Pokhara University
Faculty of Science and Technology

Course Code.: CMP 382 (3 Credits) Full marks: 100 Course Title: Software Dependability **(3-1-2)** Pass marks: 45 Nature of the course: Theory/Practice/Theory & Practice Time per period: 1 hour Year, Semester: Total periods: 45 Level: Bachelor Program: BE

1. Course Description

This course is designed to provide students with theoretical knowledge and practical skills in ensuring the dependability of software systems across a range of contexts and industries**.** In software engineering, dependability is the ability to provide services that can be defensibly trusted over some time. This may also encompass mechanisms designed to increase and maintain the dependability of a system or software. This course deals with software dependability, which is a measure of a system's different properties, such as availability, reliability, fault tolerance, rejuvenation, survivability, safety, and security.

2. General Objectives

To provide students with a solid understanding of the fundamental concepts and components of Software reliability. By the end of this course, students will be able to:

1. Understand the key concepts and dimensions of software dependability. 2. Analyze and model software reliability and availability using mathematical models. 3. Implement fault-tolerant systems and assess their effectiveness.
4. Apply software rejuvenation techniques to improve system longevity. 5. Design and evaluate security measures to enhance software dependability

3. Contents in Detail

Specific Objectives	**Contents**
:	
Define software dependability and its importance in software engineering. Identify and describe the key dimensions of software dependability. **Unit 1:Software Dependability (4 hours)**	● Overview of Software Dependability ● Key Dimensions of Dependability: Availability, Reliability, Safety, Security,Faults,Error and Failure ● Fault tolerance and recovery techniques ● Dependability vs reliability Vs safetyvs security

Explain the basics of Markov Chain modeling

Differentiate between types of Markov Chains and their uses in dependability analysis.

Identify scenarios where Markov modeling is and isn't appropriate.

Explain the significance of software reliability in system dependability. Calculate and interpret key reliability metrics.
Apply mathematical models to analyze and predict software reliability.

Define and explain the concept of software availability.
Calculate availability metrics and interpret their significance.
Model and analyze software availability.

Identify the unique challenges of developing safety-critical software. Develop safety requirements and perform hazard analysis.
Create safety cases to support software certification.
.

Markov Chain Modeling (4 hours)

● Introduction to Markov Chains and Their Application in Software Dependability
● Types of Markov Chains: Discrete-Time, Continuous-Time
● Advantages and Disadvantages of Markov Modeling
● When Not to Use Markov Modeling

Unit 3: Software Reliability (8 hours)

● Introduction to Software Reliability
● Significance of Reliability in Software Systems ● Reliability Metrics: MTBF, MTTF, Failure Rate ● Reliablity Prediction and estimation
prediction(MTTF,MTBF,MTTCF(mean time to critical Failure),avaibality,relaiblity and defect density of the software
● FMEA
● Mathematical Models of Software Reliability

Unit 4:Software Availability (4 hours)

● Introduction to Software Availability
● Importance of Availability in System Dependability
● Availability Metrics: Uptime, Downtime, Availability Ratio
● Mathematical Models of Availability

Unit 5: Software Safety (5 hours)

● Safety-Critical Systems
● Safety Requirements and Hazard Analysis ● Safety Engineering Processes: Design, Implementation, Testing
● Safety Cases and Their Role in Certification

Explain the principles of software fault tolerance and its importance.
Design and implement fault-tolerant systems using architectural patterns. Evaluate the effectiveness of fault tolerant mechanisms in improving dependability.

Unit 7: Software Rejuvenation (5 hours)

Understand the concept of software rejuvenation and its role in maintaining system dependability.

Apply analytical models to plan and implement software rejuvenation strategies.

● Introduction to Software Rejuvenation and Software Aging
● Analytical Models for Software Rejuvenation
● Software Rejuvenation in Transaction-Based Software Systems
● Case Study: IBM X-Series Cluster Servers ● Approaches and Methods of Software Rejuvenation
● Granularity of Rejuvenation

Describe the role of security in software dependability and Develop security requirements and design secure systems.

Apply security testing techniques to ensure the dependability of software

Unit 8:Software Security (5 hours)

Unit 6: Software Fault Tolerance (6 hours)

● Introduction to Software Fault Tolerance ● System Survivability and Measuring System Survivability
● Design pattern for Fault Tolerance.
● Fault Minimization and Tolerance Techniques
● Fault Tolerance Architecture: N-Version Programming, Recovery Blocks

Dependability
● Security Requirements for Dependable Systems ● Secure Systems Design Principles ● Security Testing and Assurance Techniques ● Common vulnerabilities and attack vectors

● The Relationship Between Security and

Comprehend the importance of software maintenance and evolution in the context of a dependable system.

(4 hours)

● Managing changes and updates in a dependable system
● Handling dependencies and ensuring consistency ● Reliability Maintenance over time

4. Methods of Instruction ▪ Lectures:

Unit 9:Software Maintenance and Evolution-

o Lectures will cover core theoretical concepts and provide foundational knowledge on Software reliability and availability. They will be interactive, using multimedia presentations, real-world examples, and case studies to illustrate key points. Lectures will encourage questions and discussions to deepen understanding.

▪ **Case Studies and Real-World Examples:**

o Case studies will be used to demonstrate how Software companies use Fault tolerance reliability test, systems to solve problems and achieve strategic goals. Students will analyze these cases, identify challenges, and propose solutions, bridging the gap between theory and practice..

▪ **Hands-on Labs and Practical Sessions:**

o Practical lab sessions will allow students to apply theoretical knowledge using various tools and technologies. For instance, students will set up virtual machines, implement basic cybersecurity measures, and Details is mentioned in Practical Session

5. List of Tutorials

The following tutorial activities of 15 hours per group of maximum 24 students should be conducted to cover all the required contents of this course.

S.N.	Tutorials
1	Introduction to dependability attributes: reliability, availability, safety, security, and maintainability
2	Differentiate between types of Markov Chains and their uses in dependability analysis
3	Case studies on system failures due to dependability issue
4	Implementing and testing error detection and correction mechanisms 5 Case studies: analysis of safety incidents in software systems
6	Designing secure and dependable systems: best practices
7	Fault Tree Analysis (FTA) and Failure Mode and Effects Analysis (FMEA) Using static and dynamic analysis tools for dependability assurance
8	Mutation Testing and Test Strategies(Stree, Robustness)
9	Proof of correctness, Model check and verification tools

6. **Practical tools and guidelines**

Practical Tools

1. Automated Testing tools
   Selenium
2. Continuous Integration/Continuous Deployment (CI/CD) Tools
   Jenkins:GitLab CI/CD:
3. Reliability and Performance Testing Tools
   Apache JMeter:.
4. Static and Dynamic Analysis Tools
   SonarQube and unit testing tools like Junit, pytest.
5. Security Testing Tools
   OWASP ZAP

Guidelines:

1. Hands-On Projects: Real-world project engagement and case studies to apply dependability principles.
2. Integration with CI/CD: Integrate testing and reliability tools into your CI/CD pipelines.
3. Use Realistic Scenarios: Test systems under realistic conditions and failure scenarios to ensure robustness.
4. Monitor and Analyze: Continuous monitoring system performance and analyzing test results to identify and address potential issues.

Course Project Presentations

- Review of all key concepts covered in the course
- Presentation of final projects by students, demonstrating their understanding of software dependability
- Feedback and discussion on project work

7. **Evaluation system and Students' Responsibilities**

Evaluation System

In addition to the formal exam(s) conducted by the Office of the Controller of Examination of Pokhara University, the internal evaluation of a student may consist of class attendance, class participation, quizzes, assignments, presentations, written exams, etc. The tabular presentation of the evaluation system is as follows.

Internal Evaluation	**Weight Marks**	**External Evaluation**	**Marks**
Theory	30		
Attendance & Class Participation 10% Assignments 20% Presentations/Quizzes 10% Internal Assessment 60%			
Practical	20	Attendance & Class Participation 10% Lab Report/Project Report 20% Practical Exam 20% Project Work 40% Viva 10%	Semester-End examination 50
	Total Internal 50	Full Marks: 50 + 50 = 100	
	Students' Responsibilities:		

Each student must secure at least 45% marks in the internal evaluation with 80% attendance in the class to appear in the Semester End Examination. Failing to obtain such score will be given

NOT QUALIFIED (NQ) and the student will not be eligible to appear in the End-Term examinations. Students are advised to attend all the classes and complete all the assignments within the specified time period. If a student does not attend the class(es), it is his/her sole responsibility to cover the topic(s) taught during the period. If a student fails to attend a formal exam, quiz, test, etc. there won't be any provision for a re-exam.

8. **Prescribed Books and References**

Text Book

- Sommerville, I. (2016). *Software Engineering* (10th ed.). Pearson.

Reference Books

- Musa, J. D. (1999). *Software reliability engineering*. McGraw-Hill.
- De Paoli, F., Omicini, A., & Ricci, A. (2013). *Engineering dependable software systems*. IOS Press.
- Lyu, M. R. (Ed.). (1996). *Handbook of software reliability engineering*. McGraw-Hill

5. List of Tutorials

The following tutorial activities of 15 hours per group of maximum 24 students should be conducted to cover all the required contents of this course.

S.N.	Tutorials
1	Introduction to dependability attributes: reliability, availability, safety, security, and maintainability
2	Differentiate between types of Markov Chains and their uses in dependability analysis
3	Case studies on system failures due to dependability issue
4	Implementing and testing error detection and correction mechanisms
5	Case studies: analysis of safety incidents in software systems
6	Designing secure and dependable systems: best practices
7	Fault Tree Analysis (FTA) and Failure Mode and Effects Analysis (FMEA) Using static and dynamic analysis tools for dependability assurance
8	Mutation Testing and Test Strategies(Stree, Robustness)
9	Proof of correctness, Model check and verification tools

1.

Practical tools and guidelines

2.

Best Practices for Practical Application

1.

Hands-On Projects: Real-world project engagement and case studies to apply dependability principles.

2.
3.

Integration with CI/CD: Integrate testing and reliability tools into your CI/CD pipelines.

4.
5.

Use Realistic Scenarios: Test systems under realistic conditions and failure scenarios to ensure robustness.

6.
7.

Monitor and Analyze: Continuous monitoring system performance and analyzing test results to identify and address potential issues.

8.

1.

Automated testing tools

2.

1.1 Unit Testing Tools,
JUnit, pytest
1.2 Integration Testing Tools
Postman
1.3 Functional Testing Tools
Selenium
1.4 Performance Testing Tools:
Apache JMeter

1.

Integrating Software Safety with CI/CD tools

2.

Jenkins:
Description: An open-source automation server used for continuous integration and continuous delivery.
GitLab CI/CD:

1.

Reliability and Performance Testing Tools

2.

Apache JMeter:
Description: A tool for performance and load testing, useful for testing system reliability under load.