GROUP 5 WORK

Tony Kamau Musyimi	**SC150/0521/2021**
Shanice Wanjiku	**SC150/0797/2022**
Macbriton Muraya	**SC150/0313/2022**
Timothy Muthoka	**SC150/4031/2022**
Winfred Gatwiri	**SC150/0803/2022**

1)SYSTEM ANALYSIS ACTIVITIES

Gather Detailed Information.

Planning documents and policy statements are mostly reviewed to obtain additional information. They also study the existing system of company's business area including its system and work documentation.

Define Requirements

After the information gathered, system analysts define requirements for the new or upgraded system based on this information. According to Sat-zinger, Jackson, and Burd (2012), system requirement include the system must perform (functional requirements) and such related issues as user-interface formats and requirements for reliability, performance, and security (nonfunctional requirements).

Prioritize Requirements

Users and system analysts have to discuss to determine which functions are truly important and which are fairly important but not really required. Prioritizing requirements need to be done due to the limitation of resources and the need to justify the scope of system.

Develop User-Interface Dialogs

To most users who are working with the new or upgraded system, the user-interface becomes an issue. It does because they feel quite unsure about their requirements and the desired form of user-interface. These all are the matter of user-interface. Hence, system analysts need to develop user-interface prototypes on the actual devices which users will use and go on a demo to the users in later stages.

Evaluate Requirements with Users

In practice, the addition of system requirements may occur. The issue is that the addition occurs after the analysts have gathered the information in earlier stage. If this does so, analysts usually use an iterative process to fulfill system requirements with the new requests from users. Requirements models and prototypes are complete and accurate by conducting the processes of eliciting requirements, building models and prototypes, and evaluating them with users in many times.

2) DESIGNING WITH SOFTWARE SPECIFICATION REQUIREMENTS

Designing with software specification requirements is an essential step in the software development process. It involves creating a detailed plan or blueprint for the software based on the specified requirements. This design phase helps in visualizing the overall structure, components, and functionality of the software before the actual development begins.

Here are a few key points to consider when designing with software specification requirements:

1. Understand the requirements: Thoroughly analyze and comprehend the software specification requirements provided by the stakeholders. This includes functional requirements (what the software should do) and non-functional requirements (performance, security, etc.).

2. Identify system components: Break down the software into smaller modules or components. Define the relationships and dependencies between these components. This helps in organizing the design and understanding the overall system architecture.

3. Define interfaces: Specify the interfaces between different components or modules. Clearly define the inputs, outputs, and communication protocols between these interfaces. This ensures proper integration and interoperable.

4. Design data structures: Identify the data elements and structures required for the software. Define the data models, databases, and data flow diagrams. This helps in managing and manipulating data efficiently.

5. Establish design patterns: Utilize design patterns and best practices to solve common software design problems. This promotes re-usability, maintainability, and scalability.

6. Consider constraints: Take into account any constraints or limitations imposed by the software specification requirements. This may include hardware limitations, budget constraints, or time restrictions.

7. Document the design: Create comprehensive design documentation that includes diagrams, flowcharts, and descriptions of the software's architecture, components, and interfaces. This documentation serves as a reference for developers during the implementation phase.