**BRAINERD MURITHI MUTHAMIA**

**PLP ASSIGNMENT**

1. **Define software engineering?**

*It is a branch of engineering that involves quantifiable approaches to the development, operation and maintenance of software.*

1. **What is software engineering, and how does it differ from traditional programming? Software Development Life Cycle (SDLC):**

*Software engineering differs from traditional programming in that it takes a broader, more systematic approach to developing, maintaining, and managing software systems. Traditional programming is largely concerned with writing code to solve specific issues or accomplish certain tasks. The Software Development Life Cycle (SDLC) is a structured process used for developing software, ensuring high quality and efficiency..*

1. **Explain the various phases of the Software Development Life Cycle. Provide a brief description of each phase. Agile vs. Waterfall Models:**

***Planning:*** *Entails determining the project's scope, objectives, and viability and developing a project plan.****Requirements Analysis:*** *Involves gathering and assessing user and stakeholder demands in order to establish precise software requirements.****Design****: Involves architecting the entire system and describing the software architecture, components, interfaces, and data models.****Implementation (Coding):*** *Creating actual code based on design specifications.****Testing:*** *Verifying and validating software to verify it fulfills requirements and is defect-free.****Deployment****: Entails releasing completed software to consumers, which includes installation, configuration, and initial support.****Maintenance:*** *Entails providing continuing assistance, correcting bugs, and implementing upgrades or enhancements as needed.*

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? Requirements Engineering:**

*The Waterfall model is a linear and sequential approach that requires each phase to be completed before proceeding to the next, providing simplicity, clear milestones, and easy progress tracking, whereas the Agile model is iterative and incremental, emphasizing flexibility, customer collaboration, and frequent delivery of functional software increments.*

*Agile and Waterfall methods differ primarily in their documentation, risk management, adaptability, structure, and customer participation.*

**Structure:***Waterfall is linear and sequential, with discrete, non-overlapping phases.  
Agile is an iterative and incremental process with overlapping phases.*

***Flexibility:****Waterfall: Inflexible; adjustments are difficult and expensive after a phase is done.  
Agile: Extremely adaptable; can easily handle changes throughout the development process.*

***Customer involvement:****Waterfall is limited to the initial requirements phase and final delivery.  
Agile: Continuous involvement and regular input throughout the project.*

***Documentation:*** *Waterfall: Extensive initial documentation.  
Agile requires minimal documentation and focuses on working software.*

***Risk Management:*** *Waterfall approach: Risks are identified and planned for early in the project.  
Agile involves continuous risk management with each iteration.*

***Delivery:****Waterfall: When the product is completed, it is distributed in its entirety.  
Agile development involves delivering incremental builds throughout the project.*

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

*The methodical and disciplined process of identifying, recording, and overseeing a software system's needs is known as requirements engineering, or RE. It is an essential step in the software development process that guarantees the finished product satisfies stakeholders' requirements and expectations.*

*The Process*

***Elicitation:*** *Obtaining requirements from stakeholders through a variety of methods, including interviews, surveys, workshops, observations, and document analysis.****Analysis and negotiation:*** *involve understanding and prioritizing requirements, resolving conflicts between the needs of many stakeholders, and negotiating trade-offs to establish an agreement.****Specification:*** *entails documenting the requirements in a clear, succinct, and unambiguous manner. This may entail generating formal documentation, user stories, use cases, and models.****Validation:*** *entails ensuring that the written requirements accurately reflect the stakeholders' needs and are feasible to implement within the given restrictions. This could include reviews, prototyping, and simulations.****Management:*** *entails maintaining and controlling requirements throughout the project's lifecycle. This includes keeping track of modifications, guaranteeing traceability, and coordinating stakeholder communication about needs.*

*Importance*

***Enhances Quality:*** *Having precise and well-defined criteria makes it easier to create systems that satisfy user needs and lower the likelihood of errors.****Cost and Time Efficiency:*** *By identifying requirements early on, fewer expensive and time-consuming changes are required later on in the development process.****Stakeholder Satisfaction:*** *Achieving corporate objectives and meeting stakeholder needs is made possible by well-managed requirements.****Risk Mitigation:*** *Proactive actions can be done by identifying potential risks and problems early in the process through understanding and addressing requirements*.

1. **Explain the concept of modularity in software design. How does it improve maintainability and scalability of software systems?**

*Modularity is a fundamental idea in software architecture that includes breaking down a software system into discrete, independent components or modules, each of which encapsulates a single functionality or a collection of related functionalities.*

***Isolation of Changes:*** *Low coupling means changes in one module are less likely to affect others.****Simplified Debugging and Testing:*** *Independent module testing simplifies bug identification and resolution.****Smaller, well-defined modules*** *improve readability and understandability compared to monolithic codebases.****Clearer documentation:*** *Modular systems typically have organized and focused documentation.****Reusable Components:*** *Well-designed modules can be reused in multiple projects or parts of the same project.*

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

***Testing Units*** *Verifying an application's smallest testable components, or units, which are usually single functions or methods, is the main goal of unit testing. This stage of testing, which is often carried out by developers in the development stage, tries to guarantee that every unit functions as planned when used independently.*

***Integration Testing*** *Integration testing checks that integrated units or modules interact correctly and function as a unit. Developers or a dedicated integration testing team can handle this level of testing.* ***System Testing****System testing entails verifying that the entire, fully integrated software system operates in accordance with the given specifications.*

***Assessment of Acceptance****The last stage of testing before software is shipped to customers or put on the market is called acceptance testing.*

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 (VCS) are tools for managing changes to source code over time. They maintain track of all code changes in a unique form of database.*

***Importance in Software Development***

***Collaboration:*** *VCS enables multiple developers to work on the same project simultaneously without overwriting each other's changes.*

***History and Audit:*** *Every change made to the code is recorded with a timestamp, the author's name, and a message describing the change.*

***Backup:*** *By using a remote repository, VCS ensures that there is always a backup of the codebase.*

***Code Review:*** *Many VCS tools integrate with platforms that facilitate code review processes.*

***Branching and Merging:*** *VCS allows developers to create branches for new features, bug fixes, or experiments without affecting the main codebase.*

***Version Control System Examples***

*The most popular DVCS is called* ***Git****. Recognized for its effectiveness, adaptability, and robust backing of branching and merging.****Subversion (SVN):*** *a well-liked CVCS that has gained popularity due to its uncomplicated version control methodology.****Mercurial*** *is another DVCS that is well-known for its performance and user-friendliness, much like Git.*

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

*A software project manager is responsible for* ***organizing*** *the many parts required to get a project from inception to completion.   
They are in charge of* ***setting the project scope and objectives****, which includes meeting with stakeholders to understand their needs and ensuring that the project's goals are aligned with the company's strategic direction.  
A software project manager's main responsibilities include* ***managing resources****.   
A software project manager is responsible for* ***time management*** *which is critical for meeting project deadlines.****Risk management*** *is another critical component of the software project manager's responsibilities. They identify possible risks early in the project and devise measures to mitigate them.****Effective communication*** *is essential in the function of a software project manager.*

1. **Define software maintenance and explain the different types of maintenance activities. Why is maintenance an essential part of the software lifecycle?**

*The term "software maintenance" refers to all of the tasks associated with improving and managing software following its deployment.*

***Corrective Maintenance:*** *Involves the correction of defects or bugs discovered in the software after it has been deployed.*

***Adaptive Maintenance:*** *Involves modifying the software to adapt it to changes in the environment, such as hardware or software platform upgrades.*

***Perfective Maintenance:*** *Enhancing the software to improve its performance, maintainability, or other attributes based on changing user requirements.*

*Preventive Maintenance:*

***Proactively*** *maintaining the software to prevent future issues or to improve its maintainability.*

***Emergency Maintenance:*** *Addressing critical issues or bugs that require immediate attention to avoid significant disruptions to users or operations.*

***Maintenance is an essential part of the software lifecycle for the following reasons:***

***Fixing bugs and resolving issues:*** *Defects or bugs can pop up during or after deployment. To guarantee that the program functions as intended, upkeep enables developers to locate, evaluate, and resolve these problems.*

***Adaptation to Changing Requirements:*** *As technology advances, so do user needs, corporate requirements, and technological surroundings.*

***Improving Software Quality:*** *The goals of maintenance procedures like preventative and perfective maintenance are to raise the caliber, functionality, and maintainability of the program.*

***Customer satisfaction:*** *Consistent maintenance guarantees that problems reported by customers are swiftly resolved.*

1. **What are some ethical issues that software engineers might face? How can software engineers ensure they adhere to ethical standards in their work?**

***Privacy Concerns:*** *When working with sensitive or personal data, software engineers may run into moral conundrums about data privacy.*

***Security Vulnerabilities:*** *When it comes to fixing security flaws and averting data breaches, engineers may run into moral quandaries.*

***Accountability and Transparency:*** *Concerning accountability and transparency, engineers may run across issues.*

***Accessibility:*** *In order to ensure that software systems are useable by people with impairments, engineers may need to consider ethical concerns connected to accessibility.*

*Engineers may encounter moral dilemmas pertaining to conflicts of interest.*

***Software engineers can take several steps to ensure they adhere to ethical standards in their work:***

***Keep Up:*** *Remain informed about the latest ethical precepts, behavior norms, and software engineering best practices.*

*The development of ethical* ***decision-making*** *abilities is crucial.*

***Never Stop Learning:*** Stay up to date on the latest developments in software engineering, emerging technologies, and moral dilemmas.

***Encourage Inclusion and Diversity:*** *Make the case for inclusion and diversity in software development teams and the products that are being developed.*

***Put User Privacy and Data Security First:*** *Put robust security measures in place to safeguard user privacy and data.*

*.*