SE_Day1_Assignment

#Part 1: Introduction to Software Engineering

1.Explain what software engineering is and discuss its importance in the technology industry

software engineering is a branch of computer science used for developing, testing and maintaining software.

reliability- it ensures software performs as expected without bias especially for critical applications like healthcare ,finance.

efficiency - it helps to optimize developer workflow while maintaining high quality standards.

scalability and flexibility - it ensures that the system can handle an increased load without affecting performance.

security - implement protection practice like authentication, authorization and encryption to secure users information.

2.Identify and describe at least three key milestones in the evolution of software engineering.

Mastering complexity: (1983-1992), personal computer expanding, data and functional convergence, new programming approach.

Mastering process: (1968-1982), software engineering, ensure correctness models

Mastering machine: (1956-1967), hardware dependent high level languages, online code and fix.

List and briefly explain the phases of the Software Development Life Cycle.

planning - identify the software requirement or purpose and scope.

requirement analysis - identify the final user specification.

design - building the framework.

coding - converting software design into tangible code.

testing - examine the software for any bugs and glitches

Compare and contrast the Waterfall and Agile methodologies. Provide examples of scenarios where each would be appropriate.

waterfall methodology - Linear and sequential, each phase is completed before moving on.

- there is Low flexibility,

changes are hard to incorporate once a phase is complete.

- Customer feedback comes late, after the product is developed.

- Testing is done at the end of the development process.

agile methodology - Iterative and incremental, with multiple cycles (sprints).

- High flexibility, adapts to changing requirements.
- Regular customer feedback is incorporated into every sprint.
- Testing is continuous and done after each iteration.

Describe the roles and responsibilities of a Software Developer, a Quality Assurance Engineer, and a Project Manager in a software engineering team.

Software Developer - developing applications,programs and systems using programming languages and frameworks.
- maintaining and updating software to keep it functional.
- collaborating with other team members to ensure best practice when developing software.
- reporting to the project manager about the progress of the software development.

Quality Assurance Engineer - collaborate with stakeholders to understand and clarify software requirements.
- create development standards and procedures for the programmers to follow
- confirm that the software meets the requirement before deployment.
- analyse the product to identify bugs and suggest changes to make them more efficient.
- develop and execute automation scripts using open source tools.

Project Manager - assembles and leads the software development team.
- discuss the project and it's requirements with the client and software developers.
- create a blueprint for the project.

- tracking and communicating information regarding the project milestone.
- deliver the complete software to the client and regularly check its performance.

Discuss the importance of Integrated Development Environments (IDEs) and Version Control Systems (VCS) in the software development process. Give examples of each.

An **integrated development environment** (IDE) is a software platform that facilitates the creation of other software applications by providing a space to write, compile, and debug code, sometimes with value-adding tools that reduce development efforts. eg Visual Studio Code (VSCode)

importance:

- Programming languages have rules for how statements must be structured. Because an IDE knows these rules, it contains many intelligent features for automatically writing or editing the source code.
- An IDE can format the written text by automatically making some words bold or italic, or by using different font colors. These visual cues make the source code more readable and give instant feedback about accidental syntax errors.
- an IDE can make suggestions to complete a code statement when the developer begins typing.
- IDEs increase programmer productivity by performing repeatable development tasks that are typically part of every code change. The following are some examples of regular coding tasks that an IDE carries out.

- An IDE compiles or converts the code into a simplified language that the operating system can understand. - Some programming languages implement just-in-time compiling, in which the IDE converts human-readable code into machine code from within the application.
- The IDE allows developers to automate unit tests locally before the software is integrated with other developers' code and more complex integration tests are run.
- Debugging IDE enables a step through the code, line by line, as it runs and inspect code behavior. IDEs also integrate several debugging tools that highlight bugs caused by human error in real time, even as the developer is typing.

Version Control Systems (VCS) - are software tools that help software teams manage changes to source code over time. eg Git

importance:

Collaboration: Enables multiple developers to work on the same codebase without conflicts.

Change Tracking: Records detailed history of changes, allowing easy analysis of each modification.

-**Branching and Merging**: Supports creating branches for new features and merging them back into the main code.

Error Recovery: Allows reverting to previous versions if new changes introduce errors

What are some common challenges faced by software engineers? Provide strategies to overcome these challenges.

-**rapid technological advancement** places considerable pressure on software engineers to stay current.

Solution: adopting continuous learning practices and using agile methodologies to adapt to emerging trends, keeping their skills sharp in an ever-evolving industry. -

Time Constraints - Software engineering is a demanding and time-intensive field, often requiring engineers to work under high pressure to meet tight deadlines.

Solution: adopt agile methodologies, such as Scrum, to streamline workflows by dividing large projects into manageable sprints

-**Limited Infrastructure** - limited high-performance software engineering tools and computing platforms and inefficient data storage architectures.

Solution: Software engineers must rely heavily on a robust infrastructure to perform their jobs effectively.

Changing Software Requirements - Software requirements are often dynamic and subject to frequent changes, making it challenging for engineers to design and develop solutions that meet users' needs while accounting for future updates and bug fixes.

Solution: engineers can adopt approaches like agile development, which emphasizes iterative progress and adaptability, and modular design, which

enables flexibility by breaking systems into manageable, independent components.

Software Security - Programming secure software is a complex and challenging task.

Solution: research ways to defend against hacking, malware, phishing, insider and third-party threats

Software Accessibility and Usability - Overly complex software can frustrate or confuse users.

Solution: Use scalable architecture, Emphasize reliability.

Explain the different types of testing (unit, integration, system, and acceptance) and their importance in software quality assurance.

Unit tests - are close to the source of an application, They consist in testing individual methods and functions of the classes, components, or modules used by your software. - it ensures that each unit performs its intended function correctly, isolated from other components.

Integration tests - verify that different modules or services used by your application work well together.

- help to ensure data flows smoothly between modules and interfaces work as expected.

System testing -Focus on the entire software system as a whole, including all functionalities and interactions.

-It helps to verify that the system meets all functional and non-functional requirements, including performance, usability, and security .

Acceptance tests - are formal tests that verify if a system satisfies business requirements. They require the entire application to be running while testing and focus on replicating user behaviors.

- Whether the software meets the needs of the end-user and is ready for deployment.

#Part 2: Introduction to AI and Prompt Engineering

1. Define prompt engineering and discuss its importance in interacting with AI models.

   prompt engineering is the process where you guide generative AI solutions to generate desired outputs.

   Importance:

   Improved user experience - Prompt engineering makes it easy for users to obtain relevant results in the first prompt. It helps mitigate bias that may be present from existing human bias in the large language models' training data.

   Increased flexibility - A prompt engineer can create prompts with domain-neutral instructions highlighting logical links and broad patterns.

   developer control - Prompt engineering gives developers more control over users' interactions with the AI. Effective prompts provide intent and establish context to the large language models. Provide an example of a vague prompt and then improve it by making it clear, specific, and concise.

2. Provide an example of a vague prompt and then improve it by making it clear, specific, and concise. Explain why the improved prompt is more effective.

Draw a picture of a person

Draw a full-body portrait of a young woman with long brown hair, wearing a red jacket and blue jeans, standing in a park on a sunny day with trees and grass in the background.

Clarity: The improved prompt specifies what is being asked (a full-body portrait) rather than just a "person.

Specific Details: Describing the woman's appearance (long brown hair, red jacket, blue jeans) and the setting (park, sunny day, trees, grass) gives clear guidance on the image to be created.

Concise: The additional details provide a clear picture without being overly complicated, making it easier for the artist to understand exactly what is needed.