San José State University
College of Science, Department of Computer Science
CS 151 Object-Oriented Design, Section 6, Fall 2018
Dr. Angus Yeung

SELF-EVALUATOIN FORM FOR FINAL PROJECT

PART A.

Final Project (25 points) consists of two parts: project write-up (60%) and code evaluation (40%)

Final Project Write-up (Slide Deck) – 60% (15 out of 25 points)

Requirement	Score	Your Expected Score	Check by Grader
1. **Problem Statement** (3) a) Present clearly the pain points of the problem your team is trying to address. (1) b) Survey of competitive or comparable solutions in real world. (1) c) List in what aspects your team's solution is better than other solutions. (1)	3	3	
2. **Proof of Concept** (3) a) Describe how your team perform rapid prototyping and try out the ideas quickly. (1) b) List any existing sample code you found useful. (1) c) Describe what main idea / feature you are testing out in prototyping. (1)	3	3	
3. **Use Cases** (4) a) List at least 4 Use Cases for your application. (2) b) Note down all possible variations for each use case. (2)	4	4	
4. **CRC Modeling** (4) a) Model at least 4 main classes in your application. (2) b) Add main methods, linkages and dependencies to your main classes. (2)	4	4	
5. **UML Modeling** (8) a) Draw UML Class diagram for the classes in 4a and show clearly the relationship among the classes. (4) b) Draw a sequence diagram for the dynamics of one scenario in your application. (2) c) Draw a state diagram to show objects with interesting states. (2)	8	6	
6. **Good Design Practice** (10) a) Generate professional JavaDocs to describe the usages of fields and methods for those classes in 4a. (2) b) Choose any class that follows the 5C criteria for good class interface design: Cohesion, Completeness, Convenience, Clarity and Consistency. Explain clearly how this class conforms to the criteria. (4) c) Write unit tests for the classes described in 4a. (4)	10	6	

7. **Object Oriented Design** (6) a) Show two classes in your application that are considered as well encapsulated. (2) b) Show a design that closely follows the principle of loose coupling. (2) c) Demonstrate a design in your application that exhibits polymorphism. (2)	6	6	
8. **Design Patterns** (9) Use UML class diagrams to show: a) Template Method Pattern or any one of the behavioral design patterns in your program. (3) b) Decorator Pattern or any one of the structural design patterns in your program. (3) c) Factory Method or any one of the creational design patterns in your program. (3)	9	9	
9. **Java Object Model** (7) a) Override the `equals()` method of one of your classes. Your method shall meet the requirements for `equals` method: reflexive, symmetric, transitive, and properly handling of `null`. (2) b) Add serialization feature to a class and demonstrate the saving of object instances into a file stream and loading from the file stream into object instances. (3) c) Invoke a method using Java Reflection classes instead of normal ways. (2)	7	4	
10. **Concurrency** (6) a) Show at least one running thread in your program. (2) b) Demonstrate your understanding of thread synchronization by implementing one producer thread and one consumer thread. (4)	6	6	