Q1a. WHY WATERFALL MODEL IS LIMITED TO ITERATION

Sequential nature: The waterfall model follows a strict sequence of phases, such as requirements gathering, design, implementation, testing, and maintenance. Each phase depends on the completion of the previous one. This sequential approach makes it difficult to incorporate feedback and changes once a phase is completed.

High cost of change: Making changes late in the development process in the waterfall model can be costly and time-consuming. Since each phase builds upon the outputs of the previous phase, changes in one phase often require rework in subsequent phases. This can lead to delays and increased project costs.

Q1b. DIFFERENT TYPES OF APPLICATION SOFTWARE

i). Stand-alone applications These are application systems that run on a local computer, such as a PC. They include all necessary functionality and do not need to be connected to a network. Examples of such applications are office applications on a PC, CAD programs, photo manipulation software, etc.

ii). Interactive transaction-based applications These are applications that execute on a remote computer and that are accessed by users from their own PCs or terminals. Obviously, these include web applications such as e-commerce applications where you can interact with a remote system to buy goods and services. This class of application also includes business systems, where a business provides access to its systems through a web browser or special-purpose client program and cloud-based services, such as mail and photo sharing. Interactive applications often incorporate a large data store that is accessed and updated in each transaction.

iii). Embedded control systems These are software control systems that control and manage hardware devices. Numerically, there are probably more embedded systems than any other type of system. Examples of embedded systems include the software in a mobile (cell) phone, software that controls anti-lock braking in a car, and software in a microwave oven to control the cooking process.

iv). Batch processing systems These are business systems that are designed to process data in large batches. They process large numbers of individual inputs to create corresponding outputs. Examples of batch systems include periodic billing systems, such as phone billing systems, and salary payment systems.

v. Entertainment systems These are systems that are primarily for personal use and which are intended to entertain the user. Most of these systems are games of one kind or another. The quality of the user interaction offered is the most important distinguishing characteristic of entertainment systems.

Q1c. DISTINGUISH BETWEEN WATERFALL MODEL AND INCREMENTAL MODEL WITH EXAMPLES OF SOFTWARE APPLICATION

The waterfall model is a linear and sequential approach to software development, where each phase flows downward like a waterfall, and progress is made from one phase to the next in a predefined order (requirements, design, implementation, testing, deployment, maintenance). example Application**:** Enterprise Resource Planning (ERP) System while the incremental model breaks down the software development process into smaller, incremental builds or iterations. Each iteration adds new functionality to the system, building upon the previous increments, and allowing for feedback and adjustments throughout the development lifecycle. Example Application**:** Mobile Application Development

Q1d. CONFLICT MIGHT ARISE WHEN DESIGN AN ARCHITECTURE FOR WHICH BOTH AVAILABILITY AND SECURITY REQUIREMENT

Designing an architecture to meet both availability and security requirements can be challenging due to potential conflicts between these critical non-functional requirements. Ensuring high availability often involves redundant systems and failover mechanisms, while robust security measures require encryption, authentication, and access controls. However, implementing security measures can introduce complexity and overhead that may impact system performance and availability. Resource allocation, managing complexity, and balancing risk further compound these challenges. Striking the right balance between availability and security requires careful trade-offs, resource management, and risk assessment to achieve an optimal architecture that meets both requirements effectively.

Q2. a.i A SYSTEM TO CONTROL ANTI-LOCK BRAKING IN A CAR:

Incremental model: For developing an anti-lock braking system (ABS) for a car, the Incremental model is the most appropriate software process model. This model allows for iterative development, enabling continuous refinement and improvement of the ABS functionality over multiple iterations. Early feedback from stakeholders ensures that potential issues are addressed promptly, enhancing the system's reliability and safety. Additionally, the Incremental model facilitates risk management by allowing for the incremental identification and mitigation of potential risks throughout the development lifecycle. Its flexibility and adaptability accommodate changes in requirements or technological advancements, ensuring that the ABS remains aligned with industry standards and evolving needs. Continuous improvement through iterative enhancements ensures the ongoing optimization of the ABS to meet the dynamic demands of automotive safety.

Q2a ii. A VIRTUAL REALITY SYSTEM TO SUPPORT SOFTWARE MAINTENANCE:

Agile approach: For a virtual reality (VR) system supporting software maintenance, the Agile software development process model would be the most appropriate choice. Agile's iterative and incremental approach allows for continuous improvement, adapting to evolving maintenance needs. Its emphasis on collaboration facilitates close interaction between developers, maintenance personnel, and end-users, ensuring alignment of goals and effective coordination of activities. Agile's rapid prototyping and feedback loops enable quick validation of design assumptions and user input, leading to a VR system that meets maintenance needs efficiently. Additionally, Agile's focus on continuous improvement ensures ongoing optimization of the system's performance and usability in line with changing maintenance practices and technologies.

Q2a iii. A UNIVERSITY ACCOUNTING SYSTEM THAT REPLACES AN EXISTING SYSTEM:

Incremental model: For replacing an existing university accounting system, the incremental software process model is the most appropriate choice. This approach involves delivering the new system in stages, gradually replacing modules or features of the old system with functional components of the new one. Incremental delivery allows for continuous feedback from users and stakeholders, ensuring that the new system aligns with the university's accounting needs. It also mitigates risks associated with data migration and user adoption by addressing issues incrementally. Additionally, the incremental model offers flexibility to adapt to changing requirements and integrates smoothly with existing systems, facilitating a seamless transition to the new accounting system**.**

Q2a iv AN INTERACTIVE TRAVEL PLANNING SYSTEM THAT HELPS USERS PLAN JOURNEY WITH THE LOWEST ENVIRONMENT IMPACT:

Agile Approach: The Agile approach is the most appropriate software process model for developing an interactive travel planning system aimed at minimizing environmental impact. Agile methodologies prioritize user involvement and feedback, ensuring that the system evolves iteratively to meet user needs and environmental goals effectively. Its flexibility allows for rapid adaptation to emerging requirements and opportunities in sustainable travel practices. Incremental delivery of features provides immediate value to users while continuously improving the system's environmental impact assessment and recommendation capabilities. Continuous improvement through regular retrospectives ensures that the system evolves in response to user feedback and changing environmental priorities, enhancing its effectiveness in promoting sustainable travel practices.

Q2b PRINCIPAL STAGES OF REQUIREMENT ENGINEERING PROCESS WITH SUITABLE DIAGRAM

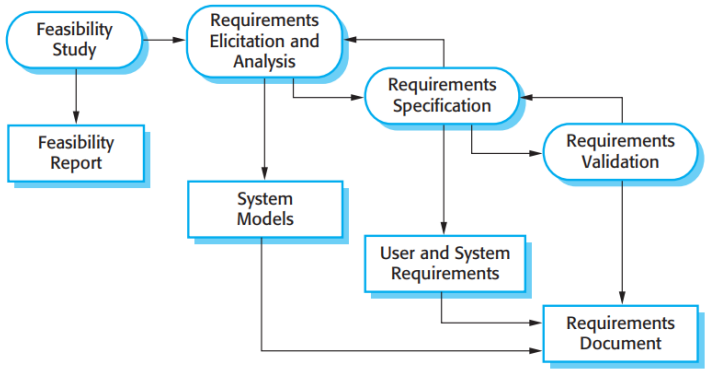
Feasibility study: An estimate is made of whether the identified user needs may be satisfied using current software and hardware technologies. The study considers whether the proposed system will be cost-effective from a business point of view and if it can be developed within existing budgetary constraints. A feasibility study should be relatively cheap and quick. The result should inform the decision of whether or not to go ahead with a more detailed analysis.

Requirements elicitation and analysis This is the process of deriving the system requirements through observation of existing systems, discussions with potential users and procurers, task analysis, and so on. This may involve the development of one or more system models and prototypes. These help you understand the system to be specified.

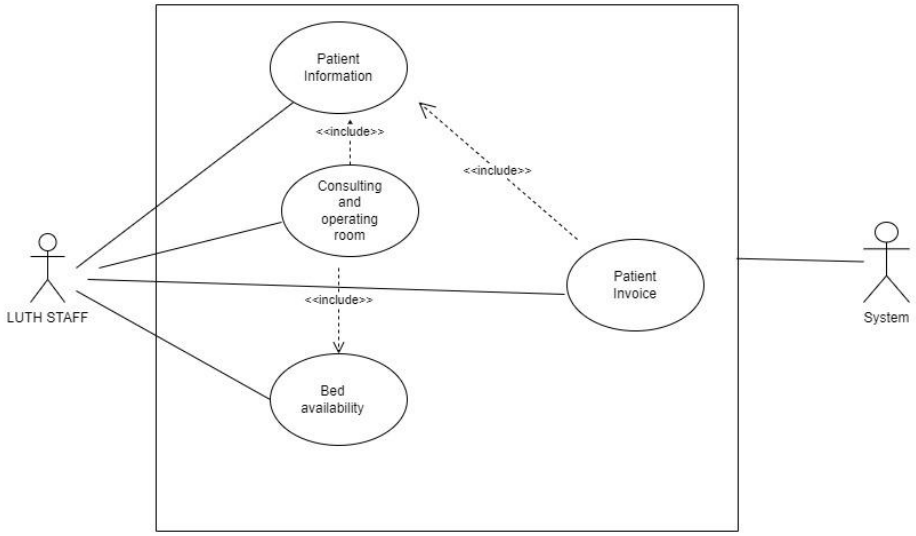
Requirements specification Requirements specification is the activity of translating the information gathered during the analysis activity into a document that defines a set of requirements. Two types of requirements may be included in this document. User

requirements are abstract statements of the system requirements for the customer and end-user of the system; system requirements are a more detailed description of the functionality to be provided.

Requirements validation This activity checks the requirements for realism, consistency, and completeness. During this process, errors in the requirements document are inevitably discovered. It must then be modified to correct these problems.



Q3. USE CASE FOR LUTH



Q3b INCREMENTAL MODEL

Incremental approach is most approach because each service will be created, tested and evaluated by users before other features are also designed. this allows for maximum user cooperation and requirement determination. The stages of the incremental model are as follows:

Specification (Initial Version): This phase involves gathering and defining the requirements for the software. The goal is to create a clear and comprehensive specification that outlines what the software is supposed to achieve.

Development (Intermediate Versions): In this phase, the software is developed in increments or smaller iterations. Instead of creating the entire system at once, developers work on smaller, manageable portions. Each increment typically adds new functionality or refines existing features.

Validation (Final Version): After each increment is developed, it undergoes validation to ensure that it meets the specified requirements and functions as intended. This phase involves testing and quality assurance to identify and fix any issues.

Repeat (Cycle): The development, validation, and specification phases are repeated for each increment until the complete system is built. Each cycle adds new features or refines existing ones based on feedback from previous increments.

Q3c. TYPES OF SOFTWARE FOR LUTH HMS

Information System: This system will take inputs such as patient data, bed availability data, and other information from users (LUTH staff) and store them in the system's database. It should also be able to output this data to users from the database. This characteristic mentioned here is a clear example of an information system.

Data collection system: As a data collection system, this system will integrate with blood analyzers and other laboratory devices to capture patient data, facilitating remote upload directly into the patient medical record database.

Q3d. NON-FUNCTIONAL ATTRIBUTE FOR LUTH HMS

security: the system should handle sensitive patient information, including medical records and personal data. Ensuring robust security measures, such as encryption, access controls, and secure authentication, is essential to protect patient privacy and comply with regulations like HIPAA (Health Insurance Portability and Accountability Act).

Reliability: The system must be highly reliable to support critical healthcare operations. Unplanned downtime or system failures can disrupt essential functions. High reliability ensures uninterrupted access to patient records and services, minimizing the risk of adverse impacts on patient care

Performance: The system deal with a large volume of transactions and data, including patient information, bed availability, and patient invoices. The system must perform efficiently to handle these tasks in a timely manner, ensuring smooth operations and minimizing delays in patient care.

Usability: The system should have a user-friendly interface and easy-to-use features that will enable the users find their way around the system at a fast pace and minimize errors in data entry and retrieval.