1. Software engineering is the systematic process of designing, implementing, testing, deploying and maintaining software
2. Software engineering follows a structured process while traditional programming does not. Software engineering covers the entire lifecycle of a software while traditional programming focuses on solving the problems at hand and fulfilling immediate tasks.
3. Software Development LifeCycle is made of the following phases:
4. Planning – This involves identifying the problem the software is supposed to solve, establishing goals, calculating costs and estimating timelines.
5. Requirement analysis – This involves market research, gathering and analyzing functional and non-functional requirements of the software.
6. Design – This involves coming up with high level and detailed designs of the software architecture, data models and user interfaces
7. Implementation – This involves actualizing the designs using a programming language. Creating a functional system.
8. Deployment – this is releasing it to the public so that they can interact with it and give their feedback.
9. Maintenance – this is the ongoing support to fix bugs, add new features and ensure continued operation.
10. Agile Model is iterative and incremental while waterfall is sequential. Agile model is flexible and adaptive while waterfall is rigid, this is because it is used to build critical systems. Agile has little to no documentation as the requirements come after each sprint while waterfall is documentation driven.
11. Requirements Engineering is the process of defining, documenting and maintaining the requirements of a software system. The process involves the following:
12. Elicitation – This involves discussing with the stakeholders to understand their needs.
13. Analysis – This involves analyzing the requirements so as to handle conflicts and ambiguity.
14. Specification – Involves documenting the requirements in a clear and detailed manner.
15. Validation – This involves ensuring that the requirements actually reflect the need of the stakeholders.
16. Management – This involves updating the requirements during the software lifecycle.

Importance: It ensures that the system built reflects the needs and wants of the stakeholders.

Modularity – This is division of software into distinct, self-contained modules each responsible for a specific functionality.

Benefits

Easy to maintain, a module can be easily be debugged without affecting other modules.

Scalable – The system has a huge potential to grow because modularity is stable.

Reusable – Modules can easily be reused across different projects.

Unit testing – It is the testing of individual standalone components to ensure they are functional and efficient.

Integration testing – It is the testing of interactions between integrated components to see how they affect each other.

System testing – It is the testing of the complete system to ensure that individual components are working together in tandem.

Acceptance testing – This is the testing in the real-world environment by the end-users to ensure its within their expectations.

Importance: It ensures software released to the end-user is within their expectations, is functional and reliable.

1. Version Control

VCS are tools to manage changes to source code over time. They allow multiple developers to collaborate on a project, track changes and revert to previous versions if needed.

Importance

Allows collaboration as different developers can work together without physically being at the same place.

Examples; Git, Mercurial

1. The role of software project manager:

Planning -defining project scope, timelines and resources. Execution – overseeing the development process and ensuring milestones are met. Monitoring – tracking progress, managing risks and ensuring quality standards. Communication – coordinating between stakeholders, developers and clients. Budget management – ensuring the project stays within budget.

Challenges

Ensuring the right resources are available when needed.

Identifying and mitigating potential risks

Balancing stakeholder expectations and ensuring satisfaction.

1. Software maintenance is the process of modifying and updating software after its initial deployment. It includes:

Corrective maintenance – fixing bugs and defects

Adaptive maintenance – updating software to work with other systems

Preventive maintenance – making changes to prevent future issues

Maintenance ensures that software continues to function correctly and remains relevant as user needs and technology evolve.

1. Ethical considerations

Ensuring user data is protected and is kept confidential

Developing secure software to prevent unauthorized access

Respecting copyrights and avoiding plagiarism

1. Ensuring ethical standards

Code of ethics – adhering to professional codes of ethics

Transparency – being open about data collection and usage

Accountability – taking responsibility for the software they develop and its impact on users and society.