Python OOP Group Project Assignment

Objective: To design and implement a Python application demonstrating object-oriented programming (OOP) principles.

Group Size: 5 students per group

Project Selection:

Choose ONE of the following projects, or propose a similar one of comparable complexity.

You can see a partial example at the end of this document

1. **Online Shopping Cart System**
   - Classes: Product, Customer, Cart, Order, Payment
   - Features: Product catalog, checkout process, payment processing.
   - OOP Concepts: Encapsulation, Abstraction.
2. **Student Management System**
   - Classes: Student, Teacher, Course, Enrollment, Grade
   - Features: Student registration, course enrollment, grade management.
   - OOP Concepts: Encapsulation, Abstraction, Inheritance.
3. **Movie Ticket Booking System**
   - Classes: Movie, Theater, ShowTime, Ticket, Customer
   - Features: Booking movie tickets, selecting showtimes, seat selection.
   - OOP Concepts: Encapsulation, Polymorphism.
4. **Vehicle Rental System**
   - Classes: Vehicle, Customer, Rental, Payment
   - Features: Vehicle rental, pricing, availability, customer information.
   - OOP Concepts: Encapsulation, Inheritance.
5. **Hospital Management System**
   - Classes: Patient, Doctor, Appointment, Room, Bill
   - Features: Patient records, appointment scheduling, billing.
   - OOP Concepts: Encapsulation, Abstraction.
6. **Library Management System**
   - Classes: Book, Member, Loan, Librarian
   - Features: Book borrowing/returning, overdue notices, member management.
   - OOP Concepts: Encapsulation, Inheritance, Polymorphism.
7. **Hotel Reservation System**

- Classes: Hotel, Room, Reservation, Guest, Payment
- Features: Room availability, reservation management, guest check-in/out.
- OOP Concepts: Encapsulation, Inheritance, Polymorphism.

8. **Inventory Management System**
    - Classes: Inventory, Product, Supplier, Order, Warehouse
    - Features: Stock tracking, reordering products, supplier management.
    - OOP Concepts: Encapsulation, Abstraction.
9. **Bank Loan Processing System**
    - Classes: Loan, Customer, Bank, InterestRate, Payment
    - Features: Loan application, approval, repayment schedule.
    - OOP Concepts: Inheritance, Polymorphism.
10. **Food Delivery System**
    - Classes: Restaurant, Customer, Order, DeliveryPerson, Payment
    - Features: Menu display, order placement, delivery tracking.
    - OOP Concepts: Inheritance, Encapsulation, Polymorphism.
11. **Smart Home Automation System**
    - Classes: Home, Device, Light, Thermostat, SecuritySystem
    - Features: Device control, scheduling, security monitoring.
    - OOP Concepts: Encapsulation, Polymorphism.
12. **Expense Tracker Application**
    - Classes: Expense, Category, User, Report
    - Features: Expense categorization, tracking, reporting.
    - OOP Concepts: Inheritance, Abstraction, Encapsulation.
13. **E-learning Platform**
    - Classes: Course, Student, Instructor, Lesson, Quiz
    - Features: Course enrollment, lesson management, grading.
    - OOP Concepts: Inheritance, Encapsulation, Abstraction.
14. **Fitness Tracking Application**
    - Classes: Workout, User, Goal, Progress, Coach
    - Features: Workout tracking, goal setting, progress measurement.
    - OOP Concepts: Encapsulation, Abstraction.
15. **E-voting System**
    - Classes: Voter, Candidate, Election, Vote
    - Features: Voter registration, voting, result calculation.
    - OOP Concepts: Inheritance, Encapsulation.
16. **Pet Adoption System**

- Classes: Animal, Adoption, Shelter, Customer
- Features: Animal catalog, adoption process, payment tracking.
- OOP Concepts: Inheritance, Polymorphism.

17. **Parking Management System**
    - Classes: ParkingLot, Vehicle, Ticket, Payment, ParkingSpot
    - Features: Spot assignment, vehicle tracking, payment processing.
    - OOP Concepts: Polymorphism, Encapsulation.
18. **Chat Messaging Application**
    - Classes: Message, User, ChatRoom, Attachment
    - Features: Sending/receiving messages, file attachments.
    - OOP Concepts: Inheritance, Encapsulation.
19. **Bus Reservation System**
    - Classes: Bus, Route, Ticket, Passenger, Payment
    - Features: Bus schedule, seat reservation, payment processing.
    - OOP Concepts: Polymorphism, Inheritance.
20. **Flight Booking System**
    - Classes: Flight, Passenger, Ticket, Airport, Payment
    - Features: Flight booking, seat selection, ticket purchasing.
    - OOP Concepts: Inheritance, Encapsulation.
21. **Restaurant Management System**
    - Classes: Restaurant, Table, Order, Menu, Payment
    - Features: Table reservation, order processing, billing.
    - OOP Concepts: Polymorphism, Inheritance.
22. **Weather Monitoring Application**
    - Classes: WeatherStation, Sensor, Report, Alert
    - Features: Weather data collection, analysis, alerts.
    - OOP Concepts: Inheritance, Abstraction.
23. **Real Estate Management System**
    - Classes: Property, Agent, Customer, Sale, Lease
    - Features: Property listing, sale/lease management, customer details.
    - OOP Concepts: Inheritance, Encapsulation.
24. **Online Examination System**
    - Classes: Exam, Question, Student, Result
    - Features: Exam creation, grading, result generation.
    - OOP Concepts: Polymorphism, Encapsulation.
25. **Ticket Reservation System**

- Classes: Event, Ticket, Customer, Seat, Payment
- Features: Event listings, seat reservation, payment.
- OOP Concepts: Inheritance, Encapsulation.

26. **Home Budgeting System**
    - Classes: Budget, Income, Expense, Category, Report
    - Features: Income/expense tracking, reporting.
    - OOP Concepts: Encapsulation, Abstraction.
27. **Social Media Platform**
    - Classes: User, Post, Comment, Like, Follow
    - Features: Posting content, commenting, following users.
    - OOP Concepts: Polymorphism, Encapsulation.
28. **Job Recruitment System**
    - Classes: Job, Employer, Candidate, Application, Interview
    - Features: Job posting, applications, interview scheduling.
    - OOP Concepts: Encapsulation, Polymorphism.
    -
29. **Online Auction System**
    - Classes: Auction, Bid, User, Product
    - Features: Auction creation, bidding process, winner announcements.
    - OOP Concepts: Inheritance, Encapsulation.

Group Responsibilities:

- Each group member must understand the full implementation of each section and be able to explain it.
- **Random Assignment:** The specific area each member presents about will be randomly assigned during the presentation.
- **Collaboration:** You will likely, each have a primary area of focus, but need to be ready to explain each section.

Deliverables:

- **Python Code:** Well-structured, documented, and functional Python code implementing the chosen project.
- **Presentation:** A short presentation demonstrating the application and explaining the OOP concepts used.
- **Report:** A brief report describing the project, design choices, challenges encountered, and lessons learned.

Grading:

- **Functionality:** Does the application work as intended?
- **OOP Principles:** Are OOP concepts (encapsulation, abstraction, inheritance, polymorphism) correctly and effectively applied?
- **Code Quality:** Is the code well-structured, readable, and documented?
- **Presentation:** Is the presentation clear, informative, and engaging?

Due Date: To be announced.

Important Notes:

- **Start Early:** Don't wait for the due date to start working on the project. Plan your time and break down the tasks.
- **Communication:** Communicate effectively within your group. Discuss design decisions, share progress, and help each other.

Partial Example:

Project: ATM Simulation System

ATM Simulation System

- Classes: BankAccount, Customer, ATM, Transaction
- Features: Withdrawal, deposit, transfer operations, different account types.
- OOP Concepts: Encapsulation, Abstraction, Inheritance, Polymorphism.

Classes and Objects:

- **BankAccount**, **Customer**, **ATM**, **Transaction** classes.
- Attributes for **BankAccount**: account_number, balance, account_type.
- Objects represent individual bank accounts and customers.

Encapsulation:

- Private attributes for **balance** in BankAccount.
- Getter and setter methods to check balance, withdraw, and deposit money.

Abstraction:

- Abstract the ATM operations (withdrawal, deposit, transfer) without exposing the internal workings of the bank database or transaction history to the user.

Inheritance:

- Subclasses **SavingsAccount** and **CurrentAccount** inherit from **BankAccount**.
- **SavingsAccount** may have interest-earning features, while **CurrentAccount** could have an overdraft option.

Polymorphism:

- Different behaviors for withdrawing money in **SavingsAccount** and **CurrentAccount** (e.g., withdrawal limit, overdraft).