# FINAL PROJECT GUIDELINES

**MASTER IN INFORMATION TECHNOLOGY (MIT)** **Course:** Advanced Object-Oriented Programming

## 1. Introduction

The Final Project is a capstone academic requirement for the course Advanced Object-Oriented Programming under the Master in Information Technology (MIT) program. This project is designed to assess the student’s ability to apply advanced object-oriented principles, software engineering methodologies, and professional development practices in the design and implementation of a substantial software system.

Students are expected to demonstrate analytical thinking, technical depth, and adherence to graduate-level academic and professional standards.

## 2. Course Learning Outcomes Addressed

Upon successful completion of the Final Project, students shall be able to:

1. Apply advanced object-oriented programming concepts in complex software systems.
2. Design scalable, maintainable, and extensible applications using appropriate abstractions.
3. Integrate established design patterns and architectural principles into software solutions.
4. Evaluate and justify design decisions using sound engineering judgment.
5. Produce professional-quality documentation and technical presentations.

## 3. Project Description and Scope

The Final Project shall involve the analysis, design, implementation, testing, and documentation of a non-trivial software application or framework. Projects may be individual or group-based, subject to the approval of the course instructor.

The project must be sufficiently complex to demonstrate mastery of advanced object-oriented programming concepts beyond introductory or intermediate levels.

## 4. Technical and Design Requirements

### 4.1 Core Object-Oriented Requirements

Each project must clearly demonstrate the following:

* Encapsulation and information hiding
* Abstraction through interfaces and/or abstract classes
* Proper use of inheritance and polymorphism
* Clear separation of responsibilities among classes
* Robust exception handling and defensive programming techniques

### 4.2 Advanced Object-Oriented Concepts

Students are required to implement and justify the use of **at least three** of the following advanced concepts:

* Software Design Patterns (e.g., Factory, Strategy, Observer, Singleton, MVC)
* SOLID design principles
* Dependency Injection
* Composition over Inheritance
* Immutable object design
* Generic programming (e.g., templates or generics)
* Event-driven or reactive architectures
* Reflection or meta-programming

## 5. Acceptable Project Categories

Proposed projects may include, but are not limited to:

* Enterprise or business-oriented systems
* Simulation or modeling applications
* Software frameworks or reusable libraries
* Concurrent or distributed systems
* Intelligent or rule-based systems
* Educational or visualization tools

*All project topics must receive prior approval from the instructor before development begins.*

## 6. Technical Constraints and Standards

* **Programming Language:** As prescribed by the course syllabus (e.g., Java, C++, Python, C#)
* **Version Control:** Mandatory use of Git-based repositories
* **Coding Standards:** Compliance with recognized industry or language-specific standards
* **Build and Execution:** The system must compile and execute successfully in a clean environment

## 7. Documentation Requirements

### 7.1 Required Deliverables

1. **Project Proposal**
   * Problem definition and objectives
   * Scope and limitations
   * High-level system architecture
2. **Design Documentation**
   * UML Class Diagram (mandatory)
   * UML Sequence or Activity Diagram (recommended)
3. **Technical Report / README**
   * System overview
   * Installation and execution procedures
   * Description of major components
   * Discussion of design decisions

### 7.2 Code-Level Documentation

* Appropriate inline comments where necessary
* Formal documentation (e.g., Javadoc or docstrings) for public classes and methods

## 8. Testing and Quality Assurance

Students are expected to:

* Develop unit tests for core modules
* Demonstrate correct handling of exceptional and boundary cases
* Ensure the system operates reliably under normal conditions
* *Automated testing frameworks are strongly encouraged.*

## 9. Evaluation Criteria

| **Evaluation Component** | **Weight** |
| --- | --- |
| Object-Oriented Design and Architecture | 30% |
| Application of Advanced OOP Concepts | 20% |
| Code Quality and Maintainability | 15% |
| Functional Correctness and Completeness | 15% |
| Documentation and UML Artifacts | 10% |
| Oral Presentation and Demonstration | 10% |

## 10. Project Presentation

Each student or group shall deliver a formal oral presentation of approximately 10–15 minutes, which shall include:

* Overview of the problem and solution
* Explanation of system architecture and design choices
* Demonstration of key functionalities
* Responses to technical questions from the evaluator

## 11. Academic Integrity and Ethical Standards

All submitted work must adhere to institutional policies on academic integrity. Specifically:

* Submitted code and documentation must be original.
* Any external libraries, frameworks, or references must be properly cited.
* Unauthorized collaboration or plagiarism will be subject to disciplinary action.
* Use of AI-assisted tools must be disclosed and is subject to instructor approval.

## 12. Submission Instructions

* Source code and documentation shall be submitted through the official Learning Management System (LMS) or approved repository link.
* Submissions must follow the prescribed directory and naming conventions.
* Late submissions shall be subject to penalties as defined in the course policy.

## 13. Concluding Statement

The Final Project represents a comprehensive assessment of the student’s readiness to apply advanced object-oriented programming techniques in professional and research-oriented environments. Emphasis shall be placed not only on functional correctness, but also on design quality, clarity, and long-term maintainability.