**Title (Units):** **COMP4107 Software Design, Development and Testing (3,3,0)**

**Course Aims:** This course covers software design, development, and testing principles, with a particular focus on leveraging a contemporary programming language and environment, implementing design patterns, developing applications for a specific platform such as Android, and employing effective software testing techniques. Throughout the course, students will acquire knowledge and practical skills, and gain hands-on experience in designing, developing, and testing software applications or innovation system utilizing cutting-edge tools and frameworks.

**Prerequisite:** COMP3047 Software Engineering OR COMP3116 Modern Software Engineering

**Course Intended Learning Outcomes (CILOs):**
Upon successful completion of this course, students should be able to:

| No. | Course Intended Learning Outcomes (CILOs) |
|---|---|
| | **Knowledge** |
| 1 | Explain advanced software design principles with a contemporary programming language and environment. |
| | **Professional Skill** |
| 2 | Implement a range of design patterns to create software solutions that are flexible, maintainable, and/or scalable. |
| 3 | Develop feature-rich platform-based applications using modern development tools. |
| 4 | Apply effective software testing strategies and techniques to ensure the quality of the software being developed. |

**Calendar Description:** This course covers software design, development, and testing principles, with a particular focus on leveraging a contemporary programming language and environment, implementing design patterns, developing applications for a specific platform such as Android, and employing effective software testing techniques. Throughout the course, students will acquire knowledge and practical skills, and gain hands-on experience in designing, developing, and testing software applications utilizing cutting-edge tools and frameworks.

**Teaching and Learning Activities (TLAs):**

| CILOs | Type of TLA |
|---|---|
| 1 | Lectures and in-class exercises for acquiring software design principles in a contemporary programming language and environment |
| 2 | Software development exercises and projects for implementing design patterns. |
| 3 | Hands-on labs and projects for developing applications using software development platform. |
| 4 | Practical exercises and projects for applying software testing strategies and techniques. |

**Assessment:**

| No. | Assessment Methods | Weighting | CILOs to be addressed | Description of Assessment Tasks |
|---|---|---|---|---|
| 1 | Coding Assignments and In-class Exercises | 35% | 1-4 | The coding assignment and in-class exercises are designed to enhance students' understanding of software design patterns and their practical application. |
| 2 | Group Project | 25% | 1-4 | The group project provides opportunities for students to practice and demonstrate their skills and abilities in applying and integrating the principles and techniques of software design, development, and testing learned. |
| 3 | Examination | 40% | 1-4 | Final examination questions evaluate students' in-depth knowledge and understanding of the key |

| | | | | |
|---|---|---|---|---|
| | | | | principles and techniques necessary for developing reliable software systems. |

**Assessment Rubrics:**

| Criteria | Software Design Principles, Programming and Environment | Design Patterns | Platform-based Development with Modern Framework | Software Testing and Quality Assurance |
|---|---|---|---|---|
| A (Excellent) | Demonstrates a clear understanding of software design principles and methodologies. Applies concepts effectively. | Demonstrates a thorough understanding of various design patterns and their application. Implements patterns effectively. | Applies a modern framework effectively to develop platform-based applications. Demonstrates strong skills in UI development. | Implements comprehensive software testing strategies and techniques effectively. Demonstrates strong understanding of quality assurance. |
| B (Good) | Shows a good understanding of software design principles and methodologies. Applies concepts accurately. | Shows a good understanding of design patterns and their application. Implements patterns accurately. | Applies a modern framework accurately to develop platform-based applications. Demonstrates solid skills in UI development. | Implements software testing strategies and techniques accurately. Demonstrates good understanding of quality assurance. |
| C (Satisfactory) | Demonstrates a basic understanding of software design principles and methodologies. Applies concepts with some errors. | Demonstrates a basic understanding of design patterns and their application. Implements patterns with some errors. | Applies a modern framework with some errors to develop platform-based applications. Demonstrates basic skills in UI development. | Implements software testing strategies and techniques with some errors. Demonstrates basic understanding of quality assurance. |
| D (Marginal Pass) | Demonstrates limited understanding of software design principles and methodologies. Applies concepts with significant errors. | Demonstrates limited understanding of design patterns and their application. Implements patterns with significant errors. | Applies a modern framework with significant errors to develop platform-based applications. Demonstrates limited UI skills. | Implements software testing strategies and techniques with significant errors. Demonstrates limited understanding of QA. |
| F (Fail) | Lacks understanding of software design principles and methodologies. Applies concepts incorrectly or incompletely. | Lacks understanding of design patterns and their application. Implements patterns incorrectly or incompletely. | Fails to apply a modern framework to develop platform-based applications. Lacks UI development skills. | Fails to implement software testing strategies and techniques. Lacks understanding of quality assurance. |

**Course Content and CILOs Mapping:**

| **Content** | | **CILO No.** |
|---|---|---|
| I | Software Design Principles, Programming and Environment | 1 |
| II | Design Patterns | 2 |
| III | Platform-based Development with Modern Framework | 3 |
| IV | Software Testing and Quality Assurance | 4 |

**References:**


- Soshin and A. Arhipov, *Kotlin Design Patterns and Best Practices: Build scalable applications using traditional, reactive, and concurrent design patterns in Kotlin*, 2nd ed. Birmingham, England: Packt Publishing, 2022.
- E. Freeman and E. Robson, *Head first design patterns: Building extensible and maintainable object-oriented software*, 2nd ed. Sebastopol, CA: O′ Reilly Media, 2020.
- N. Smyth, *Jetpack compose 1.3 essentials: Developing android apps with jetpack compose 1.3, android studio, and kotlin*. Payload Media, 2023.
- L. Gleason, V. Gonda, and F. Sproviero, *Android test-driven development by tutorials (second edition): Learn android TDD by building real-world apps*. Razeware, 2021.
- A. Forrester, E. Boudjnah, A. Dumbravan, and J. Tigcal, *How to Build Android Apps with Kotlin: A practical guide to developing, testing, and publishing your first Android apps*, 2nd ed. Birmingham, England: Packt Publishing, 2023.
- Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1994.


**Course Content:**

**Topic**

I. Software Design Principles, Programming and Environment
   A. Introduction to software design principles and methodologies
   B. Basics of a modern object-oriented programming language (e.g., Kotlin)

II. Design Patterns
   A. Introduction to design patterns and their significance
   B. Creational patterns: Singleton, Factory, Abstract Factory
   C. Structural patterns: Adapter, Decorator
   D. Behavioral patterns: Observer, Strategy, Command and more
   E. Modern patterns encompassing functional, reactive, and concurrent programming

III. Platform-based Development with Modern Framework
   A. Overview of a modern framework for platform-based development (e.g., Android)
   B. Utilizing framework features for building user interfaces, data management, navigation, and/or connectivity

IV. Software Testing and Quality Assurance
   A. Importance of software testing and quality assurance in the development process
   B. Types of software testing: unit testing, integration testing, and UI testing
   C. Test-driven development (TDD) principles
   D. Writing unit tests with testing frameworks
   E. Automated UI testing with a testing framework