**Assignment 1**

**Create a one-page infographic that outlines the SDLC phases highlighting the importance of each phase and how they interconnect.**

Software Development Life Cycle (SDLC) is the process by which software comes to life. It can vary depending on the framework chosen by the team (more on that later), but whatever path you take, the journey from idea to final software in the user’s hands is what we call SDLC.

**So, what are the phases of the software development life cycle?**

**A full SDLC has 7 basic stages:** Planning, requirements, design and prototype, software development, testing, deployment, and maintenance.

In some cases, depending on different variables (project, team, manager, etc.), certain steps can be omitted, split, or combined.

A well-planned SDLC helps teams reduce costs and release software faster by having a set plan to adhere to (even if some frameworks are more chaotic than others, it is still controlled chaos) and providing a clear “bird’s eye view” to help identify inefficiencies and roadblocks.

**Working of SDLC**

There are 7 software development life cycle phases, and they are to be approached sequentially, although in some cases, two might run concurrently (just as development and testing).

SDLC “works” when your team organizes and executes according to this sequence; how they tackle each individual step will depend upon the framework they chose (more on that later.) For now, let’s outline what happens in each stage/phase.

1. **Planning**

* In this phase, the project leads to defining the project’s purpose and the desired result.
* If the team is developing for a customer instead of to market, the project manager meets with them to discuss the product, its purpose, and the results they want to achieve. The team gathers as much information about the product from the customer.
* By the end of the planning phase, the team leads should have a working estimate of how much the project will cost and who will be part of the project. They also set a project deadline and milestones and overall create the basic structure for the project.
* By the end of this phase (or, at the very least, the next one), each team member must understand their roles and tasks.

1. **Requirements**

* This second phase of the software development life cycle is often done concurrently with the first. Here, the project lead analyzes the product or client’s goals and decides on the features to aim for as a final goal. Defining and establishing requirements determines what the application will do once launched, the necessary components, and the resources needed to launch it.
* For example, if a team wants to develop software to control a robot that cleans, then the physical robot would be a requirement (component) in the process.

1. **Design and Prototype**

* Once phases 1 and 2 are understood and established, developers can start designing the software.
* The design phase defines how a software application will work. During this phase, teams decide on the programming language, screen layouts, and relevant documentation they will use.

**Some of the fundamental aspects’ developers cover during this phase are:**

* **Architecture:** Teams define if they want a specific type of template or if they want to implement any type of industry practice.
* **User Interface:**Teams define the way users will be interacting with the platform.
* **Security:** Developers must define how they will keep the application secure. This means they need to decide how to protect user data and general app data.
* **Programming:** Define the project’s tech and tool stack.
* Prototyping is also part of this phase. A prototype is a basic idea of how the application looks and works.
* Prototypes allow customers to get a sneak peek of how their application will look; they might even discover that their original idea is not good enough and change it during this phase.

1. **Software Development**

During this phase, developers start programming.

* If they work on a small project, one developer takes over the coding tasks, while on large projects, the codebase might be worked on by several developers.
* Before starting to code, teams must have clear predefined guidelines to ensure the code’s quality. In this phase, developers start building the entire system and shaping the project.
* Depending on each team’s model, the phase may be conducted in sprints (Agile) or a single block (Waterfall). Teams spend most of their time during this phase ensuring that the application will work efficiently.

1. **Testing**

* Often, testing happens in parallel with development, as developers write and test the code they’ve produced before moving on to the next coding task.
* During this phase, different types of testing occur, such as code quality, unit testing, integration testing, performance testing, and security testing.
* Running testing in parallel with development means that bugs can be fixed within the same sprint or time block, which is more efficient than adding a whole block of coding to be done at the end of the project. It also mitigates the trouble of bug fixes generating new bugs themselves.

1. **Deployment**

* The deployment process starts once the testing phase is over and there are no bugs or errors in the development backlog.
* The team ensures that the software is up-to-date and secure enough for users and pushes it from the development environment to a live environment–usually an app store.
* During this phase, the tech support team looks for user feedback and ensures it reaches the dev team.

1. **Maintenance**

* At this point in the SDLC cycle, the application is successfully launched and being used.
* Yet this last phase is still important because bugs or errors missed during testing are bound to appear. Simultaneously, by studying user behavior and feedback, the team can start to think about and plan for upgrades.

An SDLC serves as a roadmap for developers and project managers, guiding them to deliver high-quality, robust software that meets user expectations and is delivered on time and within budget. It is the cornerstone of a robust software development process.