1. Define Software Engineering:

* Software Engineering is a systematic approach to the development, operation, maintenance and improvement of software systems.

1. What is software engineering, and how does it differ from traditional programming?

* Software is a structured approached to software development that focuses on systemizing and optimizing software development to ensure high quality reliable and maintainable software products while traditional programming is focused primarily writing code to a particular problem.

1. Software Development Life Cycle (SDLC):

* It is a structured framework that defines the stages involved in the development of software system.

1. Explain the various phases of the Software Development Life Cycle. Provide a brief description of each phase:

* Planning: Define project objectives and scope.
* Gather requirements and analyze user requirements.
* Design software architecture and components and interfaces.
* Code the software based on the design.
* Conduct testing.
* Monitor and maintain the system, fix bugs.

1. Agile vs. Waterfall Models:

* Agile model is a flexible and more iterative approach to software development often preferred to projects where requirements are likely to change while waterfall model is a traditional linear approach to software development suitable for projects that have well defined and suitable requirements.

1. Compare and contrast the Agile and Waterfall models of software development. What are the key differences, and in what scenarios might each be preferred?

Key differences include:

Agile model: Iterative development.

Short sprints with frequent feedback.

Active collaboration.

Shorter development cycles

Waterfall model: Linear sequential development.

Multi-phase, detailed planning.

Little involvement.

Longer development cycles.

Preferred scenarios:

Agile: complex and rapidly changing environment.

Need for flexibility and feedback.

Frequent releases of updated software.

Waterfall: Well defined and stable requirements.

Low risks of significant changes.

Large development teams.

1. What is requirements engineering? Describe the process and its importance in the software development lifecycle. Software Design Principles:

* Requirements engineering is process of eliciting, analyzing, documenting, and managing requirements for a software system.

Process of requirements engineering:

* Eliciting requirements: Gathering requirements from stakeholders
* Analyzing requirements: Analyzing and prioritizing requirements to understand their impact on the software system and to identify any conflicts or inconsistencies.
* Documenting requirements: Documenting requirements in a clear and unambiguous manner using formal notations.
* Managing requirements: Managing changes to requirements throughout the software development lifecycle.

Importance of requirements engineering:

* Enhances communication
* Guides development.
* Minimizes risks.

1. Explain the concept of modularity in software design. How does it improve maintainability and scalability of software systems? Testing in Software Engineering:

* Modularity in software design is a concept that involves breaking down a software into smaller independent modules or components.
* Modularity increases maintainability and scalability of software systems in the following ways: Isolation of changes, ease of modification, reusability and scalability.
* Testing is a practice that involves evaluating software products to identify defects, errors and deviations from expected behaviours.

1. Describe the different levels of software testing (unit testing, integration testing, system testing, acceptance testing). Why is testing crucial in software development? Version Control Systems:

* Unit testing: Testing individual units or components of the software in isolation to verify their correctness and functionality.
* System testing: Testing the complete software system to assess its compliance with specified requirements.
* Integration testing: Testing the interactions between integrated modules or components to validate the behavior of the system as whole.
* Acceptance testing: Involving end-users to validate that the software meets their requirements and expectations before deployment.

1. What are version control systems, and why are they important in software development? Give examples of popular version control systems and their features. Software Project Management:

* Version control systems are software tools that track and manages changes to files and code over time.

Examples of version control systems:

* Git: It is a distributed version of control system known for its speed, flexibility and strong merging capabilities. Enables developers to work offline.
* Subversion: Subversion is a centralized version control system that tracks changes to files and directories. It allows for versioned directories, atomic commits, and efficient branching.

1. Discuss the role of a software project manager. What are some key responsibilities and challenges faced in managing software projects? Software Maintenance:

* Software project management involves planning, organizing and controlling of resources and tasks to achieve specific software development goals.

Role of a software project manager include:

* Defining project scope, objectives and timelines and resource requirements for project success.
* Manage resources.
* Budget management.
* Identify potential risks and implement mitigation strategies.

Challenges faced in managing software projects:

* Tendency for project requirements to expand beyond initial expectations leading to delays and budget overruns
* Dealing with complex software systems.
* Ensuring effective team communication among team members to achieve project goal.

1. Define software maintenance and explain the different types of maintenance activities. Why is maintenance an essential part of the software lifecycle? Ethical Considerations in Software Engineering

* Software maintenance refers to the activities involved in modifying, updating and enhancing software systems after their initial development and deployment.

Types of maintenance activities:

* Corrective maintenance: Fixing bug errors, and defects in software to improve its performance and usability.
* Adaptive maintenance: Modify software to adapt to changes in the environment such as operating system upgrades.
* **Perfective Maintenance:** Enhancing the software's functionality, performance, and usability based on user feedback or evolving requirements.
* **Preventive Maintenance:** Proactively identifying and addressing potential issues to prevent future problems and optimize software performance.

Maintenance is an essential part of the software lifecycle because:

* Extends lifespan of software system.
* Enhances user satisfaction.
* It is cost effective.

1. What are some ethical issues that software engineers might face? How can software engineers ensure they adhere to ethical standards in their work? Submission Guidelines: Your answers should be well-structured, concise, and to the point. Provide real-world examples or case studies wherever possible. Cite any references or sources you use in your answers.

* Privacy concerns.
* Bias in algorithms: Ensuring algorithm is fair and unbiased in decision-making process.
* Cybersecurity: Building secure software systems to prevent data breaches and cyber threats.

Software engineers can adhere to ethical standards in their work by:

1. **Following established codes of ethics:** Abiding by professional codes of conduct, such as those from IEEE or ACM.
2. **Transparency:** Being transparent about software functionality, data usage, and potential risks to users.
3. **Continuous learning:** Staying informed about ethical issues in software engineering and incorporating ethical considerations into the design and development process.

Real-world example: The Cambridge Analytica scandal, where Facebook user data was improperly harvested and used for political profiling, highlights the importance of ethical data handling practices in software development.