Bachelor of Computer Applications (BCA)

SOFTWARE ENGINEERING LAB (DBCACO308P24)

Self-Learning Material (SEM III)

Jaipur National University

Centre for Distance and Online Education

Established by Government of Rajasthan
Approved by UGC under Sec 2(f) of UGC ACT 1956
&
NAAC A+ Accredited

TABLE OF CONTENTS

Course Introduction i

Experiment 1
Requirements Specification Document for an Online Bookstore 01

Experiment 2
Use Case Diagram for a Public Library System 02

Experiment 3
Design a Class Diagram 02

Experiment 4
Software Testing Plan 03

Experiment 5
Agile Project Management Simulation 03

Experiment 6
Database Design for E-commerce Website 04

Experiment 7
Software Architecture Document 04

Experiment 8
User Interface Design 05

Experiment 9
Performance Optimization Plan 05

Experiment 10
System Integration Scenario 06

Experiment 11
Software Maintenance Plan 06

Experiment 12
Ethical Hacking Exercise 07

Experiment 13
Code Refactoring Exercise 07

Experiment 14
Disaster Recovery Plan 08

Experiment 15
Mobile Application Development 08

Experiment 16 Security Audit for a Web Application	09
Experiment 17 Scalability Testing	09
Experiment 18 DevOps Pipeline Setup	10
Experiment 19 AI-Powered Feature Implementation	11
Experiment 20 Legacy System Modernization	11

COURSE INTRODUCTION

This course on Software Engineering offers a comprehensive exploration of software development processes, methodologies, and management techniques. It is structured to provide students with a robust understanding of the paradigms and life cycle models critical to software development, including waterfall, incremental, spiral, evolutionary, prototyping, and object-oriented models. These foundational concepts establish the framework for studying the various phases of software development, from conception through to testing and maintenance.

The curriculum delves into the specifics of software requirements, emphasizing both functional and non-functional aspects. Students will engage with the requirement engineering process, learning about feasibility studies, requirement validation and management, as well as various prototyping techniques including rapid and user interface prototyping. Additionally, the course covers software document analysis and modeling, which includes data, functional, and behavioral models, structured analysis, and the creation of data dictionaries.

In terms of design, the course introduces key concepts and principles of software design, focusing on modular design, design heuristics, and architectural considerations. Students will explore different design models, architectural and data design, and specific strategies for real-time software design, including data acquisition systems.

The course also addresses critical aspects of Software Configuration Management (SCM) and Software Project Management, providing insights into version and change control, configuration audits, and SCM standards. It emphasizes the importance of measurement and estimations in managing software projects, including empirical estimation models and project scheduling techniques.

Testing methodologies form another core component of the curriculum, with an in-depth look at various testing levels, activities, types, and strategies, including black-box, structural, and regression testing. The course culminates with a discussion on contemporary trends in Software Engineering, such as reverse engineering and re-engineering, and the practical application of CASE tools through case studies.

Overall, this course equips students with the necessary skills and knowledge to effectively manage and execute software projects, ensuring they are well-prepared for careers in software development and engineering.

Course Outcomes:

At the completion of the course, a student will be able to:

1. Compare between traditional ad-hoc methods and SDLC based approach of software development.
2. Understand different theories, models, and techniques related to SDLC.
3. Apply the software engineering lifecycle for different projects by demonstrating competence in communication, planning, analysis, design, construction, and deployment
4. An ability to work in one or more significant application domains. Work as an individual and as part of a multidisciplinary team to develop and deliver quality software.
5. Developing efficient software using the latest tools and techniques. Use of computer aided designing and automated testing tools.

Acknowledgements:

The content we have utilized is solely educational in nature. The copyright proprietors of the materials reproduced in this book have been tracked down as much as possible. The editors apologize for any violation that may have happened, and they will be happy to rectify any such material in later versions of this book.

Question 1: Requirements Specification Document for an Online Bookstore

Program Statement: Develop a comprehensive Requirements Specification Document for an online bookstore. The system should allow users to create accounts, browse books by category, add books to a shopping cart, and complete purchases. The system should also provide administrative functionalities such as adding new books, managing inventory, and viewing sales reports.

Detailed Requirements:

- **User Account Management:** Registration, login, password recovery, and user profile management.
- **Book Browsing:** Ability to search for books by title, author, or ISBN, and filter results by genre, price, and publication date.
- **Shopping Cart:** Users should be able to add items to the cart, modify quantities, and remove items.
- **Checkout Process:** Integration with a payment gateway for processing credit card transactions, and order confirmation via email.
- **Administrative Functions:** Secure login for administrators, functionalities to add or remove books, update book details, and access to a dashboard that displays real-time data on sales and inventory levels.

Solution Hints: Start by defining the scope of the system and identifying all stakeholders. Use tools like user interviews and competitive analysis to gather requirements. Document the functional requirements clearly with use cases or user stories. For non-functional requirements, consider aspects like scalability, security (e.g., use of HTTPS, data encryption), and compliance with data protection regulations. Use diagrams where applicable, such as entity-relationship diagrams for database design and sequence diagrams to illustrate processes.

Question 2: Use Case Diagram for a Public Library System

Program Statement: Design a use case diagram that captures the interactions between librarians, members, and the library system. The system should support book lending, book returns, member registration, and fine payment.

Detailed Requirements:

- **Member Registration:** Capture member details and issue library cards.
- **Book Lending:** Check out books to members, including due date assignment and availability checks.
- **Book Returns:** Handle the return of borrowed books, including condition checks and fine calculations for late returns.
- **Fine Payment:** Allow members to pay fines for late returns or damaged books.
- **System Maintenance:** Regular updates to the book catalog, managing member records, and generating reports on book loans and returns.

Solution Hints: Begin with identifying the primary actors (members, librarians) and secondary actors (online payment systems, email systems for notifications). Define the main interactions these actors have with the system. For each use case, write a brief description that includes the main success scenario and any alternative flows. Consider the conditions under which exceptions might occur and how the system should respond. Diagrammatically represent these interactions in a use case diagram using UML notation. Focus on clarity and completeness to ensure that all user interactions with the system are accurately represented.

Question 3: Design a Class Diagram

Program Statement: Create a class diagram for a ride-sharing application that includes user management, ride booking, and payment processing.

Detailed Requirements:

- **User Management:** Users can be riders or drivers. Include attributes like name, contact details, and user ratings.

- **Ride Booking:** Ability to request a ride, assign a driver, and calculate the fare based on distance.
- **Payment Processing:** Support for multiple payment methods, including credit card and mobile wallet.

Solution Hints: Identify the main classes and their relationships. Define methods and attributes for each class that capture the essential functionalities. Consider using inheritance where applicable, such as a base class for users from which riders and drivers can derive.

Question 4: Software Testing Plan

Program Statement: Develop a detailed testing plan for a new social media platform that includes functional testing, integration testing, and usability testing.

Detailed Requirements:

- **Functional Testing:** Test individual features like posting updates, liking posts, and adding friends.
- **Integration Testing:** Ensure that different modules (e.g., user interface, database) work together as expected.
- **Usability Testing:** Evaluate the platform's ease of use with target audience groups.

Solution Hints: Outline different testing phases and the techniques to be used. For each type of testing, specify the scope, objectives, and methods. Prepare test cases that cover various user scenarios and possible edge cases.

Question 5: Agile Project Management Simulation

Program Statement: Simulate an agile project management environment where students must plan and execute a sprint for a hypothetical software project.

Detailed Requirements:

- **Sprint Planning:** Define sprint goals and tasks based on the product backlog.

- **Daily Stand-ups:** Organize daily meetings to track progress and address impediments.
- **Sprint Review:** Conduct a meeting at the end of the sprint to demonstrate completed work.

Solution Hints: Use agile methodologies like Scrum or Kanban. Emphasize the importance of communication, flexibility, and continuous improvement. Encourage students to use tools like JIRA or Trello for task management.

Question 6: Database Design for E-commerce Website

Program Statement: Design a database schema for an e-commerce website that handles products, orders, customers, and shipping.

Detailed Requirements:

- **Products:** Include tables for product information, categories, and inventory levels.
- **Orders:** Design tables to handle order placement, order status, and payment history.
- **Customers:** Maintain customer profiles, including shipping addresses and order history.
- **Shipping:** Include logistics information, such as shipping methods, costs, and delivery status.

Solution Hints: Focus on creating a normalized database schema to avoid redundancy and ensure data integrity. Use foreign keys to establish relationships between tables. Consider aspects like scalability and security for handling sensitive information.

Question 7: Software Architecture Document

Program Statement: Create a software architecture document for a health monitoring system that connects patients, doctors, and medical devices.

Detailed Requirements:

- **System Overview:** Describe the system’s structure and its main components.

- **High-Level Components:** Detail components such as user interfaces, databases, and external integrations.
- **Communication:** Explain how components communicate, including any APIs or protocols used.

Solution Hints: Use architectural views to describe different aspects of the system, such as logical, development, and deployment views. Include diagrams to illustrate complex concepts and interactions.

Question 8: User Interface Design

Program Statement: Design the user interface for a mobile app that allows users to book fitness classes, track progress, and connect with personal trainers.

Detailed Requirements:

- **Booking System:** Easy navigation for class schedules and booking options.
- **Progress Tracker:** Visual representations of user progress, such as graphs and statistics.
- **Personal Trainer Connection:** Features to message trainers, view trainer profiles, and schedule sessions.

Solution Hints: Focus on usability and aesthetic appeal. Use tools like Adobe XD or Sketch for prototyping. Consider user feedback to refine the design, ensuring it’s intuitive and engaging.

Question 9: Performance Optimization Plan

Program Statement: Develop a plan to optimize the performance of a large-scale web application focusing on load times and server response times.

Detailed Requirements:

- **Frontend Optimization:** Techniques such as minifying JavaScript and CSS, optimizing images, and using asynchronous loading.

- **Backend Optimization:** Database indexing, query optimization, and efficient use of caching.
- **Monitoring:** Tools and practices to monitor performance and identify bottlenecks.

Solution Hints: Establish performance benchmarks and identify key performance indicators (KPIs). Use tools like Google Lighthouse or WebPageTest for frontend analysis and database profiling tools for backend optimizations.

Question 10: System Integration Scenario

Program Statement: Outline a scenario where students must integrate an existing inventory management system with a new online sales platform.

Detailed Requirements:

- **API Usage:** Define how the systems will communicate via APIs, including necessary endpoints.
- **Data Synchronization:** Ensure consistent data across systems, handling conflicts and duplicates.
- **Error Handling:** Develop strategies to manage errors and exceptions during integration.

Solution Hints: Use middleware or an ESB (Enterprise Service Bus) for integration. Focus on data flow diagrams and sequence diagrams to visualize the integration process. Consider security aspects, particularly in API authentication and authorization.

Question 11: Software Maintenance Plan

Program Statement: Prepare a maintenance plan for a legacy software system that needs updating and optimization to meet current technological standards.

Detailed Requirements:

- **Assessment:** Conduct a thorough assessment of the current system to identify areas for improvement.

- **Update Strategy:** Plan for gradual updates without disrupting the existing user base.
- **Documentation:** Update all documentation to reflect changes and provide training for users on new features.

Solution Hints: Prioritize updates based on urgency and impact. Use version control systems to manage changes and test updates in a staging environment before deployment.

Question 12: Ethical Hacking Exercise

Program Statement: Conduct an ethical hacking exercise to identify vulnerabilities in a simulated corporate network.

Detailed Requirements:

- **Vulnerability Scanning:** Use tools like Nmap or Nessus to scan the network for vulnerabilities.
- **Penetration Testing:** Attempt to exploit identified vulnerabilities to assess the impact.
- **Report and Fix:** Prepare a detailed report of findings and suggest measures to mitigate risks.

Solution Hints: Focus on common vulnerabilities like SQL injection, XSS, and buffer overflows. Use a systematic approach to document each test and its outcomes. Ensure that all testing is authorized and within ethical boundaries.

Question 13: Code Refactoring Exercise

Program Statement: Refractor a given piece of poorly written software code to improve readability, maintainability, and performance.

Detailed Requirements:

- **Identify Issues:**Analyze the code to identify bad practices such as code duplication, long methods, and magic numbers.

- **Refactoring Techniques:** Apply techniques like extracting methods, renaming variables, and reducing dependencies.
- **Testing:** Ensure the refactored code passes all existing tests and behaves as expected.

Solution Hints: Use code analysis tools to help identify problem areas. Maintain a test-driven approach to ensure that changes do not affect the functionality. Document changes for future reference.

Question 14: Disaster Recovery Plan

Program Statement: Develop a comprehensive disaster recovery plan for an IT infrastructure that supports critical business operations.

Detailed Requirements:

- **Risk Assessment:** Identify potential threats and vulnerabilities that could impact business continuity.
- **Recovery Strategies:** Outline strategies for data backup, system recovery, and alternative work arrangements.
- **Plan Testing:** Describe the procedures for testing the plan to ensure its effectiveness.

Solution Hints: Include details on recovery point objectives (RPO) and recovery time objectives (RTO). Use scenarios to illustrate how the plan would be executed in different types of disasters. Regularly update the plan to incorporate new technologies and changes in the business environment.

Question 15: Mobile Application Development

Program Statement: Develop a plan for creating a mobile application that assists users with public transportation, including schedules, ticket booking, and real-time updates.

Detailed Requirements:

- **User Interface:** Design an intuitive interface that provides easy access to all functionalities.

- **Functionality:** Include features for viewing schedules, booking tickets, and receiving notifications about delays or changes.
- **Integration:** Ensure the app integrates with existing transportation APIs for real-time data.

Solution Hints: Use mobile development frameworks like React Native for cross-platform compatibility. Focus on user experience design to ensure the app is accessible and easy to use. Test extensively on different devices to ensure functionality and performance.

Question 16: Security Audit for a Web Application

Program Statement: Conduct a security audit on a web application to identify security threats and vulnerabilities. Propose mitigation strategies for each identified issue.

Detailed Requirements:

- **Threat Identification:** Scan the application for common vulnerabilities such as SQL injection, CSRF, and XSS.
- **Security Assessment:** Evaluate authentication, authorization, and encryption mechanisms.
- **Mitigation Strategies:** Develop detailed plans to address and mitigate each vulnerability.

Solution Hints: Use automated security scanning tools to identify vulnerabilities and manual testing to confirm and explore these issues further. Discuss the principles of secure coding practices and the importance of regular security updates and patches. For each vulnerability, provide a theoretical mitigation strategy that could include both code fixes and changes to deployment environments.

Question 17: Scalability Testing

Program Statement: Plan and execute scalability testing for a high-traffic e-commerce site to ensure it can handle increased loads, particularly during peak shopping periods.

Detailed Requirements:

- **Load Testing:** Simulate varying loads on the server to measure response times and system behavior under stress.
- **Scalability Strategies:** Evaluate and propose scaling strategies, such as horizontal scaling, vertical scaling, and the use of load balancers.
- **Performance Metrics:** Define and measure key performance indicators such as page load times, transaction completion rate, and system downtime.

Solution Hints: Use tools like JMeter or LoadRunner for load testing. Discuss the concept of cloud scalability with services like AWS Elastic Load Balancing or Google Cloud Load Balancing. Analyze the results to recommend infrastructure adjustments and software optimizations that help manage larger user volumes efficiently.

Question 18: DevOps Pipeline Setup

Program Statement: Design and set up a DevOps pipeline for continuous integration and continuous deployment (CI/CD) for a multi-service application.

Detailed Requirements:

- **Tool Selection:** Choose appropriate tools for version control, build automation, testing, and deployment.
- **Pipeline Configuration:** Configure the pipeline stages, including code checkout, build, test automation, and deployment.
- **Monitoring and Feedback:** Integrate monitoring tools and set up feedback mechanisms for ongoing improvement.

Solution Hints: Focus on integrating tools like Git for version control, Jenkins for CI/CD, Selenium for automated testing, and Docker for containerization. Highlight the importance of feedback loops using monitoring tools such as Prometheus or Grafana for system metrics. Discuss best practices in CI/CD to minimize deployment risks and improve release quality.

Question 19: AI-Powered Feature Implementation

Program Statement: Implement an AI-powered feature in an existing application, such as a recommendation system for a shopping site or a chatbot for customer service.

Detailed Requirements:

- **AI Model Choice:** Choose and justify an appropriate AI model based on the feature's requirements.
- **Integration Approach:** Detail the steps for integrating the AI model into the existing system architecture.
- **Testing and Evaluation:** Outline methods for testing the AI feature and evaluating its performance against predefined metrics.

Solution Hints: Discuss different machine learning models like neural networks for a recommendation system or NLP models for a chatbot. Use APIs like TensorFlow or PyTorch for model development and integration. Focus on the importance of data quality, model training, and fine-tuning. Include validation techniques to measure the effectiveness of the AI feature in real-world scenarios.

Question 20: Legacy System Modernization

Program Statement: Develop a detailed plan for modernizing a legacy banking system to improve performance, security, and user experience.

Detailed Requirements:

- **System Assessment:** Analyze the current system to identify bottlenecks, security flaws, and areas needing UX improvement.
- **Modernization Strategy:** Choose between re-platforming, refactoring, or replacing components of the system.
- **Implementation Roadmap:** Create a phased roadmap for the modernization process, including risk assessment and mitigation.

Solution Hints: Evaluate the benefits and drawbacks of different modernization strategies, considering factors such as cost, time, and impact on current operations. Discuss the use of modern technologies like microservices architecture, cloud computing, and API-first design. Plan for thorough testing phases to ensure the new system meets all functional and non-functional requirements without disrupting existing services.