability, and security.
Week 10: Refactoring and Evolution of Software Design Nov 12 and Nov 14, 2024	1. Understanding Refactoring: Importance, signs that refactoring is needed. 2. Refactoring Techniques: Improving readability/maintainability, refactoring to patterns, automated tools. 3. Managing Technical Debt: Definition, impact, strategies for management and prioritization. 4. Continuous Integration and Continuous Deployment (CI/CD): Role in modern development, integrating refactoring into CI/CD pipelines. 5. Case Studies: Successful software evolution, challenges, and strategies for evolving legacy systems.
Week 11: Quality Assurance and Usability Testing Nov 19 and Nov 21, 2024	1. Code Inspections: Highlight the importance of reviewing object relationships, design patterns, and principles for maintainability and correctness. 2. Automated Testing: Introduce unit and integration testing to ensure object behavior and system integrity. 3. Design for Usability: Emphasize modularity and encapsulation to create user-friendly, adaptable designs. 4. Usability Testing: Show how OOD supports usability testing, such as through UI logging and patterns like MVC. 5. Iterative Development: Stress the role of iterative design, where inspections and usability feedback guide continuous refactoring.
Week 11: Case Studies and Real-World Applications Nov 26 and Nov 28, 2024	1. Comprehensive Case Studies: Analysis of complex systems, architectural/design decisions. 2. Industry Insights and Best Practices: Guest lectures, contemporary issues in software design. 3. Lessons from Real-World Failures: Case studies of failed designs, root causes, preventive measures.

	4. Ethical Considerations in Software Design: Discussing ethical implications, ensuring fairness, accountability, transparency.
Week 12: Industry Talk and Course Review Dec 3 and Dec 5, 2024	

5. Course Materials

Course Reference Material

There are no required textbooks for this course. However, the following books are recommended as valuable references:

- **The C++ Programming Language, 4th Edition** by Bjarne Stroustrup
- **Programming Principles and Practice Using C++, 2nd Edition** by Bjarne Stroustrup
- **Design Patterns: Elements of Reusable Object-Oriented Software** by Erich Gamma et al.
- **UML Distilled: A Brief Guide to the Standard Object Modeling Language, 3rd Edition** by Martin Fowler
- **Object-Oriented Analysis and Design with Applications, 3rd Edition** by Grady Booch et al.
- **Exploring Raspberry Pi: Interfacing to the Real World with Embedded Linux** by Derek Molloy

Additional references will be provided as needed. Please check Brightspace for updates.

Lecture Notes and Brightspace

- Lecture materials will be available on Brightspace (Online Web Learning) prior to each class.
- Regular checking of the course site on Brightspace is essential for news and updates. It is the primary method of information dissemination for this course.
- Assistance with the site can be sought on the Brightspace help page or by contacting the Western Technology Services Helpdesk at phone number 519-661-3800 or ext. 83800.

Course Delivery and Format

- The course is planned to be delivered in-person for the entire term, subject to any unforeseen complications.
- Any changes aligning with University guidelines will be communicated through announcements via Brightspace.

6. Methods of Evaluation

Final Project (80%)

The final project for CS3307A involves designing, modeling, and implementing a real-world software system using Object-Oriented Design and Analysis (OODA) principles in C++. Students may work individually (preferred) or in teams of two. Students will engage in requirements gathering, UML system modeling, and the application of design patterns to ensure their system is scalable, maintainable, and efficient. The project will unfold in three phases: a project proposal, an intermediate design with partial implementation, and a final submission with complete system implementation and documentation.

Students will be evaluated on their system design, UML diagrams, implementation, and testing, with a strong emphasis on the correct application of object-oriented principles and design patterns. Detailed guidelines and submission timelines are provided on Brightspace.

Final Exam (20%)
The final exam will evaluate your overall understanding of the course content, encompassing both fundamental and advanced OOD principles. The exam will include a mix of multiple-choice, short answer, and problem-solving questions. It will test your ability to apply the concepts and techniques discussed throughout the course. More information on the format and topics covered will be shared on Brightspace as the exam date approaches.

Due Dates of Deliverables

Deliverable	Due Date	Weight
Project Proposal	October 22, 2024	20%
Intermediate Design and Partial Implementation	November 19, 2024	30%
Final Project Submission	December 17, 2024	30%
Final Exam	December 12, 2024	20%

7. Student Absences

For the 2024-2025 academic year, the handling of absences in this course will follow Western University's updated Academic Consideration for Student Absences policy. The approach to academic considerations for each assessment type is outlined below:

Project Proposal (20%) – No built-in flexibility
Extensions for this component will only be granted through formal academic consideration. If a student is unable to submit the proposal by the deadline, they must request an academic consideration through the appropriate channels.

Intermediate Design and Partial Implementation (30%) – Limited built-in flexibility
Students are allowed to use one late coupon to extend the deadline by up to 48 hours without needing documentation. Beyond this, academic considerations will be required.

Final Project Submission (30%) – No built-in flexibility
Due to the importance of the final project and the proximity to the course end date, this submission will not have built-in flexibility. Extensions will only be granted under formal academic consideration.

Final Exam (20%) – No built-in flexibility
The final exam must be completed as scheduled unless a formal academic consideration is granted. In such cases, a makeup exam will be arranged.

The new academic considerations policy allows students to self-report absences for up to 48 hours without documentation. For more details on how to request academic considerations, please refer to the full policy here:

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

8. Accommodation and Accessibility

Religious Accommodation
When a course requirement conflicts with a religious holiday that requires an absence from the University or prohibits certain activities, students should request accommodation for their absence in writing at least two weeks prior to the holiday to the course instructor and/or the Academic Counselling office of their Faculty of Registration. Please consult University's list of recognized religious holidays (updated annually) at

https://multiculturalcalendar.com/ecal/index.php?s=c-univwo

Accommodation Policies
Students with disabilities are encouraged to contact Accessible Education, which provides recommendations for accommodation based on medical documentation or psychological and cognitive testing. The policy on Academic Accommodation for Students with Disabilities can be found at

https://www.uwo.ca/univsec/pdf/academic_policies/appeals/Academic Accommodation_disabilities.pdf.

9. Academic Policies

The website for Registrarial Services is http://www.registrar.uwo.ca.

In accordance with policy,

https://www.uwo.ca/univsec/pdf/policies_procedures/section1/mapp113.pdf,

the centrally administered e-mail account provided to students will be considered the individual's official university e-mail address. It is the responsibility of the account holder to ensure that e-mail received from the University at their official university address is attended to in a timely manner.

Scholastic offences are taken seriously and students are directed to read the appropriate policy, specifically, the definition of what constitutes a Scholastic Offence, at the following Web site:

http://www.uwo.ca/univsec/pdf/academic_policies/appeals/scholastic_discipline_undergrad.pdf.

All required submissions may be subject to submission for textual similarity review to the commercial plagiarism detection software under license to the University for the detection of plagiarism. All papers submitted for such checking will be included as source documents in the reference database for the purpose of detecting plagiarism of papers subsequently submitted to the system. Use of the service is subject to the licensing agreement, currently between The University of Western Ontario and Turnitin.com (http://www.turnitin.com).

10. Support Services

Please visit the Science & Basic Medical Sciences Academic Counselling webpage for information on adding/dropping courses, academic considerations for absences, appeals, exam conflicts, and many other academic related matters: https://www.uwo.ca/sci/counselling/.

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.

Western is committed to reducing incidents of gender-based and sexual violence and providing compassionate support to anyone who has gone through these traumatic events. If you have experienced sexual or gender-based violence (either recently or in the past), you will 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 support@uwo.ca.

Please contact the course instructor if you require lecture or printed material in an alternate format or if any other arrangements can make this course more accessible to you. You may also wish to contact Accessible Education at

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

if you have any questions regarding accommodations.

Learning-skills counsellors at the Student Development Centre (https://learning.uwo.ca) are ready to help you improve your learning skills. They offer presentations on strategies for improving time management, multiple-choice exam preparation/writing, textbook reading, and more. Individual support is offered throughout the Fall/Winter terms in the drop-in Learning Help Centre, and year-round through individual counselling.

Western University is committed to a thriving campus as we deliver our courses in the mixed model of both virtual and face-to-face formats. We encourage you to check out the Digital Student Experience website to manage your academics and well-being: https://www.uwo.ca/se/digital/.

Additional student-run support services are offered by the USC, https://westernusc.ca/services/.