

**Department of Computer Engineering**

**Academic Term: First Term 2023-24**

**Class: T.E /Computer Sem – V / Software Engineering**

<b>Practical No:</b>	<b>7</b>
<b>Title:</b>	Design Using Object-Oriented Approach with Emphasis on Cohesion and Coupling in Software Engineering
<b>Date of Performance:</b>	12-09-23
<b>Roll No:</b>	9598
<b>Team Members:</b>	Ryan D'Mello

**Rubrics for Evaluation:**

<b>Sr. No</b>	<b>Performance Indicator</b>	<b>Excellent</b>	<b>Good</b>	<b>Below Average</b>	<b>Total Score</b>
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**

**Class: T.E /Computer Sem – V / Software Engineering**

**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.

## SE EXP 7: Architecture

### SHORTCOMINGS For JobSurge,a job portal for blue collar workers:

1. **Accessibility** : The Users with disabilities may face some challenges while viewing the website which may make it less accessible.
2. **Verification** : lacks a verification process, such as sending an email or a message, when a user registers for a specific event.
3. **Poor Customer Support**: Insufficient customer support can lead to frustration and dissatisfaction among users.
4. **Mobile Accessibility**: Website may face challenges while opening on mobile phone.
5. **Lack of Scalability**: May not handle a growing user base and increasing events, it can lead to performance issues and downtime.
6. **Ineffective Marketing and Promotion**: Don't have a well-defined marketing strategy, Website may struggle to attract event organizers and attendees.
7. **Limited Payment Options**: Restricting payment options can deter potential event organizers and attendees.
8. **Inaccurate Information**: User can face challenges if information entered is fake.
9. **Notification failure**: May fail to offer event notifications to users.

### UPDATES For:

1. Enhanced User Profiles: Users can now create comprehensive profiles with detailed employment history, skills, and certifications, making it easier for employers to find suitable candidates.
2. AI-Powered Job Matching: Implemented AI algorithms to match job seekers with job postings based on their profiles, improving the relevance of job suggestions.
3. Real-time Chat Support: Added a live chat feature for instant support, allowing users to get quick answers to their queries and assistance with technical issues.
4. Video Resumes: Users can upload video resumes, providing a more interactive and engaging way to showcase their skills and personality to potential employers.
5. Job Post Analytics: Employers can now track the performance of their job postings, including the number of views, applications, and applicant demographics, helping them refine their recruitment strategies.
6. Enhanced Mobile App: The mobile app has been revamped for a more user-friendly experience, with improved navigation and responsiveness.
7. In-App Video Interviews: Integrated a video interview platform, enabling employers to conduct initial video interviews directly through the portal, saving time and resources.
8. Personalized Job Alerts: Users can set up custom job alerts based on their preferences, receiving notifications about relevant job openings.
9. Skill Assessment Tests: Added an option for employers to include skill assessments as part of the application process, helping to evaluate candidates' qualifications more effectively.

10. Localization: Expanded the platform to support multiple languages and regions, making it more accessible and relevant to a global user base.

## **POSTLABS:**

### **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 long-term projects.