**SE ASSIGNMENT 2: Faith Njuguna**

Email: [faithnjuguna2003@gmail.com](mailto:faithnjuguna2003@gmail.com)

Greetings,

I have had to do this assignment on a word document because my IDE(Visual Studio Code) could not install the extension for GitHub Classroom since it brings this error: “This extension is deprecated as it is no longer being maintained.” and the due time for submission was approaching.

Kindly provide me with the steps that I may follow to resolve this issue.

Regards,

Faith Njuguna.

**SE ASSIGNMENT 2: Faith Njuguna**

Define Software Engineering:

Software Engineering is a discipline that involves the application of engineering principles to the design, development, maintenance, testing, and evaluation of software. It differs from traditional programming in its emphasis on a systematic, disciplined, and quantifiable approach. While traditional programming focuses primarily on coding and implementation, software engineering encompasses the entire lifecycle of software development, including planning, analysis, design, testing, and maintenance.

The Software Development Life Cycle (SDLC):

The Software Development Life Cycle (SDLC) is a structured process that defines the stages involved in the development of software. The phases include:

1. Planning: This phase involves defining the project goals, scope, resources, timeline, and budget. It sets the groundwork for the entire project.

2. Requirements Analysis: In this phase, the requirements of the software are gathered, analyzed, and documented. This involves interacting with stakeholders to understand their needs and expectations.

3. Design: This phase focuses on creating the architecture and design of the software. It includes high-level design (system architecture) and low-level design (detailed design of components).

4. Implementation (Coding): During this phase, the actual code is written based on the design documents. It is the process of converting design into a working software component.

5. Testing: The software is rigorously tested to identify and fix defects. This phase ensures the software meets the required standards and functions correctly.

6. Deployment: The software is deployed to the production environment where it will be used by the end-users. It involves installation, configuration, and initial support.

7. Maintenance: This phase involves updating and improving the software to adapt to changes, fix bugs, and enhance functionality over time.

* Agile vs. Waterfall Models:

Agile and Waterfall are two distinct methodologies in software development.

- Waterfall Model: A linear and sequential approach where each phase must be completed before the next one begins. It is suitable for projects with well-defined requirements and little expected change.

- Advantages: It is simple, easy to manage and has clear milestones.

- Disadvantages: Inflexibility, difficulty in accommodating changes once a phase is completed.

- Scenarios: It is best for projects with clear, unchanging requirements and well-understood technology.

- Agile Model: An iterative and incremental approach that allows for flexibility and continuous improvement. Development is broken into small cycles (sprints) with frequent reassessment and adaptation of plans.

-Advantages: It is flexible, customer involvement, quick delivery of functional software.

-Disadvantages: Requires constant feedback, can be challenging to manage without experience.

- Scenarios: Ideal for projects with evolving requirements or when rapid delivery of features is needed.

Requirements Engineering:

Requirements Engineering is the process of defining, documenting, and maintaining the requirements for a software system. It involves:

1. Elicitation: Gathering requirements from stakeholders through interviews, surveys, and observation.

2. Analysis: Understanding and modeling the gathered requirements to ensure they are clear, complete, and feasible.

3. Specification: Documenting the requirements in a detailed and structured manner.

4. Validation: Ensuring the documented requirements accurately reflect the stakeholders' needs.

5. Management: Handling changes to the requirements as the project progresses.

Importance: It ensures that the software meets the needs of the users, reduces the risk of project failure, and provides a clear foundation for design and development.

Software Design Principles:

Modularity is a design principle that divides a software system into smaller, manageable, and independent modules. Each module performs a specific function and can be developed, tested, and maintained independently.

- Improves Maintainability: Easier to update and fix parts of the system without affecting the whole.

- Enhances Scalability: New features or modules can be added with minimal impact on existing components.

Testing in Software Engineering:

Different levels of software testing ensure the software functions correctly at various stages of development:

1.Unit Testing: Testing individual components or modules for correctness. Typically performed by developers.

2. Integration Testing: Testing the interactions between integrated modules to identify interface defects.

3. System Testing: Testing the complete and integrated software system to ensure it meets the specified requirements.

4. Acceptance Testing: Conducted by end-users to ensure the software meets their needs and is ready for deployment.

Importance: Testing is crucial to identify and fix defects early, ensure quality, and validate that the software meets the specified requirements.

Version Control Systems:

Version Control Systems (VCS) are tools that help manage changes to source code over time. They track revisions, allow multiple developers to work on the same project, and facilitate collaboration.

- Importance: They prevent code conflicts, maintain a history of changes, and allow for rollback to previous versions.

- Examples include:

- Git: Distributed VCS, popular for its branching and merging capabilities.

- SVN (Subversion): Centralized VCS, known for its simplicity and reliability.

Software Project Management:

A Software Project Manager oversees the planning, execution, and closing of software projects. Key responsibilities include:

- Planning: Defining project scope, goals, resources, and schedule.

- Team Management: Coordinating and leading the project team.

- Risk Management: Identifying, analyzing, and mitigating risks.

- Monitoring and Control: Tracking project progress and making necessary adjustments.

Challenges: Balancing scope, time, and resources, managing stakeholder expectations, and ensuring team collaboration.

Software Maintenance:

Software Maintenance involves modifying software after its initial deployment to correct faults, improve performance, or adapt to a changed environment. Types of maintenance include:

1. Corrective Maintenance: Fixing bugs and defects.

2. Adaptive Maintenance: Updating software to work in a new or changed environment.

3. Perfective Maintenance: Enhancing software functionality and performance.

4. Preventive Maintenance: Making changes to prevent future problems.

Importance: Ensures the software remains useful, efficient, and secure over time.

Ethical Considerations in Software Engineering:

Software engineers may face ethical issues such as:

- Privacy: Protecting user data from unauthorized access and misuse.

- Security: Ensuring the software is free from vulnerabilities that could be exploited.

- Intellectual Property: Respecting copyright and avoiding plagiarism.

- Transparency: Being honest about the capabilities and limitations of the software.

Ensuring Ethical Standards: Adhering to professional codes of conduct, such as those from the ACM or IEEE, and prioritizing the public good in decision-making.

**References**

1. Pressman, R. S. (2014). Software Engineering: A Practitioner's Approach. McGraw-Hill.

2. Sommerville, I. (2015). Software Engineering. Pearson.

3. Beck, K., & Andres, C. (2004). Extreme Programming Explained: Embrace Change. Addison-Wesley.