CIT 207 – OBJECT-ORIENTED PROGRAMMING
FINAL PROJECT RUBRICS

Project Requirements Overview

The program should:

- Demonstrate encapsulation, inheritance, abstraction, and polymorphism.
- Include one-to-one and one-to-many relationships.
- Be interactive with validated user input.
- Contain computational logic that models real-world processes.
- Implement CRUD functionalities.
- Have comprehensive error handling.
- Include a clear and consistent class diagram.
- Conduct a project presentation and Q&A session.
- Submit peer and self-evaluation.

Criteria	Description	Total Points	Scoring Guide
1. Program Design and Architecture		**25**	
Encapsulation	Private fields with getters and setters properly implemented.	5	5 – All classes correct 4 – Minor issues 3 – Inconsistent 2 – Few classes applied 1 – Missing
Inheritance	Correct use of parent-child relationships.	5	5 – Efficient and clear hierarchy 4 – Slight redundancy 3 – Partly structured 2 – Incorrect logic 1 – Not used.
Abstraction	Use of abstract classes or interfaces for generalization.	5	5 – Proper abstraction 4 – Slight misuse 3 – Minimal 2 – Incomplete 1 – Absent

Polymorphism	Dynamic method overriding or overloading used effectively.	5	5 – Consistent and purposeful 4 – Mostly functional 3 – Minimal use 2 – Limited 1 – None
Logical Architecture	Modular organization and maintainable code structure.	5	5 – Clean, modular, and scalable 4 – Slightly uneven 3 – Partially structured 2 – Disorganized 1 – Confusing
2. Relationships		**10**	
One-to-One Relationship	Properly linked individual entities.	5	5 – Accurate and functional4 – Minor inconsistency3 – Partial2 – Incorrect1 – None
One-to-Many Relationship	Implemented using collections for multiple associations.	5	5 – Fully correct 4 – Minor logic issues 3 – Basic but functional 2 – Flawed 1 – Not implemented
3. Functionality, Logic, and Error Handling		**30**	
CRUD Functionality	Fully functional create, read, update, and delete.	10	10 – All operations correct 8–9 – Minor issues 6–7 – Missing one function 4–5 – Weak implementation 1–3 – Incomplete
Computational Logic	Realistic operations that reflect real-world processes.	8	8 – Accurate and efficient 6–7 – Minor flaws 4–5 – Simplistic logic 2–3 – Illogical process 1 – None

User Interactivity	Properly handles user input and provides meaningful feedback.	6	6 – Clear and user-friendly 5 – Minor validation issues 4 – Average interaction 2–3 – Weak interaction 1 – None
Error Handling	Correct handling of exceptions and invalid inputs.	6	6 – Robust exception management 5 – Handles major cases 4 – Some inputs unchecked 2–3 – Minimal error catching 1 – Absent.
4. Code Quality and Readability		**15**	
Code Style and Formatting	Clean indentation, consistent naming, readable structure.	8	8 – Professional and consistent 6–7 – Few inconsistencies 4–5 – Acceptable but mixed 2–3 – Poor formatting 1 – Unclear
Efficiency and Reusability	Code organized for reuse and resource efficiency.	7	7 – Modular and efficient 5–6 – Minor redundancy 3–4 – Some inefficiency 2 – Repetitive 1 – Wasteful and non-modular
5. Class Diagram		**10**	
Accuracy and Completeness	Correct depiction of classes, attributes, methods, and relationships.	4	4 – Fully accurate and complete 3 – Minor errors 2 – Partially aligned 1 – Incomplete
Consistency and Conventions	Follows proper UML notation, naming conventions, formatting, and consistency with code.	3	3 – All conventions followed 2 – Minor inconsistencies 1 – Major deviation or unclear notation
Clarity and Presentation	Diagram is labeled, readable, and visually well-organized.	3	3 – Clear and professional 2 – Slightly cluttered 1 – Difficult to interpret

6. Presentation and Q&A		**10**	
Presentation Quality	Organized delivery with logical flow and explanation.	5	5 – Clear and engaging 4 – Well-structured 3 – Average 2 – Poorly coordinated 1 – Confusing
Q&A Response	Explains design rationale and answers questions effectively.	5	5 – Confident and accurate4 – Mostly correct3 – Hesitant but fair2 – Unclear reasoning1 – Unable to explain
6. Peer and Self Evaluation		**30**	
	TOTAL	**130**	

Summary:

Program Design and Architecture	25
Relationships	10
Functionality, Logic, and Error Handling	30
Code Quality and Readability	15
Class Diagram	10
Presentation and Q&A	10
Peer and Self Evaluation	30
Total	130