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

Software engineering is the branch of computer science that deals with the design, development, testing and maintenance of software application. It is an engineering approach to software development.
Traditional programming refers to the conventional approach of writing code to create specific instructions for a computer to follow. Software programming is different from traditional programming in that it uses a data driven approach, while traditional programming is typically rule based and deterministic.
Software programming is intuitive and user friendly, it is easy to use, easy to understand and navigate. Software developers can add new features and update existing features with new technologies.

Software development life cycle:
Explain the various phases of the software development life cycle. Provide a brief description of each phase. Agile vs waterfall models.

A full software development life cycle has seven basic stages; Planning, Requirements, Design and Prototype, Software development, Testing, Deployment, and Maintenance.

Planning is the stage where you are gathering business requirements from your clients or stakeholders. It is the phase when the project plan is developed, it is the stage that identifies, prioritizes, and assigns the tasks and resources required to build the structure for a project.

The requirement phase in the software development life cycle includes the scope of work necessary to define, analyze and document business and end-user requirements. It is the process of creating, maintaining, evolving, implementing and delivering the requirements artifacts throughout the life cycle of a system or project.

A prototype in software development is a simulation of how a real product will work and feel. It is used for design feedback and user testing. The design phase involves transforming the software requirements gathered during the requirements phase into a structured design document a. The design and prototype phase is to allow users of the software to evaluate developers proposals for the design of the eventual product by actually trying them out, rather than having to interprete and evaluate the design based on descriptions.

The software development stage is the actual development phas. It is where the development team members divide the project into software modules and turn the software requirements into code that makes the product.

Testing: Before getting the software product out the door to the production environment, tests are carried out on it to discern any issues that need to be corrected and whether it is good enough to be released to the public.

The deployment phase is the final step in the software development life cycle and delivers the final product to the customer in a live production environment. After the product deploys, it is ready for customers to use.

The maintenance phase is where the software is monitored to ensure it continues to function as it was designed to, and repairs or upgrades are performed as needed.

These stages are essentially the same in all models of software development.
The agile model arranges the software development life cycle phases into several development cycles, with the team delivering small, incremental software changes in each cycle. The agile software development life cycle is a methodology that relies on cooperative decision making and product development in short cycles or sprint mode.

The waterfall model is a linear step by step approach that is ideal for projects with a clear scope and a predictable timeline. It is a straightforward process usually in a downward spiral (like a waterfall). It involves rigorous planning upfront to ensure that the project stays on track, with progress tracked closely and issues addressed promptly.
Waterfall is a better method when a project must meet strict regulations as it requires deliverables for each phase before proceeding to the next one. Alternatively, Agile is better suited for teams that plan on moving fast, experimenting with direction and don't know how the final project will look before they start.

Compare and contrast the agile and waterfall models of software development. What are the key differences and in what scenarios might each be preferred;

The waterfall phase is a breakdown of development activities into linear sequential phases, this means that they are passed down onto each other. Each phase depends on the deliverables of the previous one and corresponds to a specialization of task. The approach tends to be among the less iterative and flexible approach as progress flows largely in one direction (downwards like a waterfall)through the phases of the software development life cycle.

The agile model of software development is the iterative approach. It tells everyone involved with the project what needs to be done. In that gile method instead of delivering a large project only when all parts are complete, a team breaks it down into smaller parts, and delivers these completed smaller parts in regular cycle.

The agile and waterfall models both go through the same software development life cycles, the only difference is the way in which they go through the process.

Contrast:

* Waterfall strictly assigns roles to project team members, with specific duties and responsibilities defined for each team member. In contrast, the agile model empowers team members to collaborate on different aspects of the project over time, leading to a more self-organizing team structure.
* In waterfall, planning is a linear process done at the beginning of the project, with all requirements and objectives laid out in detail upfront. In contrast, agile planning is a continuous process throughout the project's life cycle, with adjustments made as new information or requirements emerge.
*The waterfall methodology generally discourages changes to the project's scope, even with change requests used correctly. This is because the methodology requires an extensive amount of time spent in the beginning trying to get the plan right, which can make changes more costly after the project has begun. On the other hand, agile is more adaptable to changes in scope, with the development team able to adjust quickly as requirements change.
*The waterfall method is designed for long-term projects with predetermined timelines. The project is completed linearly, with each phase dependent on the previous one. Agile, however, uses short iterations to deliver value rapidly, allowing teams to adjust plans over time and achieve shorter time frames.
*Waterfall projects tend to take longer because all requirements must be agreed upon before development can begin. Agile projects, on the other hand, are usually delivered more rapidly than waterfall projects due to the iterative development cycles used in agile.
* Agile allows for quick delivery of projects with shorter lifecycles, as each iteration delivers a workable product. Waterfall requires the completion of all tasks before any work can be released.
* Agile encourages teams to respond quickly and adaptively to changes during the development process. Waterfall is less flexible and resistant to change once the project's scope has been defined.
*Testing is essential to the agile and waterfall methodologies, but the approaches differ significantly. Agile emphasizes incremental testing to identify and resolve issues throughout the development process. In waterfall, testing is usually done at specific milestones, often towards the end of the project.
*Agile relies on minimal documentation, focusing on self-organizing teams and collaboration. Waterfall, in contrast, relies heavily on documenting each step in detail to ensure that all team members are on the same page.
* Agile emphasizes informal communication, with frequent interactions between individuals or small groups of stakeholders. In waterfall, communication is more formal, with detailed communication plans and progress reports shared across multiple stakeholders.

What is requirements engineering?

Requirements engineering is the process of defining, documenting, and maintaining requirements in the engineering design process. It is a common role in systems engineering and software engineering.

Requirements engineering helps to ensure that the software system is delivered on time, within budget, and to the required quality standards. Provides a solid foundation for the development process, which helps to reduce the risk of failure. This is when the project team begins to understand what the customer wants from the project.

Modularity:

Modularity is a fundamental design principle in software engineering aimed at creating software in a way that minimizes dependencies among the components of a system. This helps to localize the impact of changes, simplifies maintenance, and enhances the understandability of the system. Modularity in software engineering refers to the design approach that emphasises the separation of concerns, where a complex software system is divided into smaller, loosely coupled modules.

Each module performs a specific function or handles a particular feature, and they interact through well-defined interfaces. This approach promotes a clear division of labour, allowing developers to focus on individual modules without being overwhelmed by the entire system's complexity.

Modularity greatly enhances scalability. By breaking down the software into independent modules, you can more easily adapt, extend, or replace parts of the system without affecting others. Modularity aims to improve software development by partitioning complex problems into more manageable sub-problems.

By breaking down a large software system into smaller modules, developers can focus on writing code that is easier to understand, test, and maintain.

*Modularity promotes code reusability,Well-defined modules can be easily reused in different projects, saving developers time and effort.

Instead of having to reinvent the wheel, they can leverage existing modules that have already been thoroughly tested and proven to work effectively.

This not only speeds up development but also improves the overall quality of the software

Testing:

Testing is the phase in the software development life cycle where the software developers test the project in order to detect if there are any faults with it and ensure that it is in perfect form before releasing it for customer use.Testing is an essential risk mitigation strategy. It identifies potential issues before they reach end-users, reducing the likelihood of software failures and the associated

financial and repetitional risks. Early detection and resolution of problems save time and resources in the long run

Unit testing is the process where you test the smallest functional unit of code.Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually scrutinized for proper operation. Software developers and sometimes QA staff complete unit tests during the development process.

Integration testing, also called integration and testing, abbreviated I&T, is a form of software testing in which multiple parts of a software system are tested as a group. Integration testing describes tests that are run at the integration-level to contrast testing at the unit or system level.

System testing, also referred to as system-level testing or system integration testing, is the process in which a quality assurance team evaluates how the various components of an application interact together in the full, integrated system or application. System Testing is a type of software testing that is performed on a completely integrated system to evaluate the compliance of the system with the corresponding requirements. In system testing, integration testing passed components are taken as input.

The goal of integration testing is to detect any irregularity between the units that are integrated. System testing detects defects within both the integrated units and the whole system. The result of system testing is the observed behavior of a component or a system when it is tested.
System Testing is carried out on the whole system in the context of either system requirement specifications or functional requirement specifications or the context of both. System testing tests the design and behavior of the system and also the expectations of the customer.
It is performed to test the system beyond the bounds mentioned in the software requirements specification (SRS). System Testing is performed by a testing team that is independent of the development team and helps to test the quality of the system impartial.
It has both functional and non-functional testing. System Testing is a black-box testing. System Testing is performed after the integration testing and before the acceptance testing.

Acceptance testing is software testing that evaluates whether a system meets its business and user requirements. It is the final testing stage before a system is released to production. The main purpose of acceptance testing is to verify that a system: Meets all of its functional and non-functional requirements.Acceptance testing includes the following phases: plan, test, record, compare and result.

Once the test is written according to the plan, end users interact with the software to gauge its usability. The software should meet expectations, as defined by the business in the requirements. When the tests return results, IT should report and fix any flaws that show up. If

the results match the acceptance criteria for each test case, the test will pass. But, if test cases exceed an unacceptable threshold, they will fail.

Types of acceptance testing;
Acceptance testing encompasses various types, including user acceptance and operational acceptance.

User acceptance testing, also called end-user testing, assesses if the software operates as expected by the target base of users. Users could mean internal employees or customers of a business or another group, depending on the project.

Operational acceptance testing reviews how a software product works. This type of testing ensures processes operate as expected and that staff can sufficiently use and maintain the system. Operational acceptance testing examines backups and disaster recovery, as well as maintainability, failover and security.

The software testing lifecycle is important because it helps to ensure the quality and reliability of web applications.

Version control:

Version control (also known as revision control, source control, and source code management) is the software engineering practice of controlling computer files and versions of files; primarily source code text files, but generally any type of file.

A version control system is a software tool that automates version control. Alternatively, version control is embedded as a feature of some systems such as word processors, spreadsheets, collaborative web docs] and content management systems.

Version control includes viewing old versions and enables reverting a file to a previous version.

Version control systems enable rapid collaboration and iteration. Developers can more easily create branches and merge their changes to the main branch. They can make changes to the codebase without blocking other team members and can resolve merge conflicts with fewer steps and less risk. One of the main benefits of a centralized Version control system is that every user has quick access to the latest version of a file.

Examples of version control systems:
Git
Mercurial
Subversion
Helix Core

Azure DevOps

Git:
With Git, developers can create multiple branches, allowing for independent lines of development. Git also facilitates collaborative workflows by providing a mechanism for remote repositories. Developers can clone a repository from a remote server, work on their local copy, and later synchronize their changes with the central repository. This enables multiple developers to work simultaneously, making it easier to manage conflicts that may arise when multiple people modify the same code or file. Git is super fast and it is efficient in its performance.

Mercurial:
Mercurial is known for its user-friendly interface and intuitive workflows, making it an appealing choice for developers. It has the capability for large repositories and offers robust support for Windows platforms. Moreover, Mercurial's high extensibility allows users to customize it according to their specific requirements, enhancing its versatility for different development environments.

Helix core:
Employing Helix core by Perforce as a version control solution has numerous benefits, including convenient access to Git workflows, high speed, impressive scalability, and easy change list tracking. It also offers straightforward identification of code modifications through different tools and integration with Visual Studio via its plugin.

Subversion:
SVN benefits from user-friendly GUI tools like TortoiseSVN, enhancing the overall user experience. It also supports the inclusion of empty directories, allowing for better organization and flexibility. In terms of compatibility, SVN provides stronger support for Windows environments compared to Git. It also integrates well with Windows, leading IDEs, and Agile tools. It it characterized by its simplicity and ease of use.

Azure Devops:
Azure Devops comes with comprehensive security and built-in compliance. It is characterized by seamless integration, good security and it is user friendly.

Software project management:
In software development, project managers ensure projects stay on track, meet client needs, and achieve organizational objectives. Their role involves defining scope, managing resources, promoting team communication, and mitigating risks. They convert client visions into actionable plans, maintaining adherence throughout development.

Project managers skillfully balance scope, time, and resources, efficiently allocating assets and preventing budget overruns. As communicators, they bridge technical teams and stakeholders, ensuring timely, budget-friendly, quality software solutions, proving their invaluable presence in the field.
Software project manager's responsibilities are to analyze project constraints, establish the project objectives, coordinate the project's internal and external teams, construct the project timelines and monitor the project's key performance indicators.

Some challenges faced in managing software projects include;
Resource allocation
Communication
Poor planning and unrealistic deadlines
And so many more.

Software maintenance:
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, it is modifying and updating software to keep up with consumer needs.
There are different types of software maintenance activities, namely;
Corrective software maintenance;
Adaptive software maintenance;
Perfective software maintenance; and
Preventative software maintenance.

*Corrective software maintenance addresses the errors and faults within software applications that could impact various parts of your software, including the design, logic, and code.
*Adaptive software maintenance becomes important when the environment of your software changes. This can be brought on by changes to the operating system, hardware, software dependencies, Cloud storage, or even changes within the operating system.
*Perfective software maintenance focuses on the evolution of requirements and features that existing in your system. As users interact with your applications, they may notice things that you did not or suggest new features that they would like as part of the software, which could become future projects or enhancements.
*Preventative Software Maintenance helps to make changes and adaptations to your software so that it can work for a longer period of time. The focus of the type of maintenance is to prevent the deterioration of your software as it continues to adapt and change. Preventative software maintenance helps to reduce the risk associated with operating software for a long time, helping it to become more stable, understandable, and maintainable.

Maintenance is a crucial aspect in software development as it ensure the smooth functioning of all the systems that is needed to carry out the operation of software development. The main purpose of maintenance is to maintain software functionality, make upgrades to the coding, and ensure any repairs needed to the software are completed.
Without maintenance, any software will be obsolete and essentially useless over time.

Ethical Considerations in Software Engineering:

Respect for Privacy. Respecting user privacy is a fundamental ethical obligation for software engineers.
This involves protecting the confidentiality of others and even clients.
Algorithm bias and fairness
Intellectual property and data integrity: Intellectual property rights are another area where ethical issues arise from.
Conflict of interest.
Software engineers can adhere to ethical rules by ensuring that;
*They contribute to society and human well being.
*They avoid harm to others.
*They are honest and trustworthy