**OOSE/OOSD Project Marking Scheme (40%)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Items** | **Criteria** | **Description** | **Marks** | **Total** |
| **DIAGRAM/REPORT** | | | | |
| **Class Diagram** | Class | Minimum 5 classes, correctly labeled | 3 | 10 |
| Relationships | Includes Inheritance, Association, Aggregation or Composition | 2 |
| Attributes | Minimum 3 attributes per class | 2 |
| Methods | Minimum 3 methods per class | 2 |
| OO Concepts | Abstraction, Encapsulation, Polymorphism | 1 |
| **User Interface Design** | OO Design Rules | Modularity and Cohesion - Each UI module is modular and cohesive, focusing exclusively on a single area of functionality without overlapping tasks. | 2 | 5 |
| Low Coupling - Each UI module operates independently, with clear boundaries and minimal dependencies between the functionalities of each class | 2 |
| Clear and uniform syntax and terminology used across all commands, consistently aligned with class diagram terms and following a standardized command format. | 1 |
| **Test Cases** | Completeness | Min. 5 test cases with Action, Input, Expected Output | 4 | 4 |
| **Report** | Organization | Clear structure, minimal grammar errors | 2 | 3 |
| Formatting | Professional presentation, adherence to guidelines | 1 |
| **PROJECT IMPLEMENTATION** | | | | |
| **Object-Oriented Programming** | Class Diagram Implementation | Reflects class diagram accurately, includes min. 5 objects | 5 | 18 |
| Encapsulation & Abstraction | Classes and modules should follow encapsulation rules. Use of header and implementation files (if applicable) to separate concerns. | 4 |
| Inheritance | Implements at least 1 superclass and 1 subclass. | 3 |
| Polymorphism | Demonstrates polymorphism with at least 3 overloaded methods and 2 overridden methods. | 3 |
| Functionality | Supports core data operations, including add, modify, and delete functionalities. | 3 |
| **PEER REVIEW AND PRESENTATION** | | | | |
| **Individual Contribution** | Documented Contribution | Clearly attributed work in report and code   * Contribute in class diagram, minimum 1 test case, minimum 1 user interface design and implement mininum 1 class * State clearly the group member’s name and matric number for each contribution. For example:   In the report:  User Interface for Class Admin Created : Samantha James (yyyyy)  In the coding  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Class Admin  Implemented by Ali bin Ahmad (xxxxx)  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | 2 | 2 |
| Group Presentation | System Understanding | Demonstrates understanding of system design | 2 | 3 |
| Q&A Performance | Clear, confident responses to questions | 1 |
| Peer Evaluation | Peer Feedback on Participation | Engagement and effort rated by peers | 2 | 5 |
| Contribution Quality | Value of work contributed, as rated by peers | 2 |
| Collaboration and Communication | Supportiveness and responsiveness in group | 1 |
| **Overall Total** |  | | | 50 |
| **BONUS MARKS** | | | | |
| **Bonus Marks** | Project completed in Python | Extra reward for Python implementation | 5 | +5 |

# Project Marking Rubric

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Criteria** | **Item** | **Excellent (4) – Full Marks** | **Good (3) – 75 % of full marks** | **Satisfactory - 50 % of full marks (2)** | **Needs Improvement (1) - 25 % of full marks** | **Full Marks** |
| **Diagram/Report** | Class Diagram | 5+ well-labeled classes with clear structure and relevance. | 5+ classes mostly well-labeled; minor errors in structure. | Meets minimum requirement of 5 classes; some labels are unclear or lack relevance. | Fewer than 5 classes or unclear labeling and structure. | 3 |
| Relationships | Correctly includes all relevant relationships (Inheritance, Association,  Aggregation, Composition) | Includes most relevant relationships; minor inaccuracies. | Relationships are present but limited in scope or accuracy. | Lacks clear or correct relationships; minimal effort shown. | 2 |
| Attributes | 3+ meaningful attributes per class with consistent naming. | 3 attributes per class; mostly meaningful and consistent. | Minimum attributes met; some may lack clarity or relevance. | Attributes are unclear, inconsistent, or do not meet the minimum requirement. | 2 |
| Methods | 3+ methods per class that are well-defined and relevant to class functionality. | Methods are relevant and mostly well-defined. | Meets minimum requirement with some methods lacking relevance or clarity. | Methods are minimal, unclear, or irrelevant to class functionality. | 2 |
| OO Concepts | Effectively applies Abstraction, Encapsulation, and Polymorphism in the design. | Mostly effective use of OO concepts; minor issues in application. | Some application of OO concepts but lacks depth or clarity. | OO concepts are minimally or poorly applied, with significant errors. | 1 |
| **User Interface Design** | OO Design Rules | Highly modular and cohesive UI design, with focused functionality per module. | Generally modular UI with minor overlaps in functionality. | Modular design with some inconsistencies in cohesion or focus. | UI lacks modularity or cohesion, with significant overlap or lack of focus. | 2 |
| Low Coupling | Modules operate independently, with minimal dependencies, maintaining clear boundaries. | Mostly independent modules with minor dependencies. | Some independence in modules, but notable dependencies exist. | High interdependency among modules; unclear boundaries. | 2 |
| Syntax & Terminology | Clear, uniform terminology aligned with the class diagram; follows a consistent command format. | Mostly uniform terminology; minor inconsistencies in syntax. | Inconsistent use of terminology, but some alignment with the class diagram. | Terminology is unclear or inconsistent; lacks alignment with class diagram. | 1 |
| **Test Cases** | Completeness | 5+ comprehensive test cases covering action, input, and expected output, with all cases relevant to functionality. | 5 test cases mostly relevant and cover expected actions/outputs. | Meets minimum with limited detail or relevance in test cases. | Fewer than 5 test cases, or test cases are unclear and lack coverage. | 4 |
| **Report** | Organization | Well-structured report with minimal grammar errors, easy to follow. | Clear structure; minor grammar issues, mostly easy to follow. | Organized with notable grammar or structural issues but understandable. | Poorly organized, difficult to follow, with many grammar issues. | 2 |
| Formatting | Professional presentation with adherence to guidelines and clear formatting. | Mostly professional; minor formatting inconsistencies. | Presentation has some formatting errors, but mostly follows guidelines. | Unprofessional presentation with major formatting issues or guideline inconsistencies. | 1 |
| **Project Implementation** | Class Diagram Implementation | Accurately reflects the class diagram with at least 5 instantiated objects, functioning as intended. | Mostly accurate implementation with minor deviations from the class diagram. | Limited reflection of class diagram with inconsistencies or errors in instantiation. | Does not accurately reflect the class diagram, with few or incorrect objects. | 5 |
| Encapsulation & Abstraction | Strong use of encapsulation with well-separated headers/implementations (if applicable) and clear abstraction. | Mostly follows encapsulation rules; minor issues in separation or abstraction. | Encapsulation applied with limited separation; abstraction may lack clarity. | Minimal or incorrect use of encapsulation; poor abstraction with no separation. | 4 |
| Inheritance | Clear implementation of at least one superclass and one subclass, demonstrating correct inheritance. | Includes superclass and subclass with minor inheritance issues. | Implements inheritance, but lacks clarity or correct application. | Incorrect or minimal implementation of inheritance with significant issues. | 3 |
| Polymorphism | Demonstrates polymorphism effectively with 3+ overloaded and 2+ overridden methods. | Mostly effective use of polymorphism; minor issues in methods. | Limited or partially correct polymorphism application; meets minimum. | Minimal or incorrect use of polymorphism with errors in methods. | 3 |
| Functionality | Fully supports core operations (add, modify, delete) with reliable performance. | Supports core operations with minor functionality issues. | Limited core functionality; some operations may not work as expected. | Missing or incorrect core functionalities; poor implementation of operations. | 3 |
| **Peer Review & Presentation** | Individual Contribution | Clearly documented contributions in report and code, with attribution to group members. | Documented contributions with minor attribution issues. | Limited contribution documentation or unclear attribution. | Contributions unclear or missing, with minimal documentation. | 2 |
| Group Presentation | Shows strong understanding of system design, clearly explained during presentation. | Understands system design with minor explanation issues. | Basic understanding demonstrated; explanations may lack depth or clarity. | Limited understanding of system design, with unclear explanations. | 2 |
| Q&A Performance | Responds to questions clearly and confidently, with accurate answers. | Mostly clear and confident responses, with minor inaccuracies. | Basic responses to questions; some answers may be unclear or incorrect. | Struggles to answer questions, with unclear or incorrect responses. | 1 |
| Peer Evaluation | Positive feedback from peers on engagement, effort, quality of contribution, and collaboration. | Mostly positive peer feedback; minor issues in engagement or contribution quality. | Average peer feedback, with some noted issues in engagement or contribution. | Poor peer feedback on engagement, contribution, or collaboration. | 5 |

**Marking Reference Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Full Mark** | **Excellent (4)** | **Good (3)** | **Satisfactory (2)** | **Needs Improvement (1)** |
| 1 | 1.0 | 0.75 | 0.5 | 0.25 |
| 2 | 2.0 | 1.5 | 1.0 | 0.5 |
| 3 | 3.0 | 2.25 | 1.5 | 0.75 |
| 4 | 4.0 | 3.0 | 2.0 | 1.0 |
| 5 | 5.0 | 3.75 | 2.5 | 1.25 |