UGANDA CHRISTIAN UNIVERSITY

A Centre of Excellence in the Heart of Africa

NAME : ANEI AGANY THEM

REGISTRATION NUMBER: IS19B00/023

FACULTY/SCHOOL : SCIENCE AND TECHNOLOGY

COURSE : BACHELORS OF SCIENCE IN COMPUTER SCIENCE (BSCS3)

LECTURER : **MR. SIMON F. LUBAMBO**

COURSE UNIT : SOFTWARE PROJECT MANAGEMENT

ASSIGNMENT : **ONE**

Question:

Create a GitHub repository and upload a file defining and explaining the different software development cycles

Software Development Life Cycle is the application of standard business practices to building software applications. However here are followings differences software development life cycle along with their explanations Planning, Requirements, Design, Build, Document, Test, Deploy, Maintain.

Planning

In the Planning phase, project leaders evaluate the terms of the project. This includes calculating labor and material costs, creating a timetable with target goals, and creating the project's teams and leadership structure.

Planning can also include feedback from stakeholders. Stakeholders are anyone who stands to benefit from the application. Try to get feedback from potential customers, developers, subject matter experts, and sales reps.

Planning should clearly define the scope and purpose of the application. It plots the course and provisions the team to effectively create the software. It also sets boundaries to help keep the project from expanding or shifting from its original purpose.

Define Requirements

Defining requirements is considered part of planning to determine what the application is supposed to do and its requirements. For example, a social media

application would require the ability to connect with a friend. An inventory program might require a search feature.

Requirements also include defining the resources needed to build the project. For example, a team might develop software to control a custom manufacturing machine. The machine is a requirement in the process.

Design and Prototyping

The Design phase models the way a software application will work. Some aspects of the design include:

Architecture – Specifies programming language, industry practices, overall design, and use of any templates or boilerplate
User Interface – Defines the ways customers interact with the software, and how the software responds to input
Platforms – Defines the platforms on which the software will run, such as Apple, Android, Windows version, Linux, or even gaming consoles
Programming – Not just the programming language, but including methods of solving problems and performing tasks in the application
Communications – Defines the methods that the application can communicate with other assets, such as a central server or other instances of the application
Security – Defines the measures taken to secure the application, and may include SSL traffic encryption, password protection, and secure storage of user credentials

Prototyping can be a part of the Design phase. A prototype is like one of the early versions of software in the Iterative software development model. It demonstrates a basic idea of how the application looks and works. This “hands-on” design can be shown to stakeholders. Use feedback o improve the application. It’s less expensive to change the Prototype phase than to rewrite code to make a change in the Development phase.

Software Development

This is the actual writing of the program. A small project might be written by a single developer, while a large project might be broken up and worked by several teams. Use an Access Control or Source Code Management application in this phase. These systems help developers track changes to the code. They also help ensure compatibility between different team projects and to make sure target goals are being met.

The coding process includes many other tasks. Many developers need to brush up on skills or work as a team. Finding and fixing errors and glitches is critical. Tasks often hold up the development process, such as waiting for test results or compiling code so an application can run. SDLC can anticipate these delays so that developers can be tasked with other duties.

Software developers appreciate instructions and explanations. Documentation can be a formal process, including wiring a user guide for the application. It can also be informal, like comments in the source code that explain why a developer used a certain procedure. Even companies that strive to create software that's easy and intuitive benefit from the documentation.

Documentation can be a quick guided tour of the application's basic features that display on the first launch. It can be video tutorials for complex tasks. Written documentation like user guides, troubleshooting guides, and FAQ's help users solve problems or technical questions.

Testing

It's critical to test an application before making it available to users. Much of the testing can be automated, like security testing. Other testing can only be done in a specific environment – consider creating a simulated production environment for complex deployments. Testing should ensure that each function works correctly. Different parts of the application should also be tested to work seamlessly together—performance test,

to reduce any hangs or lags in processing. The testing phase helps reduce the number of bugs and glitches that users encounter. This leads to a higher user satisfaction and a better usage rate.

Deployment

In the deployment phase, the application is made available to users. Many companies prefer to automate the deployment phase. This can be as simple as a payment portal and download link on the company website. It could also be downloading an application on a smartphone.

Deployment can also be complex. Upgrading a company-wide database to a newly-developed application is one example. Because there are several other systems used by the database, integrating the upgrade can take more time and effort.

Operations and Maintenance

At this point, the development cycle is almost finished. The application is done and being used in the field. The Operation and Maintenance phase is still important, though. In this phase, users discover bugs that weren’t found during testing. These errors need to be resolved, which can spawn new development cycles.