1. Define Software Engineering:

Answer: Software engineering is the process of designing, building, and maintaining software systems that helps people solve problems and achieve their goals.

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

Software engineering is a technical engineering activity that mainly focuses on designing, developing and maintaining software products.

Traditional engineering is mainly focused on writing code to solve specific problems.

1. Explain the various phases of the Software Development Life Cycle. Provide a brief description of each phase.
2. Requirements: Gathering and documenting user needs and system requirements.
3. Design: Creating high-level and detailed designs of the software architecture and user interface.
4. Implementation: writing code and building the software according to the design specifications.
5. Testing: Conducting various tests to ensure the software meets quality standards and functional requirements.
6. Deployment: Releasing the software to users.
7. Maintenance: providing ongoing support, updates, and enhancements to the software after deployment.
8. Compare and contrast the Agile and Waterfall models of software development. What re the key differences, and in what scenarios might each be preferred?

Agile is flexible and iterative, emphasizing continuous feedback and adaptation while Waterfall is sequential and rigid, with distinct phases completed in order.

Agile promotes ongoing collaboration while Waterfall focuses on thorough planning and execution in predetermined stages.

Agile in dynamic environments or projects with evolving requirements. It suitable for startups, innovative products and continuous improvements initiatives.

Waterfall is used in projects with well-defined requirements and strict budgets or deadlines.

*Source: forecast.app/blog*

1. What is requirement engineering? Describe the process and its importance in the software development lifecycle.

This is the process of defining, documenting and maintaining requirements in the engineering design process. *Source: en.wikipedia.org*

Its importance: to identify, analyze, document, and validate the needs of software users, providing a roadmap for the development team and ensuring everyone works towards a common goal. *Source: typeser.io*

The process:

1. Requirement gathering: identifying stakeholders, collecting and documenting the requirements.
2. Requirement analysis: study the requirements, identify functional and non-functional requirements and create a requirement model.
3. Requirement satisfaction: write the software requirements specification, include functional and non-functional requirements and define the acceptance criteria.
4. Requirement validation: review the software requirements specification, verify the requirements and obtain stakeholder approval.
5. Requirement management: track changes and communicate changes.
6. Explain the concept of modularity in software design. How does it improve maintainability and scalability of software systems?

Modularity is a technique where complex software is divided into smaller, independent modules, such as functions, classes or components. It facilitates easier management and understanding of complex systems by breaking them down into digestible parts.

*Source: secoda.co*

1. Describe the different levels of software testing (unit testing, integration testing, system testing, acceptance testing). Why is testing crucial in software development?
2. Unit testing: testing individual components or modules of software.
3. Integration testing: testing interactions between different components or subsystems.
4. System testing: testing the entire software system as a whole.
5. Acceptance testing: testing the software against user requirements to ensure it meets user needs.

Software testing helps to improve the quality of software products. It’s a key component of quality assurance, and helps to ensure that software products meet requirements, function correctly and perform reliably.

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

Version control systems are software tools that help software teams manage changes to source code over time. They help software teams work faster and smarter.

Examples of popular version control systems: Git, GitHub, GitLab, AWS CodeCommit.

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

Software project manager: oversees the planning, execution and delivery of software projects. They align project objectives with business goals, manage resources and mitigate risks.

Key responsibilities: planning, developing the project idea, creating and leading a team, setting project deadlines, monitoring deadlines and ensuring stakeholder satisfaction.

Challenges faced: misalignment between goals and business objectives, lack of accountability, resource allocation, poor planning and unrealistic deadlines.

1. Define software maintenance and explain the different types of maintenance activities.

Software maintenance is the process of modifying, changing, and updating a software system or module to resolve errors, improve performance or adapt to a changing environment.

The different types of software maintenance:

1. Corrective software maintenance: it addresses the errors and faults within a software that could impact various parts of your software.
2. Adaptive software maintenance: the process of modifying software to keep it usable in a changing environment.
3. Perfective software maintenance: focuses on the evolution of requirements and features that existing in your system.
4. Preventive software maintenance: is a proactive measure to identifying and addressing potential issues in software before they become major problems.
5. Why is maintenance an essential part of the software lifecycle?

It enhances the software performance by eliminating errors, removing unusable development and implementing advanced development strategies.

1. Ethical considerations in Software Engineering.
2. Respecting user privacy and data security: it is the duty of the developer to protect the users’ data.
3. Ensuring accessibility and inclusivity: developers should make sure that their software is available to all users.
4. Promoting ethical business practices.