**Project Marking Scheme (40%)**

|  |  |  |
| --- | --- | --- |
| **Report (15%)** | * Class diagram   + Min. 4 classes | = 10 |
|  | * + Associations, multiplicity |  |
|  | * + Attributes (Min. 3 attributes per class) |  |
|  | * + Methods (Min. 3 methods per class) |  |
|  | * + Polymorphism     - Overriding     - Overloading |  |
|  | * + Inheritance and/or Aggregation |  |
|  | * + Others OO concepts: abstraction, encapsulation |  |
|  | * User interface Design (OO rules) | = 5 |
|  | * Test Case (Min. 5 test cases) | = 5 |
| **Object Oriented Programming (15%)** | * Reflect class diagram * Implement min 4 classes and 4 objects | =2 |
|  | * Encapsulation – separate classes and objects * Implement visibility settings | =2 |
|  | * Abstraction * Min 3 Header files and implementation files separated | =3 |
|  |
| * Inheritance * Implement min 1 super class and 1 sub class | **=2** |
|  |
| * Polymorphism * Min 3 methods overloading * Min 2 methods overriding | =3 |
|  |
|  | * Functional * Add, modify, delete data | =2 |
|  | * Main Driver (exe) * From one user interface to another | =1 |
| **Group Presentation (5%)** | * Understanding of the System Design and Prototype | = 3 |
|  | * Object-oriented concepts projected on the system | = 2 |
| **Individual Contribution (3%)** | * Contribute in class diagram, test case, user interface design and implementation | =2 |
|  | * Question and Answer | =1 |
| **Group Work (2%)** | * Active participation in meeting * Commitment and Effort to attend meeting * Supporting other members * Contribution to overall work activity | =2 |

**UML Diagrams Rubric (Assignment and Project)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Criteria** | **Excellent** | **Good** | **Average** | **Poor** |
| **1** | **Class Diagram**  **– Classes** | Minimum 4 classes are well  identified to the case study and  demonstrates  understanding of why each class falls into the category of Boundary, Entity and Controller. UML notation for class diagram is correct. | Minimum 3 classes are well  identified and  divided into Boundary, Entity  and Controller. UML notation for class diagram is correct. | Classes are identified but  not identified  properly to Boundary, Entity and Controller. | No indication of the various classes in the  class diagram. |
|  | **Class Diagram**  **– Relationship, Attributes and Methods** | All classes are  mentioned in the  diagram with their methods (return type and parameters), attributes (type and access level)  and relationships (Labelled with multiplicity) are  well presented. Correct UML notations are used. | All classes are  mentioned in the  diagram with  their methods,  attributes and  relationships but the layout and presentation is a bit confusing. | Some classes are in the diagram and a lot of missing methods,  attributes and  relationships missing. | Class diagram shows no relationship  between the various classes. |
| **2** | **User Interface (UI) Design** | All of the user interfaces reflects the class diagram by applying OO Design Approach/Rules. | All of the user interfaces reflects the class diagram with some not applying OO Design Approach/Rules. | Some of the user interfaces reflects the class diagram and some of the UI do not apply OO Design Approach/Rules. | User interfaces are not reflective of the class diagram. |
| **3** | **Test Cases** | Test cases with clearly stated Action, Input Values, Expected Output and Actual Results for at least 2 classes or 3 main methods | Test cases with missing steps or values for Action, Input Values, Expected Output and Actual Results for less than 2 classes or 3 main methods | Test cases missing either Action, Input Values, Expected Output or Actual Results | Test cases missing either Action, Input Values, Expected Output or Actual Results and missing values. |
| **4** | **Report Mechanics** | Report has an excellent organization and presents material in a very logical sequence. No spelling or grammar errors, excellent formatting, highly readable.  Functionality – working codes with usable user interface, get and set methods | The report can be followed easily and all material is present. Spelling and grammar errors rare, excellent formatting. | There are significant issues with coherence and ordering. Significant material is missing. Few pages without mechanics’ errors making reading a chore. Readability is a significant problem. | The report has no apparent organization or logical order. Mechanics errors make it a struggle to decipher meaning. |
| **5** | **OO Program** | Fully OO program is implemented by reflecting the class diagram (classes implemented in the program) applying OO concepts. The OO program must be able to add, delete and modify data. The user interfaces are well organised and functional: able transition from one user interface to another. | Fully OO program is implemented by reflecting the class diagram (classes implemented in the program) with some parts of the program applying OO concepts.  The program can perform two basic data tasks: add, delete and modify.  User interfaces are well organised. | Some of the classes are implemented as OO programming while others used structural programming.  Able to perform two basic data tasks: add, delete and modify.  User interfaces are clear with some user interfaces not being able to transition. | Structural programming is used instead of OO programming.  Only perform one data tasks: add, delete and modify. User interfaces are not well organised. |