Department of Computer Engineering

Academic Term: First Term 2023-24

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

Practical No:	**7**
Title:	**Design Using Object-Oriented Approach with Emphasis on Cohesion and Coupling in Software Engineering**
Date of Performance:	**28/09/2023**
Roll No:	**9589**
Team Members:	**Mudabbir(9589),Muhammad(9588),Nathan(9597),Aqib(9614)**

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

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

Farmer Helper Website: Complex Object-Oriented Design Lab

1. Introduction to Object-Oriented Design:

The lab commences with an in-depth exploration of object-oriented design principles essential for the Farmer Helper Website. Core concepts including classes, objects, inheritance, polymorphism, and encapsulation are introduced to provide students with a solid foundation for crafting a sophisticated and scalable system.

2. Defining the Sample Project:

The lab project focuses on enhancing the Farmer Helper Website's Crop Management System. This involves multiple modules, including User Management, Farm Management, Crop Recommendation, and Weather Integration.

3. Cohesion in Design:

Students dive into cohesion within the Crop Recommendation module. Functions like processUserInput(), integrateExternalData(), and generateRecommendations() exhibit functional cohesion, aligning with the specific needs of the module.

4. Coupling in Design:

The interdependence between modules is explored. For instance, the Crop Recommendation module interacts with the Weather Integration module to fetch real-time data. The aim is to establish low coupling, ensuring that modules can be modified independently.

5. Applying OO Principles:
Object-oriented principles are applied to design classes, considering the complexity of the Crop Management System. For instance:
User Class:

Attributes: userID: int, username: String, email: String
Methods: login(), logout(), updatePreferences()
Farm Class:

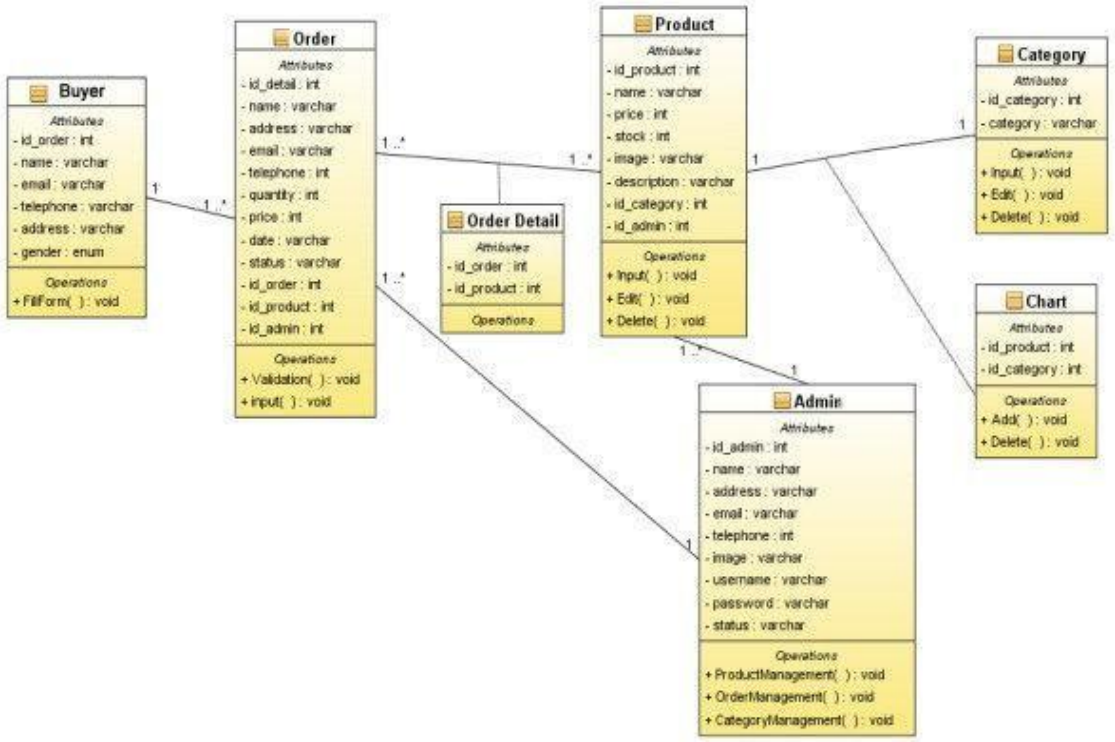
Attributes: farmID: int, location: String, size: double
Methods: addCrop(Crop), removeCrop(Crop), getWeatherForecast()
Crop Class:

Attributes: cropID: int, name: String, type: String
Methods: updateInfo(CropInfo), getRecommendations()
WeatherService Class:

Methods: getCurrentWeather(), getForecast()
CropInfo Class:

Attributes: cropID: int, growthStage: String, pests: List<String>

6. Class Diagrams:
Students create detailed Class Diagrams visually representing the intricate relationships between classes, attributes, and methods. An example includes:
Class Diagram:



7. Design Review:
In the design review session, peers provide constructive feedback on Class Diagrams. This collaborative exercise refines the design, ensuring optimal cohesion

and low coupling across the various modules.

During the design review session, the class diagrams for the Farmer Helper Website were collaboratively examined, with a focus on achieving optimal cohesion and low coupling across different modules. The following key points emerged from the review:

Cohesion Evaluation:

Strengths:

The design demonstrates high functional cohesion within each class, with methods logically grouped and aligned with the responsibilities of the respective classes.
The inclusion of specialized classes such as CropInfo for composition within the Crop class enhances clarity and maintains a clear separation of concerns.
Suggestions for Improvement:

In some instances, additional clarification or comments in the class diagrams could enhance understanding, especially regarding the purpose of specific methods or attributes.
Consideration for temporal cohesion could be explored further to ensure that operations related to the same time frame are appropriately grouped.
Coupling Evaluation:

Strengths:

The design exhibits a thoughtful effort to minimize direct dependencies between modules. Key relationships, such as between Crop and WeatherService, are appropriately managed, promoting low coupling.
The utilization of aggregation and composition relationships appropriately encapsulates the interactions between classes, allowing for flexibility in future modifications.
Suggestions for Improvement:

Further examination of potential dependency paths, especially in scenarios involving external data integration, could ensure that modules remain resilient to changes and are modular in design.
Providing explicit dependency arrows and multiplicity notations would enhance the clarity of relationships, aiding in a more comprehensive understanding of the system's structure.
Class Diagram Clarity:

Strengths:

The class diagrams effectively capture the intricate relationships between classes, showcasing a comprehensive understanding of the system's architecture.
The use of appropriate symbols and annotations contributes to the clarity of the diagrams.
Suggestions for Improvement:

Consider grouping related attributes and methods within classes, especially for classes with a considerable number of elements, to enhance the visual organization of information.
Ensuring consistency in the application of notation conventions would contribute to a more polished and professional appearance.
Overall Observations:

Positive Aspects:

The design reflects a strong grasp of object-oriented principles and their practical application to the Farmer Helper Website.
The collaborative nature of the design review session facilitated valuable discussions and insights among peers.
Recommendations:

Continued collaboration and discussion among team members can contribute to further refinement and optimization of the design.
Encouraging documentation practices, such as comments within the class diagrams, can serve as useful references for future developers and maintainers.
The design review session was instrumental in refining the Farmer Helper Website's class diagrams, ensuring that the design aligns with best practices, promotes maintainability, and is well-suited for the dynamic requirements of the agricultural technology platform.

8. Reflection:
The lab concludes with a reflective discussion on the significance of object-oriented design principles, cohesion, and coupling in crafting a maintainable and flexible Crop Management System for the Farmer Helper Website. Students draw insights from their practical experiences, appreciating the critical role of thoughtful design in the success of the website.

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.