SECTION A

No 1

a)

Successful project management involves meeting three key constraints: scope (what needs to be done), time (the schedule to complete the work), and cost (the financial resources required). These elements form the Project Management Triangle, and any change to one constraint often impacts the others. Balancing these requires strategic decision-making and resource management.

1. Scope:

This refers to the precise objectives, deliverables, and tasks involved in the project. Take for instance planning a birthday party.

If you intend to invite more guests, hence increasing the scope, the time needed for planning must be extended, and one may be forced to increase one's budget due to accommodating more guests who will take more food and more decorations are needed. In case one increases the scope, consideration of the meeting of deadlines and keeping within budget is supposed to be made.

2. Time:

This constraint encompasses the schedule of the project, including deadlines for particular milestones.

Using the same example of a birthday party, if the time available for preparation is short, say two days, one may have to reduce the scope by either inviting fewer people or simplifying the menu so that everything will be ready on time. Meeting a tight schedule often requires prioritizing tasks and possibly sacrificing some elements of the project.

3. Cost:

This refers to the budget for the project. You may be on a very strict budget regarding the birthday party and thus would have to limit your guest or use less expensive decorations and food options.

You will need to extend the timeline, either to allow for fundraising or to find more cost-effective solutions, if the first estimates prove too low; hence, balancing the cost constraint against time and scope.

Most successful balancing in real-life project management includes trade-offs of a single attribute in favour of others. In a software development instance, to enable completion at a faster time, an under-pressure group could just limit the product's scope to meet such time requirements or invest resources (cost) for achieving both quality and timely execution. The thing that would matter most was being up-front with communication regarding this expectation to stakeholders through active communication for informed decision-making for project goals.

By using real-life examples, such as planning events or managing projects, we can better understand how to navigate these constraints effectively. When renovating a home, the scope might include updating the kitchen, bathroom, and living room. If you decide to add more rooms to the renovation, increasing the scope, it will likely extend the timeline and increase costs due to additional materials and labour. This could entail scaling back the renovation to kitchen and bathroom only, enabling you to stay within budget and on schedule. What this really points out is that a proper scope of work upfront is necessary, and not being afraid to make modifications as necessary.

The balance of scope, time, and cost is a strategic process where changes in one area are carefully weighed in consideration of the others.

b)

1. Structured Development: While programming allows you to write code, software engineering concepts provide you with a way to structure that code. For example, when developing a large application, such as a mobile app, using software engineering principles like modular design helps to break down the app into manageable components. This makes maintenance easier since updating one feature does not necessarily affect the workings of the entire system.

2. Best Practices: You can write some code that works, but in software engineering, a lot of emphasis is put into best practices, such as reviews and testing of the code. For example, Google and Microsoft have to perform very aggressive testing of their software to ensure it is reliable. It catches bugs early and gives better quality software.

3. Problem-Solving: Software engineering helps you in understanding any problem methodologically. For example, on an e-commerce website, if you need to provide functionality for shopping carts or payments, knowledge of design patterns may help in developing them faster. The performance is really good and also user friendly.

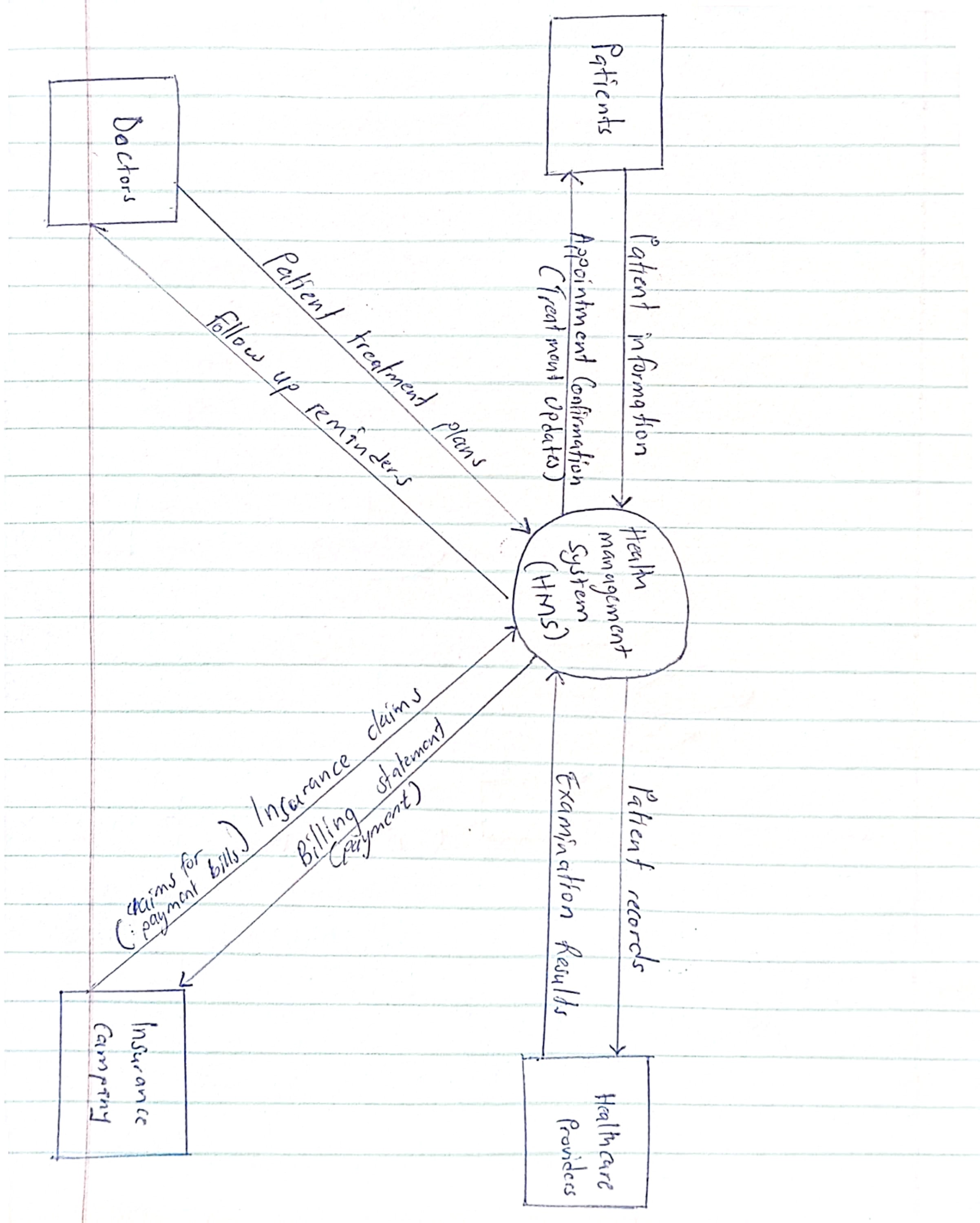
4. Project Management: Familiarity with software engineering concepts includes understanding methodologies like Agile. For example, a team building a new feature for a social media platform might use Agile sprints to plan and execute their work in short cycles, allowing for quick adjustments based on user feedback.

5. Career Advancement: Tech companies, most of them, require software engineers who would understand both the coding aspect and the software engineering principles in place. If you consider working with companies like Amazon, knowledge on how to apply concepts in software engineering helps you outshine other competitors in interviews. This company values engineers that can develop large-scale projects and work comfortably with teams.

In other words, programming is an important skill, but the concepts of software engineering increase your ability to develop better-quality software in less time, solve complex problems, and further your career as a tech professional.

c)

A data flow diagram is a representation that describes the flow of data within a system. It highlights the inputs, processes, outputs, and storage of data. In relation to an HMS, the DFD will help in describing how the patient's information and the healthcare processes interact in the interest of effective healthcare delivery.

In this scenario, the key concept of the DFD in this scenario is to highlight major processes such as patient registration, medical documentation, billing, and reporting. Each process receives some inputs, like patient data or insurance information, operates on that data, and then produces outputs, such as updated records or bills. The DFD also shows how the data flows between different entities, including healthcare providers, patients, and insurance companies.

The context diagram for the scenario above;

No 2

a)

Verification and validation are two important concepts in software development, each serving a different purpose. Here is a detailed differentiation, supported with examples related to MTN MoMo Pay, one of the popular mobile payment apps in Uganda.

1. Verification:

Is the process of checking whether the software meets the specified requirements and is built correctly. It involves reviewing documentation, design, and code to ensure everything aligns with the initial requirements.

Example: In the case of MTN MoMo Pay, verification may involve verifying information about the design specifications of an app to ensure that it has provided the essential features required, such as user registration, money transfer, transaction history, etc. Also, code reviews to assure the implementation of user authentication follows security standards put forth in the requirements documentation.

2. Validation:

Validation is the process that takes place to assess the developed software, whether it truly satisfies the needs and expectations of the end-users. It focuses on whether the right product has been built and performs the intended functions in the real world.

Example: In case of the MTN MoMo Pay app, the verification could take the form of user acceptance testing whereby real people are taken through the use of an app to navigate the app to send money, make payments, and execute transactions efficiently. In so doing, it shall solicit responses if the app is performing as, one would have desired regarding ease, efficiency, reliability, and cost-effectiveness.

In other words, verification is an assurance that the MTN MoMo Pay app has been constructed properly and according to specifications. On the other hand, validation ensures the satisfaction of the real needs of users in the Ugandan market. Each process is important for delivering a quality mobile payment solution.

b)

If I were to choose a project manager for the development of a social media application for VU students, I would require a design team with the following set of skills, along with reasons:

1. User-Centric Design Skills: This is because the app has to be intuitive and appealing for the students. A user-cantered approach ensures that the design will be according to the needs and preferences of the target audience.

2. Design Tools: Proficiency in the use of design tools like Adobe XD, Sketch, or Figma is important. In fact, these tools will help them create wireframes and prototypes seamlessly, which are necessary in the visualization of an application prior to its actual development.

3. Strong Communication Skills: Effective communication is vital for team collaboration with developers and stakeholders. It allows the design team to clearly articulate their ideas and incorporate feedback efficiently.

4. Knowledge of social media Trends: Knowledge of current trends in social media will help the team design features that appeal to the students. This makes the app relevant and appealing to its target audience.

5. Problem-solving Skills: A design team should be in a position to identify and solve design-related problems that arise in the course of development. The skill is crucial in balancing aesthetics with functionality.

6. Adaptability and Flexibility: The fast pace of the development of an app means one needs to be open to changes in requirements or feedback for iterative improvements to the design.

7. Research Skills: In-depth user research and usability testing will keep the design pegged to real user needs and behaviours, leading to an effective app.

8. Collaboration Skills: Since app development is a team effort, collaboration skills are necessary to work cohesively with developers and marketers to produce a unified product.

These skills are all important in their contribution to the overall success of the app in meeting the needs of VU students and fostering engagement within the community.

c)

There are several SDLC models that can be considered in upgrading the computer system at Victoria University, each with their strengths and applications. The following are some of the types of SDLC models, their explanations, and examples:

1. Waterfall Model:

This is a linear and sequential approach where each phase must be completed before moving to the next. It’s suitable for projects with well-defined requirements. It is simple and easy to manage and well suited for projects with clearer goals and minimal changes. Some of its problems is it’s not flexible therefore changes in later phases can be costly .

For instance, if the university has a clear understanding of the system requirements and functionalities needed, the waterfall model could be effective. Phases would include requirements gathering, system design, implementation, testing, deployment, and maintenance.

1. Agile Model:

Agile is an iterative and incremental approach; it offers a lot of leeway for flexibility and adaptation. This model is suitable in those projects where the requirement could evolve with time. It is flexible and adaptive to changes and also involves stakeholders throughout the development process. The challenge it faces is it requires active collaboration and can be challenging to manage for large teams.

Assuming the university intends to release new features based on end-user feedback during the actual implementation, Agile would offer room for regular updates and changeover. The project can then be divided into sprints where every sprint focuses on either developing specific features or making necessary improvements.

1. Spiral Model:

This combines iterative development with the systematic aspects of the waterfall model. It's useful for large, complex projects where risks need to be managed carefully. This one mainly focuses on risk mitigation and also allows for progressive refinement of the system. Its challenged by being expensive and complex to manage.

For example, if the upgrade involves integrating various systems and there are uncertainties, the spiral model allows for risk assessment and prototyping in each iteration. Phases include planning, risk analysis, engineering, and evaluation.

1. V-Model (Verification and Validation):

This model emphasizes verification and validation at each stage of the development. It's suitable for projects where quality assurance is critical. This mainly focuses heavily on quality by linking development and testing phases and also ensures errors are detected early. The challenge is that it can be time consuming and also rigid and less flexible to changes during development.

An example could be if the university prioritizes thorough testing of the new system at each stage; the V-Model would ensure that for every phase of development, there's a corresponding phase of testing to attain high-quality results.

1. Incremental Model:

The system is developed in small increments or modules. This permits partial implementation of the system. This model is useful when a university wants to implement certain features as quickly as possible, while still continuing to develop others. It can be easier to manage risk as issues are resolved incrementally. A challenge is it may extend the overall time line if increments are not well defined

For instance, the university can upgrade its student enrolment system first, and then later, in successive increments, add features like online course registration.

The selection of a proper SDLC model for the upgrade project of the university depends on the specific needs and constraints of the system, including clarity of requirements, risk management, and need for flexibility. Each has its strengths that can be molded into the project goals.

SECTION B

No 4

a)

In conducting a feasibility study for a project, one must consider several categories for the success of the project. Here's a discussion of operational, technical, and economic feasibility with appropriate examples as they include below;

1. Operational Feasibility:

This concerns whether the proposed system will be effective in the existing organizational environment. It involves assessing whether the organization's current operations can support the new system and if the users are willing to adapt to it.

For instance, if Victoria University is to implement a new student information system, then operational feasibility would mean assessing whether staff are trained to use the new system and if it aligns with the university's operational procedures. The operational feasibility would be considered high if the staff could use the system without significantly disrupting their workflow.

MTN MoMo Pay for checking if users and merchants in Uganda are comfortable using the system for payments and if it integrates seamlessly with existing operational workflows.

ii . Technical Feasibility:

This represents the availability of technology that will be required for the project and the technical ability of the organization to implement it. It encompasses consideration of hardware, software, and technical skills.

For example, if the university wants to enhance its computer system by introducing cloud-based services, then technical feasibility would check whether the infrastructure is capable of cloud technology and if the IT employees have the skill to take care of it and manage the system. If the technology and expertise required are available with the university, the technical feasibility would be viable.

iii. Economic Viability:

It examines whether the project is cost-effective, whether the benefits derivable would justify the cost of investment. It covers a cost-benefit analysis that would describe financial implications.

For instance, a university may want to invest in a new Learning Management System; the economic feasibility will look at total costs of implementation, including software, training, and maintenance, against the potential benefits to be derived from improved student engagement and administrative efficiency. If the benefits significantly outweigh the costs, the economic feasibility would be considered positive.

b)

Deriving the size of a software product is essential for several reasons, including project estimation, resource allocation, and performance measurement. Here are some methods to derive software size and why it's necessary, along with examples:

1. Function Points:

This method measures the functionality provided to the user from the requirements. Function points take into consideration inputs, outputs, user interactions, and data files.

For example, if a banking application has all sorts of features like account management, transaction processing, and reporting, function points are able to quantify the size based on those functionalities. This in turn helps in estimating the required effort for development.

MTN MoMo Pay may have functions such as "Add Money" (simple), "Transaction History" (medium), and "Secure Payments" (complex), with a total FP of 300

1. LOC consists of the amount of a number of lines of code that are written to develop this software.

For example, in a web application project, a size of 10,000 lines of code can be used to estimate the complexity or time consumed in testing or maintenance of the software. LOC, however, is known to depend greatly on the programming language and coding standards.

A mobile app like MTN MoMo Pay might have 50,000 LOC to manage user authentication, transactions, and notifications.

1. Use Case Points:

This technique estimates the size based on the number of use cases in a system. Each use case is an interaction that the user may have with the system.

For instance, an e-commerce platform may have 15 use cases: user registration, product search, checkout, among others. The use case points will be able to evaluate the difficulty and size of the software.

1. Story Points:

In Agile, the size of user stories is measured by story points based on their complexity and the effort required.

For instance, if a team estimates that adding a payment feature requires 8 story points and implementing the functionality of search requires 5 story points, these are the points that give an estimation regarding the overall size of the software product.

The size of a software product needs to be understood for several reasons, including the following:

Resource Allocation by understanding the size aids in resource allocation by knowing how many developers and what budget should be utilized to keep the project on track.

Project Estimation giving correct estimation of size helps in estimating the timeline and cost, which is very important for project planning.

Performance Measurement it helps teams measure productivity and performance over time, enabling them to identify areas for improvement.

c)

Software reliability is the capability of a software system to perform its functions under specified conditions for a period either without failure or within an acceptable level of failure. In Uganda, where technology is rapidly developing, software reliability is considered key, especially in sectors like banking and healthcare. Software reliability is related to hardware reliability in several ways:

1. Interdependence:

Software runs on hardware; any hardware failure can affect the performance of software. In Ugandan banks for example in Dfcu bank, for instance, if the servers hosting banking software fail due to some hardware failures, customers may fail to access their accounts and/or undertake transactions. This might affect customers' trust and the reputation of the bank as a result.

1. Testing and Validation:

Software as well as hardware must be tested for reliability. Think of a medical device used in Ugandan hospitals like an electrocardiogram that depends on software and hardware components. If the software that controls a patient monitoring system is not tested for reliability, it may fail even when the hardware part is working and thus may put patients in danger.

1. Error Propagation:

Hardware errors may be propagated through to software errors. For example, a computer used within a Ugandan school called Tibah internationals entry system has faulty RAM; it may result in some educational software giving incorrect results or crashing, thus disrupting learning.

1. Performance Metrics:

The metrics for measuring performance for software and hardware are just about similar.

For example, Mean Time Between Failures for a Ugandan mobile phone application, say, at 1,000 hours implies that this is the time it will operate on average before it develops a failure. If the hardware it runs on, say mobile servers, has much lower MTBF, it follows that the overall system reliability is poor.

In an ATM system, hardware reliability means the functioning of components such as a card reader and cash dispenser; software reliability involves processing the transactions correctly. If one of these components fails-the software incorrectly calculates the balance or the hardware cannot dispense cash-then the entire ATM system becomes unreliable.

In summary, software reliability and hardware reliability are interrelated. Ensuring both are robust is essential for the overall performance and dependability of systems in Uganda.

No 6

a)

i. Quality Control

Quality control is the process of guaranteeing that the software product meets the predetermined quality standard and requirements. QC tests and validates to discover and eliminate defects to make sure that the app works as expected before rollout.

Take a case such as Jumia Uganda, one of the popular Ugandan shopping apps. Quality control ensures that :

Testing Functionalities: Features like product searches, adding items to a cart, and payment processing are tested for accuracy and usability.

User Experience (UX): The app is tested for responsiveness on different devices, such as low-end smartphones, ensuring it runs smoothly even with limited internet speeds.

Bug Fixing: Issues like crashes during checkout or incorrect price calculations are identified and resolved before launching updates.

For example, before allowing users to filter products by "Made in Uganda," Jumia Uganda does rigorous quality control in testing the filter functionality to show relevant local products correctly.

ii. Software Re-Engineering

Software re-engineering involves modifying an existing software system to improve its performance, maintainability, or adapt it to new requirements without changing its core functionalities. It is often undertaken when the software becomes outdated or inefficient.

For Jumia Uganda, software re-engineering may involve:

Refactoring the code to improve performance. This could be related to improving page load times, especially during high-traffic sales like "Black Friday."

Integrating features for mobile money payment options, such as MTN MoMo Pay and Airtel Money, since many people in Uganda depend on these services.

Re-engineering the app to accommodate increased stringency in data privacy. This will ensure that user information is handled securely.

It could re-engineer the Jumia Uganda app to integrate AI recommendations, hence advising on the product one is likely to buy, based on one's history. This will make it more competitive and user-friendly, yet retaining its core functions of shopping.

Both quality control and software re-engineering are essential for maintaining the performance and relevance of shopping apps like Jumia Uganda in a competitive and evolving market.

b)

A fact-finding technique suitable for collecting data from the FST students at VU would be through surveys.

A survey is a structured data collection technique where a series of questions are posed to respondents to gather information about their preferences, experiences, and opinions. It is suitable for collecting data from a large group efficiently and systematically.

To implement this technique, a well-structured questionnaire can be designed, including a mix of closed-ended questions, like multiple-choice or Likert scale questions, and open-ended questions that encourage students to provide detailed feedback. For instance, questions could focus on their satisfaction with current resources, areas for improvement, and suggestions for new services or support.

Surveys can be distributed online through tools such as Google Forms or SurveyMonkey, making it easy for students to participate at their convenience. Also, anonymity can be guaranteed for honest responses and, therefore, more valid data collection. Analysing the results of the survey may yield valuable insights that can be used in improving the educational experience for FST students at VU. To assess student satisfaction with the new lab equipment, a survey can be conducted asking FST students to rate their experience on a scale of 1-5 and provide suggestions for further improvement.

Surveys are a practical and effective fact-finding technique for VU's FST students. They ensure a broad reach, provide quantitative and qualitative data, and support evidence-based decision-making for improvements.

c)

As a project manager working on the implementation of a system to automatically manage processes for final year students undertaking their projects, I would need information on the following:

1. Student Information:

A record of the students is required, showing their names, contact details, year of study, and areas of interest; this helps in the project assignment to suit their skills and preferences.

1. Project Topics:

Gather information on various topics or areas of research for the project. It will include existing topics, new ideas, and emerging trends in the field of study.

1. Project Requirements:

Specify the requirements of each project regarding objectives, expected outcomes, resources required, and guidelines, if any, provided by the faculty.

1. Timeline and Milestones:

Indicate the timeline for project completion by mentioning key milestones like proposal submissions, reports on progress, and final presentations. This helps in the tracking of each project.

1. Supervisor Assignments:

List all faculty supervisors, indicating their research expertise and their current supervisory load, to recommend suitable supervisors for each student needing guidance.

1. Resource Availability:

Identify available resources, such as laboratories, equipment, software, and libraries, that students may need to complete their projects.

1. Budget Considerations:

If applicable, gather information about the budget for projects, including funding sources, allowable expenses, and any financial constraints.

1. Feedback Mechanisms:

Develop a system for collecting feedback from students and supervisors throughout the project lifecycle to identify areas for improvement and ensure quality.

1. Assessment Criteria:

Establish the criteria for project assessment, including grading rubrics and weightage of various components, such as research, presentation, and report writing.

1. Opportunities for Collaboration:

Provide information on any opportunities available for student collaboration, including group projects or industry/other department collaborations.

1. Risk Management:

Highlight the potential risks of projects, including time management and resource shortage, and propose mitigation strategies.

1. Post-Project Evaluation:

Intend to collect data about the post-project outcomes in terms of student reflections, supervisor evaluations, and general learning impact.

The gathered information will enable the system to manage the processes involved effectively to ensure final-year project students are well-guided while working on their projects.