A REVIEW ON INTRODUCTION TO SOFTWARE DEVELOPMENT (WEEK ONE) OF WEEKLY REPORT

Day 1

An Overview of SDLC

The Software Development Life Cycle (SDLC) is a framework for planning, developing, testing, and deploying software.

It typically involves phases such as:

(1) Requirement gathering:This phase involves collecting detailed information about what the system should do, user requirements, and constraints. The goal is to ensure a clear understanding of the project objectives.
(2) Analysis:In this phase, the requirements gathered are thoroughly analyzed to determine their feasibility and how they fit into the overall system architecture.
(3) Design:The design phase involves creating the architecture and detailed design of the software based on the analyzed requirements. This includes defining system components, data structures, interfaces, and interactions. The design is often represented through diagrams and models, providing a blueprint for the developers to follow during implementation.
(4) Implementation:During this phase, the actual coding and development of the software occur.
(5) Testing:After implementation, the software undergoes rigorous testing to identify and fix defects or issues.
(6) Maintenance:This phase involves fixing any issues that arise, making improvements, and adapting the software to changes in the environment or user requirements.
(7) Documentation:Throughout the SDLC, documentation is crucial for capturing and communicating information about the software. Documentation includes requirements specifications, design documents, user manuals, and maintenance guides.

The SDLC aims to produce high-quality software efficiently and effectively while meeting user needs and project constraints.

Day 2

Software Development Methodologies

Software development methodologies are structured approaches used to plan, design, develop, test, and maintain software. They provide frameworks and processes to guide teams in managing projects and achieving successful outcomes.

Very important software development methodologies include;

Agile**:** Focuses on iterative development, flexibility, and close collaboration with stakeholders. Projects are divided into sprints or iterations, allowing for frequent reassessment and adaptation based on feedback.

Waterfall**:** A linear and sequential approach where each phase (requirements, design, implementation, testing, deployment, maintenance) must be completed before moving to the next. It is structured and easy to manage but can be inflexible to changes.

DevOps**:** Combines development and operations to improve collaboration and efficiency throughout the software lifecycle. It emphasizes automation, continuous integration, and continuous delivery to streamline workflows and enhance deployment frequency.

Day 3

Introduction to C++ and Java

This deals with the major programming languages that are most efficient for software development, they are java and c++. This covers the advantages and disadvantages of the programming languages over the others, it also looks and the various programming paradigms, libraries, functionalities, etc.

C++**:** A versatile, high-performance programming language supporting multiple paradigms (object-oriented, procedural, and generic). It is often used for system/software development, game programming, and applications requiring high efficiency.

Java**:** An object-oriented language designed for portability across platforms via the Java Virtual Machine (JVM). It is widely used for building web applications, enterprise solutions, and Android apps.

Day 4

Java/C++ Tools and IDE Setups

Visual Studio (for C++): An integrated development environment (IDE) from Microsoft that provides robust tools for code editing, debugging, and project management. It supports various C++ standards and integrates well with build systems.

Eclipse (for Java): An open-source IDE known for its extensibility and support for Java development. It offers features like code completion, debugging, and plugin integration.

IntelliJ IDEA (for Java): A powerful IDE renowned for its intelligent code assistance, user-friendly interface, and strong support for modern Java technologies and frameworks.

Day 5

Version Control Systems (Git Basics)

Git is a distributed version control system used to track changes in source code and collaborate on software projects. Key concepts include:

Repositories**:** Storage locations for project files and version history.

Commits**:** Snapshots of changes made to files.

Branches**:** Parallel versions of the project allowing multiple features or fixes to be developed simultaneously.

Merging**:** Combining changes from different branches. Git enables efficient collaboration, maintains project history, and supports versioning and code management.

In Summary, the entire introduction has gone on a broad scale to describe and give us understanding of the different stages/phases that must be carried out by developers in order to write quality software programs.