Department of Computer Engineering

Academic Term: First Term 2023-24

Practical No:	**7**
Title:	**Design Using Object-Oriented Approach with Emphasis on Cohesion and Coupling in Software Engineering**
Date	**7/9/23**
Roll No:	**9628**
Team members:	

Rubrics for Evaluation:

Sr. No	Performance Indicator	Excellent	Good	Below Average	Total Score
1	On time Completion & Submission (01)	01 (On Time)	NA	00 (Not on Time)	
2	Theory Understanding(02)	02(Correct)	NA	01 (Tried)	
3	Content Quality (03)	03(All used)	02 (Partial)	01 (rarely followed)	
4	Post Lab Questions (04)	04(done well)	3 (Partially Correct)	2(submitted)	

Signature of the Teacher:

Department of Computer Engineering

Academic Term: First Term 2022-23

Signature of the Teacher:

Lab Experiment 07

Experiment Name: Design Using Object-Oriented Approach with Emphasis on Cohesion and Coupling in Software Engineering

Objective: The objective of this lab experiment is to introduce students to the Object-Oriented (OO) approach in software design, focusing on the principles of cohesion and coupling. Students will gain practical experience in designing a sample software project using OO principles to achieve high cohesion and low coupling, promoting maintainable and flexible software.

Introduction: The Object-Oriented approach is a powerful paradigm in software design, emphasizing the organization of code into objects, classes, and interactions. Cohesion and Coupling are essential design principles that guide the creation of well-structured and modular software.

Lab Experiment Overview:

1. Introduction to Object-Oriented Design: The lab session begins with an introduction to the Object-Oriented approach, explaining the concepts of classes, objects, inheritance, polymorphism, and encapsulation.
2. Defining the Sample Project: Students are provided with a sample software project that requires design and implementation. The project may involve multiple modules or functionalities.
3. Cohesion in Design: Students learn about Cohesion, the degree to which elements within a module or class belong together. They understand the different types of cohesion, such as functional, sequential, communicational, and temporal, and how to achieve high cohesion in their design.
4. Coupling in Design: Students explore Coupling, the degree of interdependence between modules or classes. They understand the types of coupling, such as content, common, control, and stamp coupling, and strive for low coupling in their design.
5. Applying OO Principles: Using the Object-Oriented approach, students design classes and identify their attributes, methods, and interactions. They ensure that classes have high cohesion and are loosely coupled.
6. Class Diagrams: Students create Class Diagrams to visually represent their design, illustrating the relationships between classes and their attributes and methods.
7. Design Review: Students conduct a design review session, where they present their Class Diagrams and receive feedback from their peers.
8. Conclusion and Reflection: Students discuss the significance of Object-Oriented Design principles, Cohesion, and Coupling in creating maintainable and flexible software. They reflect on their experience in applying these principles during the design process.

Learning Outcomes: By the end of this lab experiment, students are expected to:

- Understand the Object-Oriented approach and its core principles, such as encapsulation, inheritance, and polymorphism.
- Gain practical experience in designing software using OO principles with an emphasis on Cohesion and Coupling.

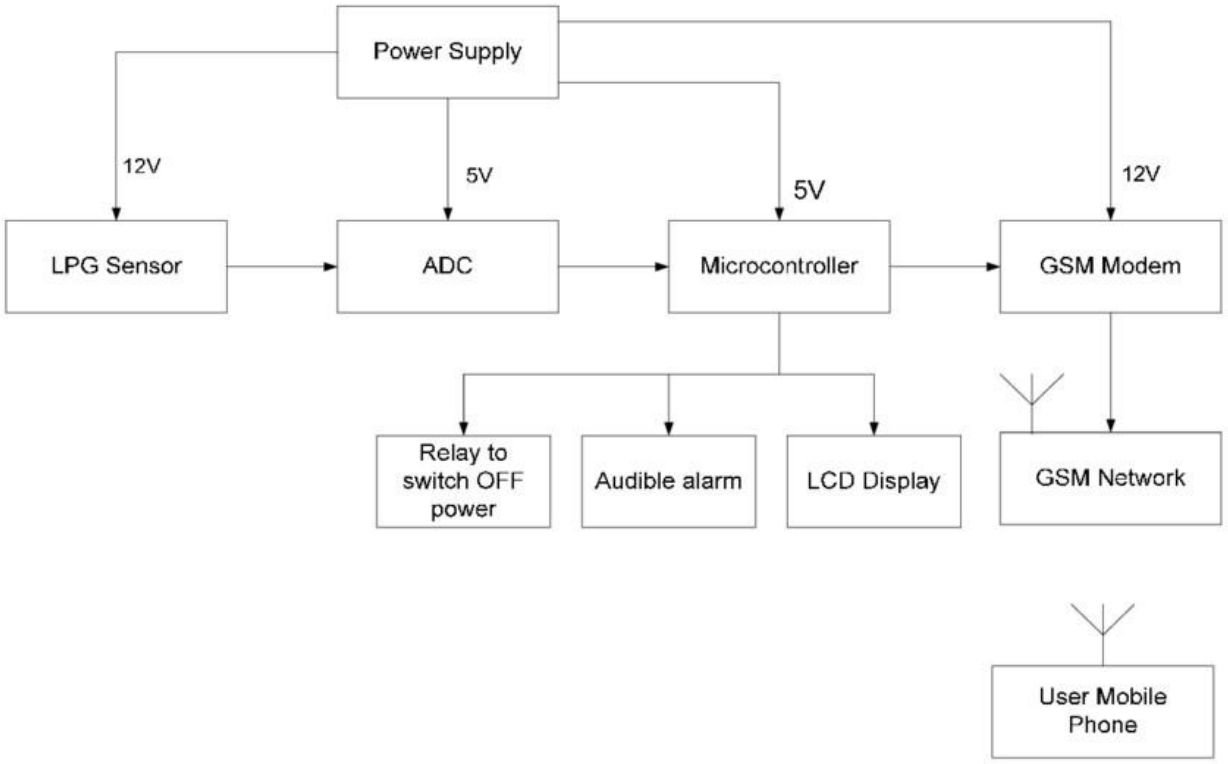
- Learn to identify and implement high cohesion and low coupling in their design, promoting modular and maintainable code.
- Develop skills in creating Class Diagrams to visualize the relationships between classes.
- Appreciate the importance of design principles in creating robust and adaptable software.

Pre-Lab Preparations: Before the lab session, students should review Object-Oriented concepts, such as classes, objects, inheritance, and polymorphism. They should also familiarize themselves with the principles of Cohesion and Coupling in software design.

Materials and Resources:

- Project brief and details for the sample software project
- Whiteboard or projector for creating Class Diagrams
- Drawing tools or software for visualizing the design

Conclusion: The lab experiment on designing software using the Object-Oriented approach with a focus on Cohesion and Coupling provides students with essential skills in creating well-structured and maintainable software. By applying OO principles and ensuring high cohesion and low coupling, students design flexible and reusable code, facilitating future changes and enhancements. The experience in creating Class Diagrams enhances their ability to visualize and communicate their design effectively. The lab experiment encourages students to adopt design best practices, promoting modular and efficient software development in their future projects. Emphasizing Cohesion and Coupling in the Object-Oriented approach empowers students to create high-quality software that meets user requirements and adapts to evolving needs with ease.



Architecture Diagram of Gas Leakage Detection System

Shortcomings:

Gas leakage detection systems are designed to detect the presence of potentially harmful or flammable gases in the environment. While these systems can be highly effective, they do have some shortcomings and limitations:

1. False Alarms: Gas detection systems can be prone to false alarms. Various factors, such as environmental conditions, sensor malfunctions, or interference from other substances, can trigger false alerts. This can lead to complacency if users become desensitized to frequent false alarms.

2. Detection Thresholds: Gas detectors may not always detect gas leaks at low concentrations, especially if the gas is dispersed or diluted quickly. Some gases, like natural gas, are odorless and colorless, making early detection challenging.

3. Limited Gas Types: Gas detection systems are typically designed to detect specific types of gases or vapors. A single system may not be suitable for detecting a wide range of gases, so different sensors or systems may be needed for various applications.

4. Maintenance Requirements: Gas detectors require regular maintenance and calibration to ensure accuracy and reliability. Neglecting maintenance can lead to sensor degradation or false readings. This adds to the overall cost of ownership.

5. Response Time: The response time of gas detectors varies, and some systems

may not react quickly enough to prevent hazardous situations in fast-spreading gas leak scenarios. Delayed responses can result in safety issues.

6. Environmental Interference: Environmental conditions such as high humidity, extreme temperatures, or the presence of other gases can affect the accuracy and reliability of gas detection systems.

7. Placement and Coverage: The effectiveness of a gas detection system depends on proper placement and coverage. Inadequate sensor placement or coverage gaps can lead to missed gas leaks.

8. Cost: High-quality gas detection systems can be expensive, especially when covering large areas or multiple gas types. This cost can be a deterrent for some organizations or homeowners.

9. Training Requirements: Users need training to interpret and respond to gas detection system alerts correctly. Lack of training can lead to improper responses or disregard of alarms.

10. Power Source Dependency: Many gas detectors rely on a power source, and power outages or battery failures can render them ineffective. Battery-powered systems have a limited operational time before requiring replacement or recharging.

11. Limited Shelf Life: Gas sensors have a finite lifespan, and they must be replaced periodically, adding to the long-term costs.

12. Compatibility: Integrating gas detection systems with existing safety or automation systems can be complex. Compatibility issues may arise when trying to connect different systems.

Updates:

Upgrading a gas leakage detection system can enhance its performance and address some of the limitations mentioned earlier. Here are some potential updates and improvements you can consider:

1. Advanced Sensors: Replace or upgrade the gas sensors with more advanced and accurate ones. New sensor technologies can provide better sensitivity, selectivity, and faster response times.

2. Wireless Connectivity: Incorporating wireless connectivity and IoT (Internet of Things) capabilities. This allows for remote monitoring and real-time alerts, making it easier to respond to gas leaks promptly.

3. Multi-Gas Detection: Investing in multi-gas detectors that can monitor multiple types of gases simultaneously. This is especially valuable if your environment involves different gas types.

4. Integration: Integrating the gas detection system with other safety and automation systems. This allows for a more comprehensive safety infrastructure that can trigger automated responses or alert other systems.

5. Data Logging and Analysis: Implementing data logging and analysis capabilities to track gas concentration trends over time. This data can be valuable for identifying potential issues and optimizing safety protocols.

6. Smart Alarms: Using smart alarms that can differentiate between genuine gas leaks and false alarms. This can help reduce the problem of alarm fatigue.

7. Backup Power: Installing backup power solutions to ensure the system remains operational during power outages. This can include uninterruptible power supplies (UPS) or backup batteries.

8. Remote Calibration: Using sensors with remote calibration capabilities, which can simplify the maintenance process and ensure sensors remain accurate.

9. Environmental Compensation: Using sensors that can compensate for environmental factors (e.g., humidity, temperature) to reduce the impact of these variables on accuracy.

Postlab:

a) Analyse a given software design and assess the level of cohesion and coupling, identifying potential areas for improvement:

The software design of our platform demonstrates good levels of cohesion and coupling. Cohesion is evident in the well-defined components with distinct functionalities such as user management, job posting management, application management, and upskilling platform management. Loose coupling is maintained by allowing flexibility and minimal interdependence between these components. To further improve the design, the platform can benefit from enhancing cohesion by ensuring each component has a single, clear responsibility. For instance, user management should solely focus on user registration, login, and role-based access control. Similarly, job posting management should only handle creating, updating, or deleting job postings. Coupling can be reduced by minimizing dependencies between components. Changes in one component (like job posting management) should not necessitate changes in another component (like user management). This can be achieved by using interfaces or abstractions between components and ensuring data flow is well managed

b) Apply Object-Oriented principles, such as encapsulation and inheritance, to design a class hierarchy for a specific problem domain.

In the "Vehicle" domain, we establish a class hierarchy using Object-Oriented principles:

1. Vehicle (Base Class):
   - Properties: make, model, year
   - Methods: start, stop, accelerate, brake

2. Car (Inherits from Vehicle):
   - Additional Properties: numDoors, fuelType
   - Additional Methods: lockDoors, unlockDoors

3. Motorcycle (Inherits from Vehicle):
   - Additional Properties: hasHelmetStorage
   - Additional Methods: putOnHelmet, takeOffHelmet

4. Truck (Inherits from Vehicle):
- Additional Properties: cargoCapacity
- Additional Methods: loadCargo, unloadCargo

This hierarchy exemplifies encapsulation, where properties and methods are contained within each class, and inheritance, which allows specialized classes to inherit properties and methods from the base class, promoting code reusability and structure.

c)Evaluate the impact of cohesion and coupling on software maintenance, extensibility, and reusability in a real-world project scenario.

In a real-world project, cohesion and coupling have significant effects:

- Software Maintenance: High cohesion simplifies changes, and low coupling reduces unintended impacts during maintenance.

- Software Extensibility: High cohesion and low coupling ease the addition of new features and components.

- Software Reusability: Well-structured, cohesive, and loosely coupled code is more reusable in various contexts.

In practice, striking a balance between cohesion and coupling is crucial for business agility, cost savings, team collaboration, and quality assurance in longterm projects.