PRG261 PROJECT 1

Paledi Kutu
577860 | LECTURE:JOHN DARATOS

Introduction:

The purpose of this report is to present the software system design for the student management system at the Belgium campus satellite campus in Stellenbosch. The system aims to register and display all students' information on demand and provide promotions based on specific criteria. The design process involves identifying functional and non-functional requirements, decomposing them into smaller modules, and defining the architecture, components, interfaces, and data for the system.

Functional Requirements:

1. Student Registration:
    1.1. Capture student information: ID, Name, Surname, Gender, Email, DOB, Phone number, Qualification.
    1.2. Calculate and apply discounts for eligible students based on age and gender.
    1.3. Store student details in the database.
2. Display Registered Students:
    2.1. Retrieve and display all registered students' details.
    2.2. Show a notification if a student qualifies for a promotion.

Non-Functional Requirements:

1. Maintain high levels of availability during peak periods.
2. User Interface:
    2.1. The system should have a user-friendly interface for easy interaction with the application.
    2.2. Clear and concise notifications should be displayed to the user.
3. Security:
    3.1. Implement appropriate security measures to protect student data.
    3.2. Ensure data integrity and prevent unauthorized access to sensitive information.
4. Facilitate easy update processes without interrupting service.
5. Achieve fast response times and low latency for critical tasks.
6. Implement backup and disaster recovery procedures for business continuity.
7. Ensure compatibility across multiple browsers and operating systems.

Object-Oriented Design:

1) Key Classes and Objects:
   a) Student:
      i) Attributes: ID, Name, Surname, Gender, Email, DOB, Phone number, Qualification, Total Fees.
      ii) Methods: getDiscount(), calculateFees(), displayStudentInfo().
      iii)
   b) Calculate Fee
      i) This class represents a fee for a specific academic program, course, or service. It contains information such as the fee amount, and if the is a discount or not.

Justification for OOP Principles and Class Members:

- Encapsulation:
  - Each class encapsulates its attributes and methods, providing clear boundaries and ensuring data integrity. For example, the student class encapsulates student-related attributes and methods, allowing for easy management and manipulation of student data.
- Inheritance:
  - While not explicitly mentioned in the requirements, inheritance can be employed to establish a hierarchical relationship between classes. For instance, if additional roles or user types are introduced, a user class can serve as a base class for different user types like Student and Administrator, inheriting common attributes and behaviours.
- Polymorphism:
  - Polymorphism allows flexibility in method implementation. For example, the student class can have a generic display Info() method, which can be overridden in derived classes to cater to specific information display requirements.
- Class Members(Methods and Attributes):
  - The chosen class members are based on the requirements and design goals. Methods are responsible for performing actions and manipulating data, while attributes store and represent the state of objects. They are designed to provide the required functionality and support system interactions effectively.

- Comments:
    - Comments should be added throughout the codebase to improve code readability and provide explanations where necessary. They aid in understanding the purpose of classes, methods, and important code segments.