Republic of Rwanda
Ministry of Education

RTB | RWANDA TVET BOARD

SOFTWARE ENGINEERING

SPESE402

APPLY SOFTWARE ENGINEERING

Competence

RQF Level: 4

Learning Hours
100

Credits:10

Sector: ICT and Multimedia

Trade: Software Programming and Embedded Systems

Module Type: Specific Module

Curriculum: ICTSES4002 TVET Certificate IV in Software Programming and Embedded System

1200

Issue Date: August 2023

Purpose statement	This module aims to equip learners with knowledge and skills necessary for analyzing, designing, implementing and managing a software project by providing a feasibility study, estimating cost and schedule, applying quality assurance, and providing configuration management of Software projects. By the end of this module, learners will be able to analyse, design, implement and manage a software project with minimum supervision.

Learning assumed to be in place	- Embedded Systems Hardware Design - Networking Fundamentals - Web APIs

<table>
<tr><td rowspan="5">Delivery modality</td><td colspan="3">Training delivery</td><td>100%</td><td colspan="2">Assessment</td><td>Total 100%</td></tr>
<tr><td colspan="3">Theoretical content</td><td>30%</td><td rowspan="3">Formative assessment</td><td>30%</td><td rowspan="3">50%</td></tr>
<tr><td colspan="3">Practical work:</td><td rowspan="3">70%</td><td rowspan="3">70%</td></tr>
<tr><td>Group project and presentation</td><td>30%</td></tr>
<tr><td>Individual project /Work</td><td>40%</td><td colspan="2">Summative Assessment</td><td>50%</td></tr>
</table>

Elements of Competence and Performance Criteria

Elements of competence	Performance criteria
1. **Manage software requirements and analysis**	1.1. Feasibility study is properly conducted according to methodical standards
	1.2. System requirements specification is properly developed in accordance with precise guidelines.
	1.3. Data collection is accurately performed based on appropriate standards.
	1.4. SDLC phases and models are properly described in line with the software development process.
2. **Design system**	2.1. Principles of software design are clearly described based on software requirements
	2.2. System logical design is properly designed in line with software requirements specifications
	2.3. System architecture design is adequately designed in line with software requirements
	2.4. System physical design is properly designed in line with software requirements
3. **Perform Software Testing**	3.1. Software testing is performed in line with system specifications
	3.2. Software quality metrics and techniques are properly identified in line with software quality assurance.
	3.3. Test planning and test case design are clearly described in line with software quality assurance.
	3.4. Test automation and testing tools are performed in line with development and deployment environment settings.

	3.5. Software documentation is well elaborated in line with software engineering requirements
4. **Manage Software Project**	4.1. Software project is appropriately planned in line with system requirements
	4.2. Project Team is properly managed based on project plan
	4.3. Project is well monitored in line with the project plan.
5. **Elaborate Software Ethics and Professionalism**	5.1. Ethical considerations in software development are explained
	5.2. Professional responsibility is clearly described in line with software development and embedded systems fields.
	5.3. Code of conduct is effectively explained in line with software development and embedded systems fields.

Intended Knowledge, Skills and Attitude

Knowledge	Skills	Attitude

Describe: ✓ Feasibility study methodologies ✓ System requirements guidelines ✓ Data collection techniques ✓ SDLC phases and models ✓ Software design principles ✓ Logical design principles ✓ System architecture ✓ Physical design principles ✓ Software testing methodologies ✓ Software quality metrics ✓ Test planning and case design ✓ Test automation and tools ✓ Software documentation ✓ Project planning process and tools ✓ Team management principles ✓ Ethical considerations principles ✓ Professional responsibilities ✓	✓ Apply design principles ✓ Design logical structure ✓ Perform software testing ✓ Identify quality metrics and techniques ✓ Elaborate documentation ✓ Plan software projects ✓ Explain codes of conduct ✓ Conduct feasibility studies ✓ Develop precise specifications ✓ Accurate data collection ✓ Implement project using SDLC phases and models ✓ Architect system ✓ Design physical aspects ✓ Plan tests and design test cases ✓ Use testing automation and tools ✓ Manage project teams ✓ Monitor projects	✓ Attention to detail ✓ Commitment to precise documentation ✓ Willingness to adhere to standards ✓ Openness to adapting to SDLC approaches ✓ Dedication to design principles ✓ Commitment to logical design ✓ Openness to adapting design ✓ Willingness to iterate and improve ✓ Commitment to software quality ✓ Dedication to comprehensive testing ✓ Openness to automation for efficiency ✓ Attention to detail in documentation ✓ Willingness to document testing processes ✓ Commitment to effective planning ✓ Dedication to team collaboration ✓ Willingness to adapt to changing plans ✓ Commitment to ethical conduct ✓ Dedication to upholding responsibilities ✓ Willingness to promote professionalism ✓ Codes of conduct

Course content	
Learning outcomes	At the end of the module the learner will be able to: 1. **Manage software requirements and analysis** 2. **Design system** 3. **Perform Software Testing** 4. **Manage Software Project** 5. **Elaborate Software Ethics and Professionalism**

Learning hours:**20**

Learning outcome 1: **Manage software requirements and analysis**

Indicative content

- **Description of SDLC phases and models**
  - ✓ Description of SDLC phases
    - Define SDLC
    - Processes
  - ✓ Description of SDLC models
    - Waterfall Model
    - Agile Model
    - Scrum
    - V-Model
    - Kanban
    - RAD (Rapid Application Development)
    - DevOps
- **Performing Feasibility study**
  - ✓ Introduction of feasibility study
    - Definition
    - Importance of feasibility study
  - ✓ Description of Types of Feasibility study
    - Technical feasibility
    - Operational feasibility
    - Economic feasibility
    - Schedule feasibility
    - Legal and Ethical feasibility
  - ✓ Application of Feasibility Study Process
    - Data Gathering and Analysis
    - Risk Management
    - Reporting and Decision
    - Communication and Implementation:
    - Documentation and Knowledge Management
- **Data collection**

- ✓ Introduction to data collection
  - Definition
  - importance
- ✓ Types of Data
  - Qualitative and quantitative data
  - Structured data and unstructured data
  - Primary and secondary data
- ✓ Data Collection Objectives
- ✓ Data Sources
- ✓ Data Collection Techniques
  - Stakeholder Interviews
  - Surveys and Questionnaires
  - Observations
  - Prototyping
  - Document Analysis
  - Focus Groups
  - Use Case Analysis
  - Data Mining
- ✓ Description of Data Collection Tools and Technologies
  - Survey software (SurveyMonkey, Google form, Formstack, etc)
  - Data visualization tools (Google Data Studio, D3.js, etc)

- **System requirements specification (SRS) Documentation process**
  - ✓ Introduction to System Requirement Specification
    - Importance
    - SRS readers
  - ✓ Types of Requirements
    - User requirements
    - System requirements
    - Software Specification
  - ✓ Requirement Engineering Process
    - Feasibility Study
    - Requirement Gathering

- Software Requirement Specification
- Software Requirement Validation

✓ Requirement elicitation

- Requirement elicitation process
- Requirement Elicitation Technique

✓ Requirement Specification Format

✓ Requirements validation

- Characteristics of good requirements
- The “rule of thumb”

✓ Software requirement characteristics

- Different types of checks in requirements validation
- Popular requirements validation techniques

Resources required for the learning outcome	
Equipment	Computers with the necessary software installed and a reliable internet connection
Materials	• Smartboards & Markers • Textbooks • Online resources • Educational videos • PPT Presentations
Tools	Software tools to assist in project management, requirements gathering, design, coding, testing, and deployment like: • JIRA • Trello • GitHub • Microsoft Visio
Facilitation techniques	• Brainstorming ➢ Group discussion on SDLC phases
Formative assessment methods /(CAT)	• Oral assessment • Written assessment • Practical assessment • Scenario-Based Assessment

Learning outcome 2: Design system	Learning hours: 25

Indicative content

- **Description of Principles of software design**
  - ✓ Introduction to Software Design Principles
    - Definition
    - Importance of software design
  - ✓ Overview of SOLID Principles
    - Single Responsibility
    - Open/Closed
    - Liskov Substitution
    - Interface Segregation
    - Dependency Inversion
  - ✓ DRY, KISS, and YAGNI Principles
    - Don't Repeat Yourself" (DRY)
    - "Keep It Simple, Stupid" (KISS)
    - "You Ain't Gonna Need It" (YAGNI)
  - ✓ High Cohesion and Low Coupling
  - ✓ Design Patterns
    - Definition
    - Categorization of Design Patterns
    - Pattern Structure
    - Creational Design Patterns (Singleton, Factory Method, Abstract Factory, Builder, Prototype)
    - Structural Design Patterns (Adapter, Decorator, Composite, Bridge, Proxy)
    - Behavioral Design Patterns (Observer, Strategy, Command, State, Template Method, Iterator, Chain of Responsibility, Visitor, Memento)
  - ✓ UML (Unified Modelling Language)
    - Definition
    - Importance

- Types of UML diagrams (Use Case Diagrams, Class Diagrams, Sequence Diagrams, Activity Diagrams, State Machine Diagrams, Component Diagrams, Deployment Diagrams, Package Diagrams, Object Diagrams, etc)

- **System Logical Design**

✓ Introduction to Logical Design
- Definition and role in software development.
- Link between requirements analysis and implementation.

✓ Components of Logical Design
- High-level architecture: Components, modules, subsystems.
- Data flow and data model: Data entities, attributes, relationships.
- Control flow design: User interactions, processes, responses.

✓ Design Considerations
- Scalability, extensibility, and modifiability.
- Security and access control planning.
- Performance considerations and architectural choices.

✓ Iterative and Adaptive Design
- Iterative nature and responsiveness to changes.
- Incorporating feedback into design adjustments.

✓ Transition to Detailed Design
- Using logical design as a foundation.
- Further breaking down components and providing implementation details.

- **System Architecture Design**

✓ Architectural Styles
- Overview of monolithic, microservices, client-server architectures.

✓ Factors Influencing Architectural Decisions
- Business requirements, user needs, technological constraints.
- Impact of chosen architecture on system properties.

✓ Architectural Patterns
- Layered architecture: Presentation, business logic, data access layers.
- Model-View-Controller (MVC) pattern: Separation of concerns.
- Event-driven architecture: Communication through events and messages.

- **System Physical Design**

✓ Introduction to Physical Design

- Importance of translating logical design into a tangible system.
- Addressing hardware and infrastructure requirements.

✓ Hardware and Infrastructure
- Servers, databases, networking considerations.
- Deployment environments: Development, testing, production.

✓ Scaling Strategies
- Horizontal scaling and load balancing techniques.
- Ensuring the system handles increased demands effectively.

✓ Disaster Recovery and Redundancy
- Backup strategies and data protection measures.
- Planning for system resilience and fault tolerance.

● **Code Organization and Best Practices**

✓ Code Organization Basics
- Directory structures and separation of concerns.
- Role of naming conventions in code readability.

✓ Design Patterns for Modularity
- Implementation of common design patterns.
- Encouraging reusability and maintainability.

✓ Version Control and Collaboration
- Introduction to Git and version control systems.
- Collaborative coding and managing code changes.

✓ Coding Standards and Quality
- Consistent code formatting and indentation.
- Effective use of comments and documentation.

✓ Error Handling and Testing
- Implementing error-handling mechanisms.
- Importance of unit testing, integration testing, and code quality.

✓ Dependency Management
- Managing libraries, dependencies, and third-party components.
- Avoiding version conflicts and staying up to date.

Resources required for the Learning outcome	
Equipment	● Computers, ● Laptops ● servers with the necessary software installed and a reliable internet connection

Materials	• Smartboards & Markers • Textbooks • Online resources • Educational videos • PPT Presentations
Tools	Software tools to assist in project management the design like: • Figma • Lucidchart • Nulab • StarUML
Facilitation techniques	• Brainstorming sessions • Group discussion on software design • Practical exercise on software design • prototyping • user interviews • design workshops
Formative assessment methods /(CAT)	• Oral assessment • Written assessment • Practical assessment • Scenario-Based Assessment

Learning outcome 3: Perform Software Testing	**Learning hours: 25**
Indicative content	

- **Description of software testing**
  - ✓ Understanding Software Testing
    - Definition and importance of software testing.
    - The role of testing in ensuring software quality.
  - ✓ Aligning Testing with System Specifications
    - Explanation of system specifications (functional, non-functional).
    - Importance of ensuring testing is aligned with these specifications.
    - How testing helps verify and validate requirements.
  - ✓ Types of Testing
    - Introduction to various testing types (unit, integration, system, acceptance).
    - Demonstrating how each type corresponds to different specification levels.
    - Examples of test cases for different testing types.
- **Identification of Software quality metrics and techniques**
  - ✓ Software Quality Metrics
    - Definition and significance of software quality metrics.
    - Examples of metrics (code coverage, defect density) and how they measure quality.
    - Importance of choosing appropriate metrics for the project.
  - ✓ Quality Assurance Techniques
    - Introduction to quality assurance techniques (reviews, inspections, audits).
    - How these techniques ensure compliance with quality standards.
    - Role of reviews and inspections in identifying defects early.
- **Test case design**
  - ✓ Test Planning
    - Importance of comprehensive test planning.
    - Steps in test planning (scope, resources, scheduling).
    - Explanation of test case design techniques (equivalence partitioning, boundary value analysis).
    - Ensuring test coverage and traceability.

- Structuring test cases for clarity and effectiveness.
- Creating a test plan document.

✓ Test Plan Quality Metrics

- Completeness
- Consistency
- Traceability
- Readability
- Maintainability

- **Test automation and testing tools**

✓ Test Automation Introduction

- Benefits of test automation (efficiency, repeatability).
- Integrating automation into the development process.

✓ Selecting Testing Tools

- Factors influencing the choice of testing tools.
- Overview of popular testing frameworks (Selenium, Cypress, JUnit, TestNG).
- Aligning tools with technology stack and project requirements.

✓ Using Testing Tools

- Hands-on practice with testing tools.
- Writing and executing automated test scripts.
- Integrating test automation into continuous integration processes.

- **Software documentation**

✓ Importance of Documentation

- Significance of clear and comprehensive software documentation.
- Different types of documentation (user manuals, technical guides).

✓ Creating Effective Documentation

- Guidelines for creating structured and user-friendly documentation.
- Ensuring documentation remains up-to-date.
- Documentation tools and formats.

Resources required for the Learning outcome	
Equipment	Computers with the necessary software installed and a reliable internet connection
Materials	• Smartboards & Markers • Textbooks • Online resources • Educational videos • PPT Presentations
Tools	Software tools to assist in software testing like: • Postman • Selenium • Cypress • JMeter • Bugzilla
Facilitation techniques	• Brainstorming • Group discussion on software testing • Practical exercise on software testing
Formative assessment methods /(CAT)	• Oral assessment • Written assessment • Practical assessment • Scenario-Based Assessment

Learning outcome 4: Manage Software Project	Learning hours: 25
Indicative content	
• **Software Project Planning** ✓ **Introduction to Project Planning** - Importance of project planning in software development. - Role of project planning in achieving project success. ✓ **Defining Project Objectives and Scope** - Identifying project goals, objectives, and scope boundaries.	

- Techniques: Stakeholder interviews, workshops, surveys.
- Documenting project scope statements and setting clear expectations.
- Tools: Project charter templates, scope statement templates.

✓ **Work Breakdown Structure (WBS)**

- Creating a WBS to break down project deliverables into manageable tasks.
- Techniques: Decomposition, mind mapping, tree diagrams.
- Organizing tasks hierarchically and assigning resources.
- Technique: Developing a detailed WBS using project management software (e.g., Microsoft Project, Wrike).

✓ **Project Scheduling**

- Network diagram
- Techniques for creating project schedules (Critical Path Method (CPM), Program Evaluation and Review Technique (PERT).
- Sequencing tasks, estimating durations, and determining critical paths.
- Tools: Gantt chart software (e.g., Microsoft Project, Trello).

✓ **Resource Allocation and Management**

- Allocating human and material resources to project tasks.
- Balancing resource constraints and resolving conflicts.
- Technique: Resource leveling using project management software.

✓ **Risk Assessment and Mitigation**

- Identifying potential project risks and assessing their impacts.
- Developing risk mitigation strategies and contingency plans.
- Create a Risk Register to document and manage identified risks and mitigation plans.
- Tools: Risk management software (e.g., RiskWatch, RiskyProject)

✓ **Cost Estimation and Budgeting**

- Estimating project costs, including personnel, equipment, and overhead.
- Developing project budgets and tracking expenditures.
- Technique: Cost-Benefit Analysis and Budgeting using spreadsheet software.

- **Manage Software Project Team**

✓ **Team Formation and Roles**

- Forming project teams and defining roles and responsibilities.
- Identifying team members' skills and expertise.
- Tools: Team collaboration platforms (e.g., Slack, Microsoft Teams).

✓ **Effective Communication**

- Importance of clear and open communication within the project team.
- Strategies for promoting effective communication and collaboration.
- Tools: Communication and collaboration tools (e.g., Slack, email, project management software).

✓ **Motivation and Leadership**

- Leadership styles and techniques for motivating project team members.
- Creating a positive work environment and fostering team morale.
- Technique: Motivation techniques like recognition and rewards.

✓ **Conflict Resolution:**

- Identifying sources of conflict within the project team.
- Techniques for resolving conflicts and maintaining team harmony.
- Tool: Conflict Resolution Techniques Matrix.

- ✓ **Performance Management**
  - Setting performance expectations and goals for team members.
  - Monitoring and evaluating team performance and providing feedback.
  - Tools: Performance management software (e.g., BambooHR, Workday)
- ✓ **Training and Skill Development**
  - Identifying skill gaps and providing training opportunities for team members.
  - Supporting professional growth within the project team.

- **Project Monitoring**
  - ✓ **Project Metrics and KPIs**
    - Defining relevant project metrics and key performance indicators (KPIs).
    - Monitoring progress against established metrics.
    - Technique: Balanced Scorecard Approach for KPI selection and monitoring.
  - ✓ **Project Status Reporting:**
    - Creating regular status reports to communicate project progress to stakeholders.
    - Highlighting accomplishments, issues, and potential risks.
  - ✓ **Change Management**
    - Monitoring changes to project scope, schedule, and requirements.
    - Evaluating change requests
  - ✓ **Issue Identification and Resolution**
    - Techniques for identifying project issues and bottlenecks.
    - Developing strategies to address and resolve issues promptly.
    - Tools: Issue tracking software (e.g., JIRA, Trello)
  - ✓ **Adaptive Planning**
    - Importance of adapting project plans based on monitoring results.
    - Modifying schedules, resource allocations, and strategies as needed.

Equipment	• Laptops or Computers • Projector for Presentations • Interactive Whiteboard or Chalkboard • Audiovisual Equipment for Online Classes • Video Conferencing Equipment for Virtual Classes • Internet Connectivity
Materials	• Project Management Books and Guides • Case Studies • Templates and Examples • Interactive Content • Whiteboards and Markers • Laptop and Presentation Tools • Online Learning Platforms • Workbooks and Exercises
Tools	• Jira • Trello • Microsoft Project

	• Asana • RiskWatch, RiskyProject • Slack, Microsoft Teams, email • BambooHR
Facilitation techniques	• Group Discussions • Brainstorming • Role-Playing • Peer Reviews • Group Projects • Case-Based Learning
Formative assessment methods /(CAT)	• Oral assessment • Written assessment • Practical Assessment • Scenario-Based Assessment

Learning outcome 5: Elaborate Software Ethics and Professionalism	**Learning hours:5**
Indicative content	

- **Ethical Considerations in Software Development**
  - ✓ **Introduction to Ethical Considerations**
    - Importance of ethics in software development and its impact on society.
    - Overview of ethical challenges in software development.
  - ✓ **Privacy and Data Ethics**
    - Addressing user data privacy, consent, and security considerations.
    - Discussing the ethical implications of data collection and storage.
  - ✓ **Bias and Fairness:**
    - Exploring biases in algorithms and software that can result in discrimination.
    - Discussing ways to mitigate bias and ensure fairness.
  - ✓ **Transparency and Accountability**
    - Ethical responsibility to provide transparency in how software operates.
    - Discussing accountability for software decisions and outcomes.
  - ✓ **Accessibility and Inclusivity**
    - Ethical obligation to make software accessible to all users.
    - Addressing the importance of considering diverse user needs.
  - ✓ **Intellectual Property and Open-Source Ethics**
    - Discussing ethical use of intellectual property and open-source software.
    - Exploring licensing and attribution considerations.
- **Description of Professional Responsibility**
  - ✓ **Introduction to Professional Responsibility**
    - Defining professional responsibility in the context of software development.
    - Emphasizing the role of software professionals in delivering quality and ethical products.

- ✓ **Roles and Responsibilities**
  - Describing the responsibilities of software developers, testers, project managers, and other team members.
  - Discussing the ethical implications of each role's decisions.
- ✓ **User-Centered Design**
  - Explaining the responsibility to prioritize user needs and well-being.
  - Discussing the impact of user-centered design on software ethics.
- ✓ **Safety and Reliability**
  - Addressing the ethical obligation to ensure software safety and reliability.
  - Discussing the consequences of software failures.
- ✓ **Continuous Learning and Improvement**
  - Emphasizing the importance of staying updated with industry trends and best practices.
  - Discussing ethical implications of not keeping skills current.

- **Code of Conduct in Software Development**
  - ✓ **Introduction to Code of Conduct**
    - Defining a code of conduct and its purpose in guiding ethical behavior.
    - Explaining its role in maintaining professionalism and trust.
  - ✓ **Components of a Code of Conduct:**
    - Describing the key elements of a code of conduct (values, principles, and expected behaviors).
  - ✓ **Adaptation to Software Development**
    - Explaining how a code of conduct is tailored to address ethical challenges in these fields.
    - Discussing examples of ethical principles specific to software development.
  - ✓ **Code of Conduct Implementation**
    - Discussing strategies for enforcing the code of conduct and promoting ethical behavior.
    - Addressing challenges and resistance that may arise.
  - ✓ **Reporting Violations and Consequences**
    - Explaining the process for reporting violations and the potential consequences.
    - Discussing whistleblower protection and handling sensitive situations.

Equipment	• Laptops or Computers • Projector for Presentations • Audiovisual Equipment for Online Classes • Internet Connectivity
Materials	• Ethics and Professionalism Resources (Articles, Codes of Ethics) • Interactive Content (Videos, Infographics) • Laptop and Presentation Tools • Online Learning Platforms • Workbooks and Exercises
Tools	• Virtual Classrooms (e.g., Zoom, Microsoft Teams) • Professionalism Assessment Tools (Self-assessment quizzes)

	• Ethical Dilemma Scenarios
Facilitation techniques	• Group Discussions • Role-Playing • Debates • Guest Speakers • Scenario Analysis
Formative assessment methods /(CAT)	• Oral assessment • Written assessment • Scenario-Based Assessment

Integrated/Summative assessment (For specific module)

Integrated situation

Integrated Scenario: The Hotel Ordering System

Scenario Background:

The Hotel Ordering System allows the user of a web browser to order pizza for home delivery. To place an order, a shopper searches to find items to purchase, adds items one at a time to a shopping cart, and possibly searches again for more items. When all items have been chosen, the shopper provides a delivery address. If not paying with cash, the shopper also provides credit card information.The system has an option for shoppers to register with the pizza shop.They can then save their name and address information, so that they do not have to enter this information every time that they place an order.

Key responsibilities:

As a Software Engineer, develop a use case diagram, Data flow Diagram and SRS for a use case for placing an order, Place Order. The use case should show a relationship to two previously specified use cases, Identify Customer, which allows a user to register and log in, and Pay by Credit, which models credit card payments.

Project tasks:

1. Identification of requirement
2. Develop a use case diagram and Data flow Diagram
3. Develop SRS
4. Identify techniques for creating project schedules

Resources

Tools	Computers and internet
Equipment	• Trello • Microsoft Project • Asana • RiskWatch, RiskyProject

	• Slack, Microsoft Teams, email • BambooHR
Materials/ Consumables	• Ethics and Professionalism Resources (Articles, Codes of Ethics) • Interactive Content (Videos, Infographics) • Laptop and Presentation Tools • Online Learning Platforms • Workbooks and Exercises

Assessable outcomes	**Assessment criteria (Based on performance criteria)**	**Indicator**	**Observation**		Marks allocation
			Yes	No	
Learning Outcome 2: Design system (80%)	2.1 Principles of software design are clearly described based on software requirements	SRS is described and designed			20 marks
	2.2 System logical design is properly designed in line with software requirements specifications	System logical design is designed			20 marks
	2.3System architecture design is adequately designed in line with software requirements	System architecture design is designed			20 marks
	2.4 System physical design is properly designed in line with software requirements	System physical design is designed			20 marks
Learning Outcome 4: Manage Software Project (20%)	4.3 Project is well planned in line with the project plan methods	Project is planned in line with the project plan methods			20 marks
Total marks					
Percentage Weightage		100%			
Minimum Passing line % (Aggregate): 70%					

References:

1. "**Software Engineering: A Practitioner's Approach**" by Roger S. Pressman

2. "**Design Patterns: Elements of Reusable Object-Oriented Software**" by Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides
3. "**Clean Code: A Handbook of Agile Software Craftsmanship**" by Robert C. Martin
4. "**The Pragmatic Programmer: Your Journey to Mastery**" by Andrew Hunt and David Thomas
5. "**Software Engineering: Theory and Practice**" By Forrest Shull and Roseanne Tesoriero