**Project Title**

*by*

*Proponents*

1. **Project Title and Team Members**

* Write the name and description of your system (e.g., "CALLUSsugan: Revolutionizing Patient Record Management for Efficiency and Care.")
* List all team members with their roles in the project (e.g., leader, researcher, writer).

1. **Purpose of the System**

* Align the purpose with the course objectives: Highlight how the system demonstrates understanding o f software development models, technical aspects, and design patterns.
* Add a sentence about its relevance to real-world applications. Example: "This system is designed to support healthcare providers in optimizing patient record workflows, ensuring high-quality care and data management.

1. **Features of the System**

* Describe the main functions of your system in plain language. Example:
  + Add new patient records.
  + Search for an existing patient’s information.
  + Edit or update patient details.
* Categorize features into functional and non-functional requirements for clarity.
* Add a brief explanation of how the features contribute to the system's purpose.

1. **How It Works**

* Use simple steps to explain how someone can use the system. Example:
  + Step 1: Open the system.
  + Step 2: Enter patient data into the form.
  + Step 3: Save the changes to update the record.
* Highlight any innovative or user-friendly elements of the workflow

1. **Roles and Contributions**

|  |  |  |
| --- | --- | --- |
| **Name** | **Role** | **Contribution** |
| Deo | Database Manager | Designed and created the database |

1. **Screenshots or Diagrams**

* Include simple images of your work (e.g., a screenshot of a form or table).
  + Label the images to show what part of the system they represent.

1. **Class Diagram**

A class diagram is a type of visual representation used in object-oriented design to model the structure of a system. It shows the static relationships between classes within the system and includes details like:

* 1. Represented as rectangles divided into sections for the class name, attributes (data), and methods (functions).
  2. Relationships: Demonstrates how classes interact, such as inheritance, associations, aggregations, and compositions.
  3. Encapsulation: Indicates the visibility of attributes and methods using symbols like + (public), - (private), and # (protected).  
       
     *(see appendix A. on how to create a class diagram)*

1. **Challenges and Solutions**

* Briefly mention problems encountered and how they were solved. Example:
  + Challenge: Difficulty creating a form.
  + Solution: Followed online tutorials and practiced together.

1. **Conclusion**

* Summarize the outcome of your project.  
  Example: "We successfully created a simple system that meets the goal of helping manage patient records more easily."

1. **Recommendations for Improvement**

* Suggest ways the system could be enhanced. Example:
  + Add a print option for patient records.
  + Include graphs or charts to visualize data.

**Grading Rubrics**

|  |  |  |
| --- | --- | --- |
| Criteria | Description | Points |
| Project Documentation | Clear and complete project documentation, including purpose, features, challenges, and solutions. | **20** |
| Implementation of OOP Concepts | Demonstrates mastery of OOP principles (e.g., encapsulation, inheritance, polymorphism). | **30** |
| Class Diagram | Accurate and well-designed class diagram showcasing classes, relationships, and visibility. | **20** |
| System Functionality | System is fully functional, meeting described features and requirements. | **20** |
| Creativity and Innovation | Innovative solutions or features enhancing user experience and system relevance. | **10** |
| Team Collaboration | Effective teamwork and fair contributions from all team members. | **10** |
| Total |  | **110** |

**Scoring Guide:**

* **Project Documentation (20 Points)**: Ensure all sections of the documentation are addressed, including the relevance to course objectives and real-world applications.
* **Implementation of OOP Concepts (30 Points)**: Evaluate how effectively OOP principles are applied in the system design and implementation.
* **Class Diagram (20 Points)**: Assess clarity, accuracy, and alignment of the diagram with system requirements.
* **System Functionality (20 Points)**: Test if all described features are implemented and working correctly.
* **Creativity and Innovation (10 Points)**: Look for unique approaches or enhancements to standard functionality.
* **Team Collaboration (10 Points)**: Verify contributions from all team members, as outlined in the roles and contributions section.

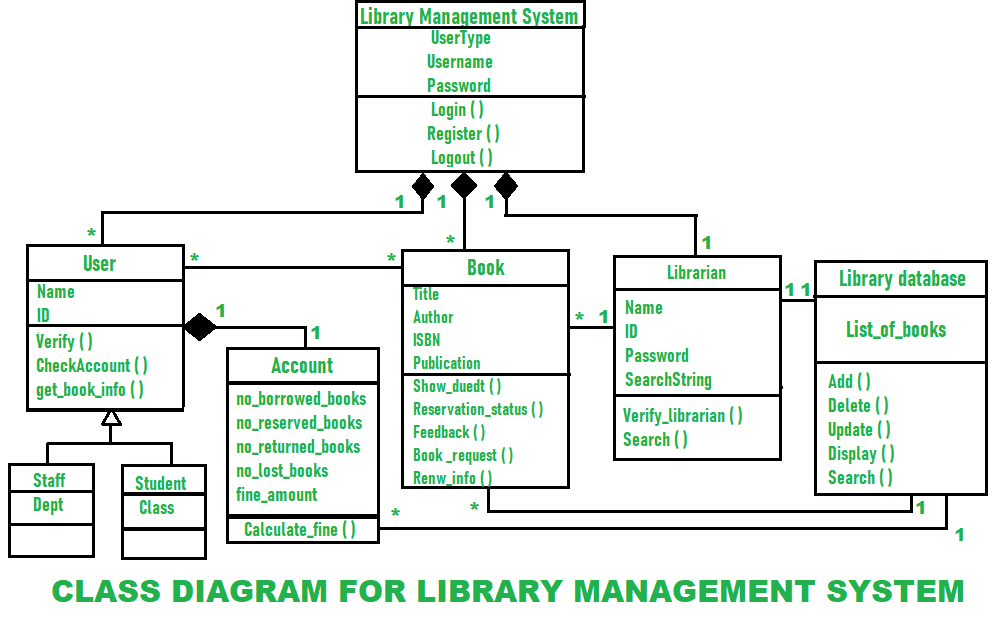
**Appendix A. Creating a class diagram**

These diagrams are a cornerstone in object-oriented design and help visualize the static structure of a system, making them an excellent tool for planning and communication.

**Steps to Create a Class Diagram:**

1. **Identify Classes**:
   * Start by listing all the main entities (real-world objects or abstract concepts) in your system. For example, in a library management system, you might have classes like Book, Member, and Librarian.
2. **Define Attributes and Methods**:
   * For each class, determine what data it will store (attributes) and what functions it will perform (methods).
   * Example for Book:
     + **Attributes**: title, author, ISBN, publicationYear.
     + **Methods**: borrow(), return(), reserve().
3. **Establish Relationships**:
   * Show how classes are related:
     + **Inheritance**: Use a line with a hollow arrow to represent a subclass-parent relationship. For instance, Fiction and NonFiction could inherit from Book.
     + **Association**: Show how objects of one class interact with objects of another (e.g., Member and Book are associated because members borrow books).
     + **Aggregation/Composition**: Use a diamond shape to indicate whole-part relationships, like a Library containing Books.
4. **Set Visibility Modifiers**:
   * Represent encapsulation with visibility markers:
     + + for public, - for private, # for protected.
5. **Refine and Iterate**:
   * Review your diagram for clarity, add notes if necessary, and ensure that it aligns with the system's requirements.

**Example Class Diagram of a Library System:**



Reference: <https://www.geeksforgeeks.org/class-diagram-for-library-management-system/>