SEPM CAE 1

1. Explain pros and cons of Waterfall model.

The Waterfall model is a linear and sequential approach to software development, where each phase must be completed before the next begins.

Pros:

- **Simplicity and Ease of Use:** The straightforward and easy-to-understand nature of the Waterfall model makes it simple to manage and implement.
- **Structured Approach:** Clear, well-defined stages make it easy to track progress and ensure that all requirements are met before moving on to the next phase.
- **Documentation:** Extensive documentation produced at each stage helps in maintaining clarity and serves as a reference for future maintenance.
- **Early Detection of Issues:** Problems can be identified early in the lifecycle, during the requirements and design phases.

Cons:

- **Inflexibility:** Once a phase is completed, it is difficult to go back and make changes, making it less suitable for projects with evolving requirements.
- **Delayed Testing:** Testing is often left until the final stages, which can lead to late discovery of critical issues.
- **Risk of Misalignment:** If initial requirements are not accurately captured, the project may not meet stakeholder expectations, resulting in significant rework.
- **Long Development Time:** The linear nature can lead to longer development times, as each phase must be completed before the next one begins.

2. Summarize the need and importance of SE.

Software Engineering (SE) is crucial for the systematic development, operation, and maintenance of software systems.

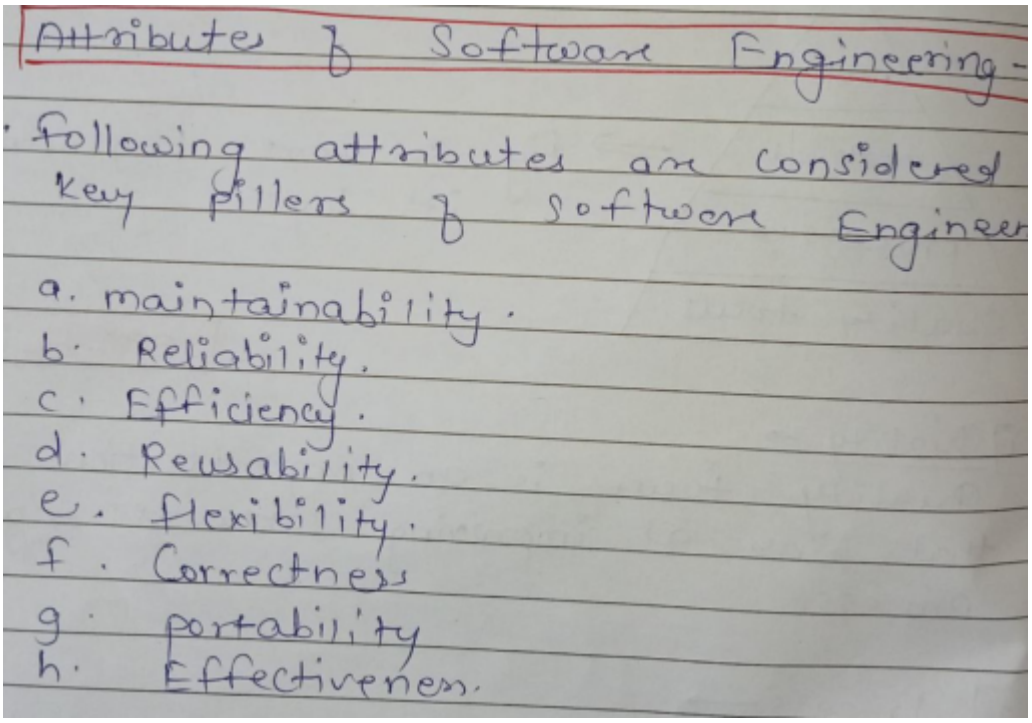
Need:

- **Complexity Management:** SE helps manage the complexity of software systems through structured approaches, ensuring high quality and reliability.

- **Resource Optimization:** Efficient use of resources, including time, money, and personnel, is achieved through systematic planning and execution.
- **Quality Assurance:** Ensures that software meets specified requirements, performs reliably, and is maintainable and scalable.
- **Scalability and Maintenance:** SE ensures that the software is scalable and maintainable. As user needs evolve, the software can be updated and extended without significant issues, ensuring longevity and relevance.
- **Risk Management:** SE involves identifying, assessing, and mitigating risks early in the development process. This proactive approach helps in preventing potential issues and minimizing their impact on the project.

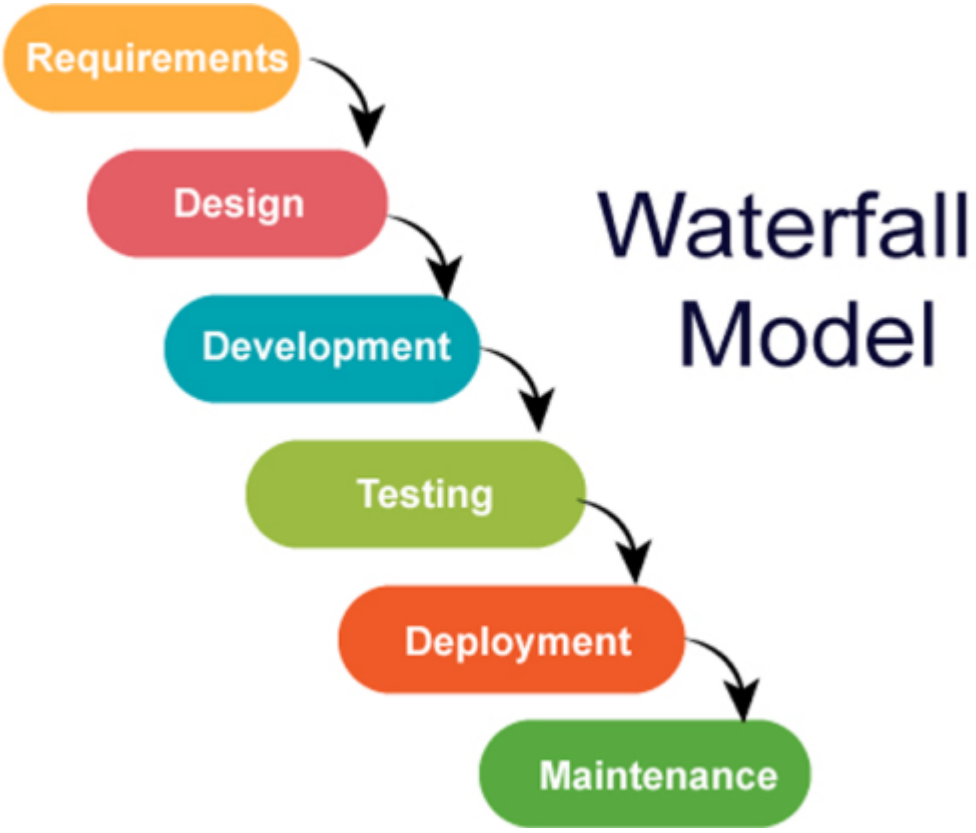
Importance:

- **Enhances Productivity:** Well-defined processes and methodologies enhance team productivity and streamline development activities.
- **Reduces Costs:** Early detection and resolution of issues reduce the overall cost of development and maintenance.
- **Ensures User Satisfaction:** By meeting user requirements and expectations, SE ensures that the end product is useful and effective.
- **Supports Maintenance:** Facilitates easier maintenance and updates through comprehensive documentation and modular design.
- **Quality Assurance:**SE incorporates various quality assurance practices, such as testing and code reviews, ensuring that the software is reliable, performs well, and meets industry standards.



3. List and explain the steps in SDLC.

The Software Development Life Cycle (SDLC) is a process used for planning, creating, testing, and deploying an information system.

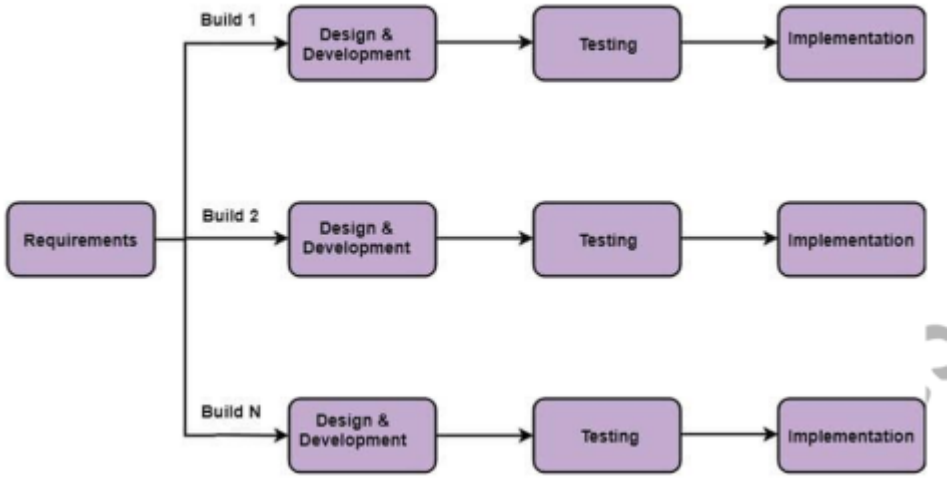


Steps:

1. **Requirement Analysis:** Gathering and documenting user requirements and business needs. It involves stakeholder interviews, surveys, and creating requirement specifications.
2. **System Design:** Translating requirements into a blueprint for constructing the software. This phase includes architectural design, interface design, and detailed design specifications.
3. **Implementation (Coding):** Actual coding and development of the software based on the design documents. This phase involves selecting appropriate programming languages, tools, and techniques.
4. **Testing:** Verifying that the software functions as intended and identifying any defects. Includes unit testing, integration testing, system testing, and user acceptance testing.
5. **Deployment:** Installing and configuring the software in a production environment. This step includes user training, documentation, and system rollout.
6. **Maintenance:** Ongoing support and enhancement of the software. This involves fixing bugs, updating features, and ensuring the system adapts to changing requirements.

4. Explain incremental model in detail.

Incremental Model



- incremental model is Widely accepted model of software development process, where the software requirements are broken down into multiple separate modules/increments in SDLC.
- Each increment is treated as sub-project & goes through all phases of SDLC incremental model.
- In this type of model instead of making goal achievement, we achieve/ divide in small steps.
- Incremental model is used when :

  a) Requirements are clearly specified & are known up-front. Certain requirements however require time.

  b) Product based companies develop their product.

  c) Projects have long development schedule.
- Phases of incremental model:

  Requirement phase

  Design & development phase

  Testing phase

  implementation
- **Advantages:**

1. **Flexibility:** Easier to incorporate changes and new requirements.
2. **Risk Management:** Reduces risk by delivering smaller, manageable parts.
3. **User Involvement:** Continuous user feedback ensures alignment with user needs and expectations.
4. **Faster Delivery:** Parts of the system can be delivered and used earlier.

- **Disadvantages:**

1. **Integration Challenges:** Integrating increments can be complex and require significant effort.
2. **Resource Intensive:** Requires continuous user involvement and thorough testing of each increment.
3. **Scope Creep:** Risk of scope creep if user feedback leads to frequent changes and additions.

5. Explain Function Point in detail.

Function Point (FP) is a metric used to measure the functionality delivered by a software system, independent of the technology used to implement it.

Key Concepts:

- **Function Types:** FPs are categorized into five types: External Inputs (EI), External Outputs (EO), External Inquiries (EQ), Internal Logical Files (ILF), and External Interface Files (EIF).
- **Complexity Factors:** Each function type is assigned a complexity weight (low, average, high) based on its characteristics and interactions.
- **FP Calculation:** The total FP count is calculated by multiplying the number of functions by their respective weights and summing them up.

Steps in FP Calculation:

1. **Identify Functions:** Identify all functions within the system based on the five types.
2. **Assign Complexity Weights:** Assign a complexity weight to each function based on predefined criteria.
3. **Calculate Unadjusted FP (UFP):** Sum the weighted function counts to obtain the UFP.
4. **Determine Value Adjustment Factor (VAF):** Assess the system's 14 general characteristics (such as performance, reusability, etc.) and calculate the VAF.
5. **Calculate Adjusted FP:** Multiply the UFP by the VAF to obtain the adjusted FP.

Advantages:

- **Technology Independent:** Provides a consistent measure of functionality, regardless of implementation technology.
- **Early Estimation:** Enables early estimation of project size, effort, and cost.
- **Focus on Functionality:** Measures the functionality delivered to the user.
- **Benchmarking:** Facilitates comparison of productivity and quality across different projects and organizations.

Disadvantages:

- **Complexity:** More complex to measure and requires detailed analysis.
- **Subjectivity: involve subjective judgement which can vary btw evaluator**
- **Learning Curve:** Requires training and experience to accurately apply the FP method.
- **Detailed Analysis:** Necessitates detailed analysis of system requirements and functions.

6. Compare LOC and FP based estimation

Lines of Code (LOC) Estimation:

- **Definition:** LOC measures the size of a software application by counting the number of lines in the source code.
- **Usage:** Primarily used for estimating the size of software, assessing developer productivity, and calculating project effort and cost.
- **Advantages:**
    1. **Simplicity:** Easy to measure and understand.
    2. **Historical Data:** Many organizations have historical data for LOC, making it easier to use for future projects.
    3. **Direct Measurement:** Directly relates to the amount of code written.
- **Disadvantages:**
    1. **Language Dependency:** Different programming languages have different levels of verbosity, making comparisons difficult.
    2. **Quality Ignored:** Focuses on quantity of code rather than quality.
    3. **Encourages Bloat:** May encourage developers to write more code than necessary to meet targets.

Function Point (FP) Estimation:

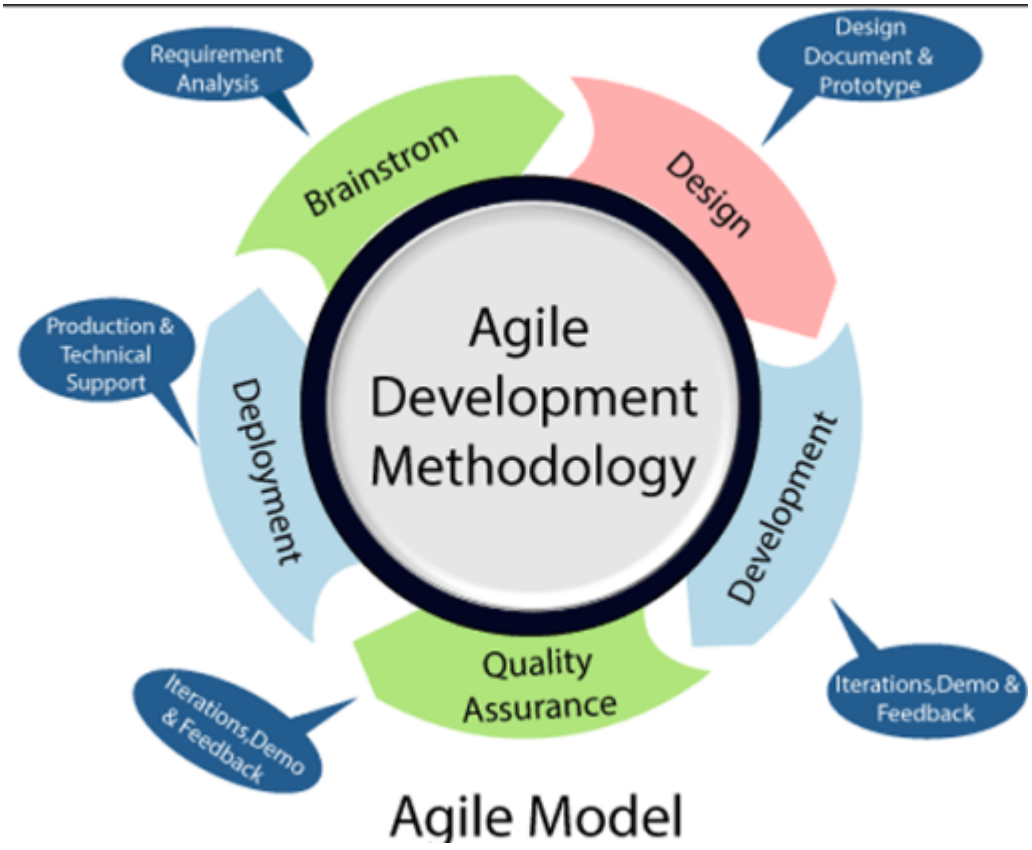
- **Definition:** FP measures the functionality delivered by the software, independent of the technology or code used.
- **Usage:** Used for estimating project size, effort, cost, and productivity based on the functionality provided.
- **Advantages:**
    1. **Technology Agnostic:** Independent of programming language and technology.
    2. **Focus on Functionality:** Measures the functionality delivered to the user.

3. **Early Estimation:** Can be estimated early in the project lifecycle, during the requirement analysis phase.

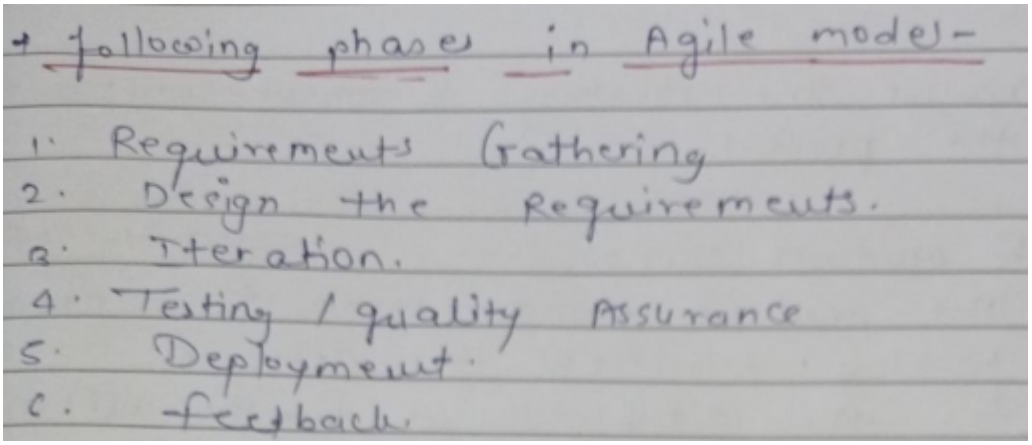
- **Disadvantages:**
  1. **Complexity:** More complex to measure and requires detailed analysis.
  2. **Subjectivity:** Involves subjective judgment, which can vary between evaluators.
  3. **Learning Curve:** Requires training and experience to accurately apply the FP method.

7. Explain Agile Model

- The Agile model is an iterative and incremental approach to software development that emphasizes flexibility, collaboration, and customer feedback.
- the entire project is divided into small incremental builds. All of these builds are provided in iterations



Agile Model



Advantages:

- **Customer Satisfaction:** Continuous delivery of valuable software ensures that customer needs are met.
- **Flexibility:** Easily accommodates changes in requirements.
- **Risk Reduction:** Frequent reviews and iterations help identify and address issues early.
- **Improved Quality:** Continuous testing and integration ensure high-quality deliverables.

8. Explain Risk in Risk Management

Risk in risk management refers to the potential for an adverse event or condition that could impact the success of a project.

Key Concepts:

- **Risk Identification:** The process of identifying potential risks that could affect the project. This includes both internal and external risks.
- **Risk Analysis:** Assessing the likelihood and impact of identified risks. This helps prioritize which risks need immediate attention.
- **Risk Mitigation:** Developing strategies to reduce the likelihood or impact of risks. This can include preventive measures and contingency plans.
- **Risk Monitoring:** Continuously tracking identified risks and identifying new risks throughout the project lifecycle. This ensures that risk management strategies are effective and up-to-date.
- **Risk Response:** Implementing risk mitigation strategies when a risk event occurs. This involves executing contingency plans and making necessary adjustments to the project plan.

Examples of Risks:

- **Technical Risks:** Issues related to technology, such as integration problems, technology changes, or technical failures.
- **Project Management Risks:** Risks related to project planning and execution, such as schedule delays, budget overruns, or resource shortages.
- **External Risks:** Risks originating outside the project, such as market changes, regulatory changes, or supplier issues.
- **Organizational Risks:** Risks within the organization, such as changes in management, organizational restructuring, or employee turnover

9. Explain Any Specialized Model

Spiral Model

- Spiral model supports risk handling.
- The exact No. of loops in the spiral is unknown & can vary from project to project.
- Each loop of the spiral is called phase of SDLC.
- - Radius of the spiral at any point represents expenses (cost) of the project so far.
- - Angular dimensions represent progress made so far in the current phase.
- phase 1:

1. - Objectives are investigated, elaborated & analyzed.
2. - Possible alternative solutions are proposed.

- *Phase 2: By developing an appropriate prototype, best possible alternative solutions are evaluated.
- *Phase 3 : - Developing & verifying the next level product.
- *Phase 4:*: Reviewing the result of stages traversed so far with the cost & planning the next iteration.

Advantages:

1. Flexible to reuse.
2. Gives confidence in the project.
3. risk management

Disadvantages:

1. Knowledgeable & experienced staff is required.
2. For developers, it is complicated & risk-driven.
3.cost

10. Explain Project Scheduling Process

Project scheduling is the process of defining and arranging project activities in a timeline to ensure project goals are met within the specified timeframe.

Key Steps:

1. **Define Activities:** Break down the project into smaller tasks or activities. This involves creating a Work Breakdown Structure (WBS) to identify all necessary tasks.
2. **Sequence Activities:** Determine the order in which tasks need to be performed. This involves identifying dependencies between tasks and creating a project network diagram.
3. **Estimate Duration:** Estimate the time required to complete each task. This can be done using expert judgment, historical data, or estimation techniques like PERT (Program Evaluation and Review Technique).
4. **Develop Schedule:** Create a project schedule by assigning start and end dates to each task. This involves using scheduling tools like Gantt charts or project management software.
5. **Allocate Resources:** Assign resources (such as personnel, equipment, and materials) to tasks. Ensure that resources are available and properly allocated to avoid conflicts and delays.
6. **Monitor and Control:** Continuously monitor the project schedule to track progress and make adjustments as needed. This involves updating the schedule, managing changes, and addressing any deviations from the plan.

Techniques:

- **Gantt Charts:** Visual representation of the project schedule, showing tasks along a timeline.
- **Critical Path Method (CPM):** Identifies the longest path of dependent tasks in the project and highlights the critical tasks that determine the project duration.
- **PERT:** Uses optimistic, pessimistic, and most likely estimates to calculate expected task durations and assess schedule uncertainty.

Advantages:

- **Clarity:** Provides a clear timeline and roadmap for project activities.
- **Resource Management:** Helps in efficient allocation and utilization of resources.
- **Risk Identification:** Identifies potential schedule risks and bottlenecks early in the process.

Disadvantages:

- **Complexity:** Creating and maintaining a detailed project schedule can be complex and time-consuming.
- **Rigidity:** Strict adherence to the schedule may reduce flexibility in handling unexpected changes.
- **Dependency Management:** Managing dependencies and ensuring that tasks are completed on time can be challenging.

11. Describe software requirements in detail

Software requirements are detailed descriptions of the system's capabilities, constraints, and interactions. They serve as the foundation for software design, development, and testing, ensuring that the final product meets the needs and expectations of stakeholders.

Types of Software Requirements:

1. **Functional Requirements:**
    - Define the specific behavior or functions of the system.
    - Describe what the system should do, including tasks, functions, and operations.
    - Examples: User authentication, data processing, report generation.
2. **Non-functional Requirements:**
    - Define the quality attributes, performance criteria, and constraints of the system.
    - Describe how the system performs its functions.
    - Examples: Performance, security, usability, reliability, scalability.

Importance of Software Requirements:

- **Clarity and Understanding:** Provide a clear understanding of what the system should do, ensuring that all stakeholders have a common vision.
- **Basis for Design:** Serve as a foundation for system design, guiding developers in creating the system architecture and components.

- **Testing and Validation:** Establish criteria for testing and validation, ensuring that the system meets specified requirements and performs as expected.
- **Project Planning:** Aid in project planning and estimation, helping to determine the scope, timeline, and resources needed for the project.
- **Change Management:** Provide a baseline for managing changes to the system, ensuring that modifications are tracked and evaluated.

Steps in Requirement Engineering:

1. **Requirement Elicitation:** Gathering requirements from stakeholders through interviews, surveys, workshops, and observation.
2. **Requirement Analysis:** Analyzing and refining requirements to ensure they are complete, clear, and feasible.
3. **Requirement Specification:** Documenting the requirements in a detailed and structured manner, typically in a Software Requirement Specification (SRS) document.
4. **Requirement Validation:** Verifying that the requirements accurately represent stakeholder needs and are achievable.
5. **Requirement Management:** Managing changes to requirements throughout the project lifecycle, ensuring that updates are controlled and documented.

12. Distinguish between functional and non-functional requirements
14. Explain user requirement and system requirement

SEPT 1

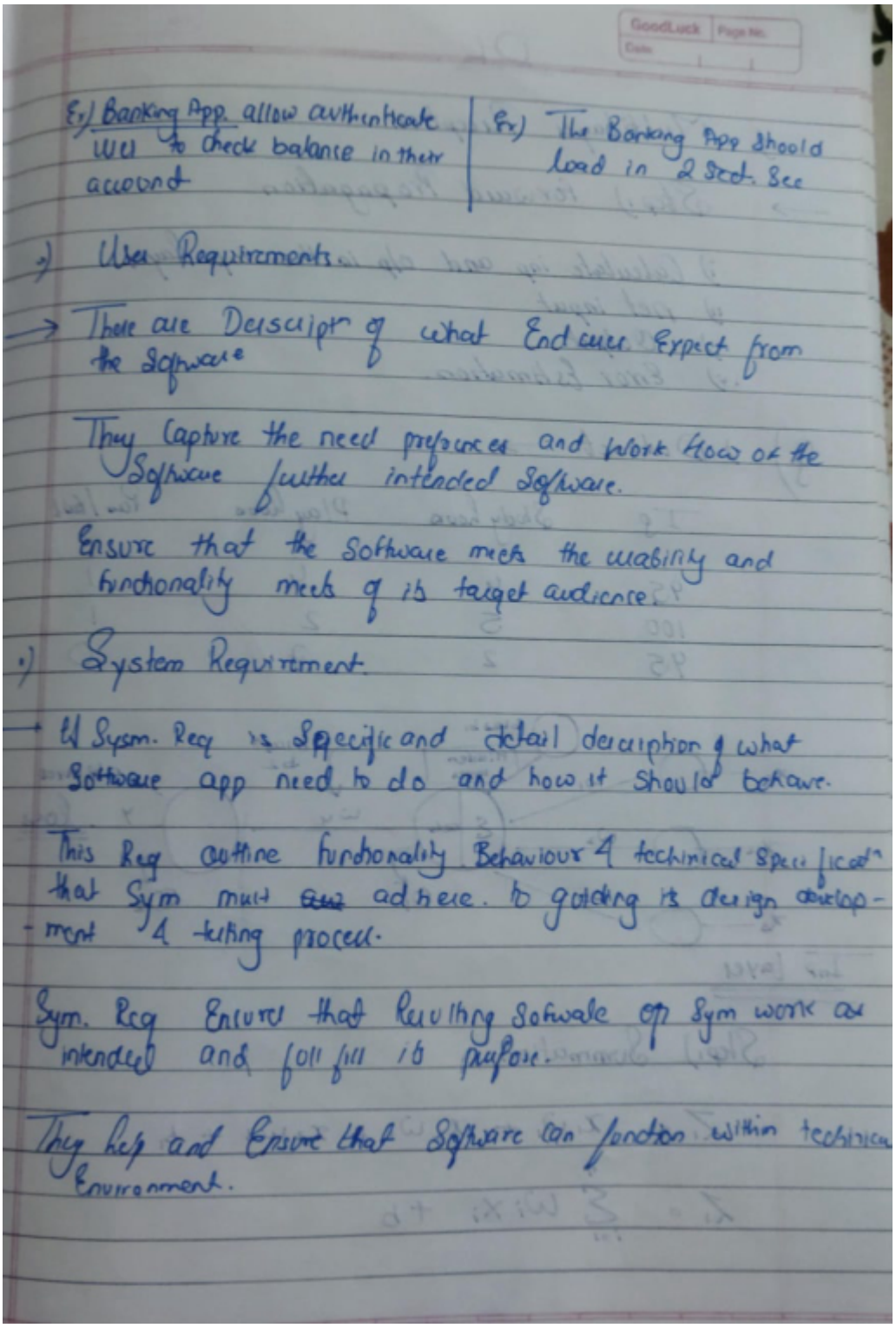
1) Requirements

→ According to IEEE Standards, Req is a cond^n or capability needed by a user to solve a problem or to achieve an objective, a condition or capability must be met or possessed by a system or system component to satisfy standard specification or other formally imposed documents,

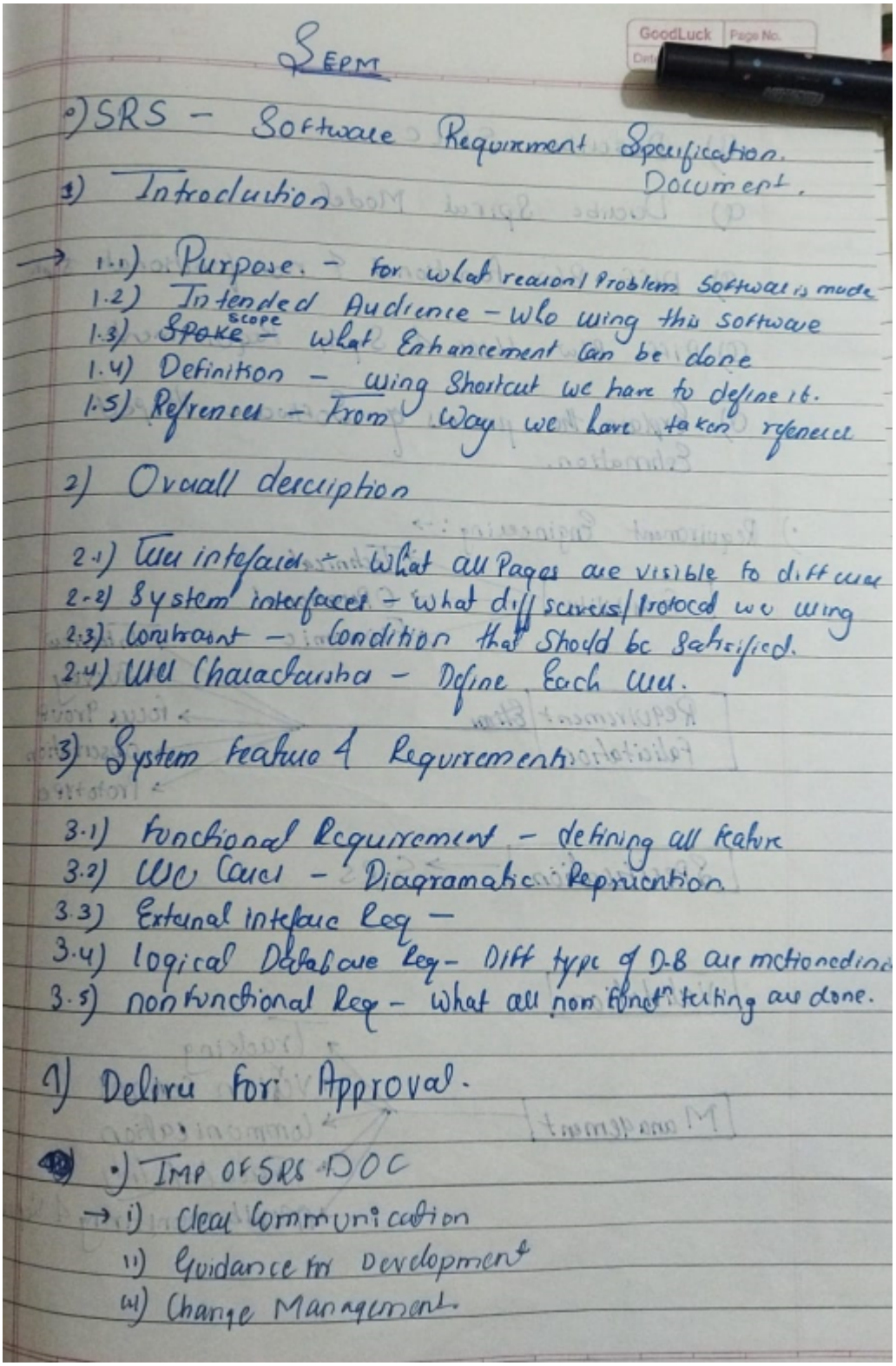
Req should be cleared & well defined.

2) Diff B/w functional & non functional

functional	Non functional.
→ i) Help to understand the feature of the system	i) Help to understand the Performance of system.
ii) There are mandatory Req	ii) Not Mandatory it all dep on Fund^n Req.
iii) There Req will always concern on the user Req.	iii) Concern on Expect^n of user
iv) There Req are specified by user.	iv) Specified by developer or Sys. Architect
v) Describes what the prod does	v) Describes how the prod works
vi) System testing, API, Integrat^n testing are carried out	vi) Usability testing, performance, Stress, Security, test
vii) First we have to perform fund^n testing	vii) Only after performing F.T we can go for non functional testing



13. Explain Software Requirement Document (SRS) in detail



15. Explain the importance of requirement engineering

Requirement engineering is a critical phase in the software development lifecycle, focusing on identifying, analyzing, documenting, and managing the requirements of a software system. Its importance can be highlighted through the following points:

1. **Ensures Alignment with Stakeholder Needs:**
    - By systematically gathering and analyzing requirements, requirement engineering ensures that the final product aligns with the needs and expectations of stakeholders, including users, clients, and regulatory bodies.
2. **Reduces Development Costs:**
    - Early identification and clarification of requirements help in avoiding costly changes and rework later in the development process. Clear requirements lead to more accurate project planning and estimation.
3. **Improves Quality and Performance:**
    - Detailed and well-documented requirements provide a solid foundation for designing and developing high-quality software that performs reliably and meets specified performance criteria.
4. **Facilitates Communication and Collaboration:**
    - Requirement engineering promotes clear communication and collaboration among all stakeholders, including project managers, developers, testers, and users. This shared understanding helps in preventing misunderstandings and misalignments.
5. **Enhances Risk Management:**
    - By identifying potential risks associated with requirements early in the project, requirement engineering helps in developing strategies to mitigate these risks, reducing the likelihood of project failure or delays.
6. **Supports Requirement Changes:**
    - A structured approach to managing requirements changes ensures that any modifications are controlled, evaluated, and documented, maintaining project scope and preventing scope creep.

16. Explain Feasibility Studies in Detail

Feasibility Study:A feasibility study is an analysis conducted to determine whether a proposed project or system is viable and worth pursuing. It assesses various aspects of the project to ensure that it can be successfully completed within constraints such as time, cost, and technical capabilities.

Types of Feasibility Studies:

1. **Technical Feasibility:**
2. **Economic Feasibility:**
3. **Legal Feasibility:**
4. **Operational Feasibility:**
5. **Schedule Feasibility:**

Steps in Conducting a Feasibility Study:

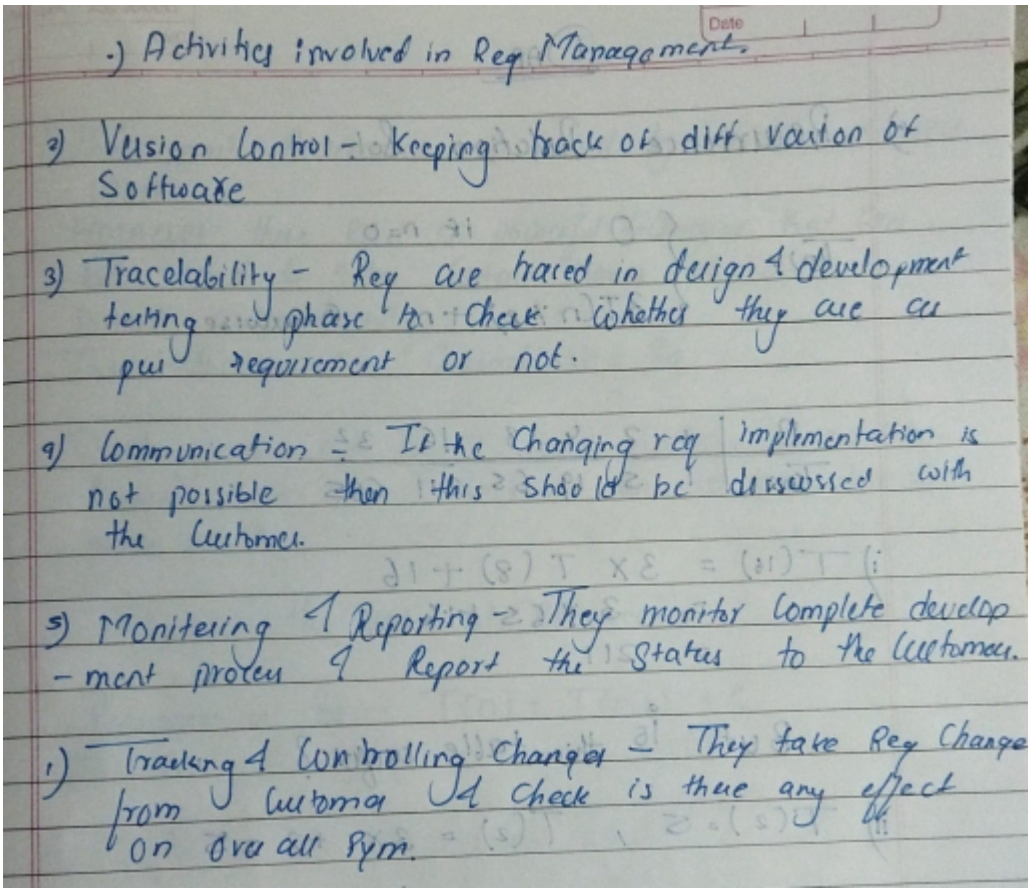
1. **Define the Project Scope:** Clearly outline the objectives, goals, and requirements of the project.
2. **Conduct Preliminary Analysis:** Identify potential obstacles and determine whether there are feasible solutions.
3. **Analyze Resources:** Assess the availability of technical, financial, and human resources.
4. **Evaluate Alternatives:** Consider different approaches and solutions to achieve the project objectives.
5. **Make Recommendations:** Provide recommendations based on the analysis, including whether to proceed with the project, modify it, or abandon it.

Importance of Feasibility Studies:

- **Risk Reduction:** Identifies potential risks and challenges early, allowing for better risk management.
- **Informed Decision-Making:** Provides a comprehensive analysis that helps stakeholders make informed decisions about the project.
- **Resource Optimization:** Ensures that resources are allocated efficiently and effectively.
- **Improved Planning:** Provides a solid foundation for project planning and execution.

17. Explain Requirement Management

- **Requirement Management:**Requirement management is the process of documenting, analyzing, tracing, prioritizing, and managing changes to requirements throughout the project lifecycle.
- It ensures that all stakeholders have a clear understanding of the requirements and that the final product meets those requirements.

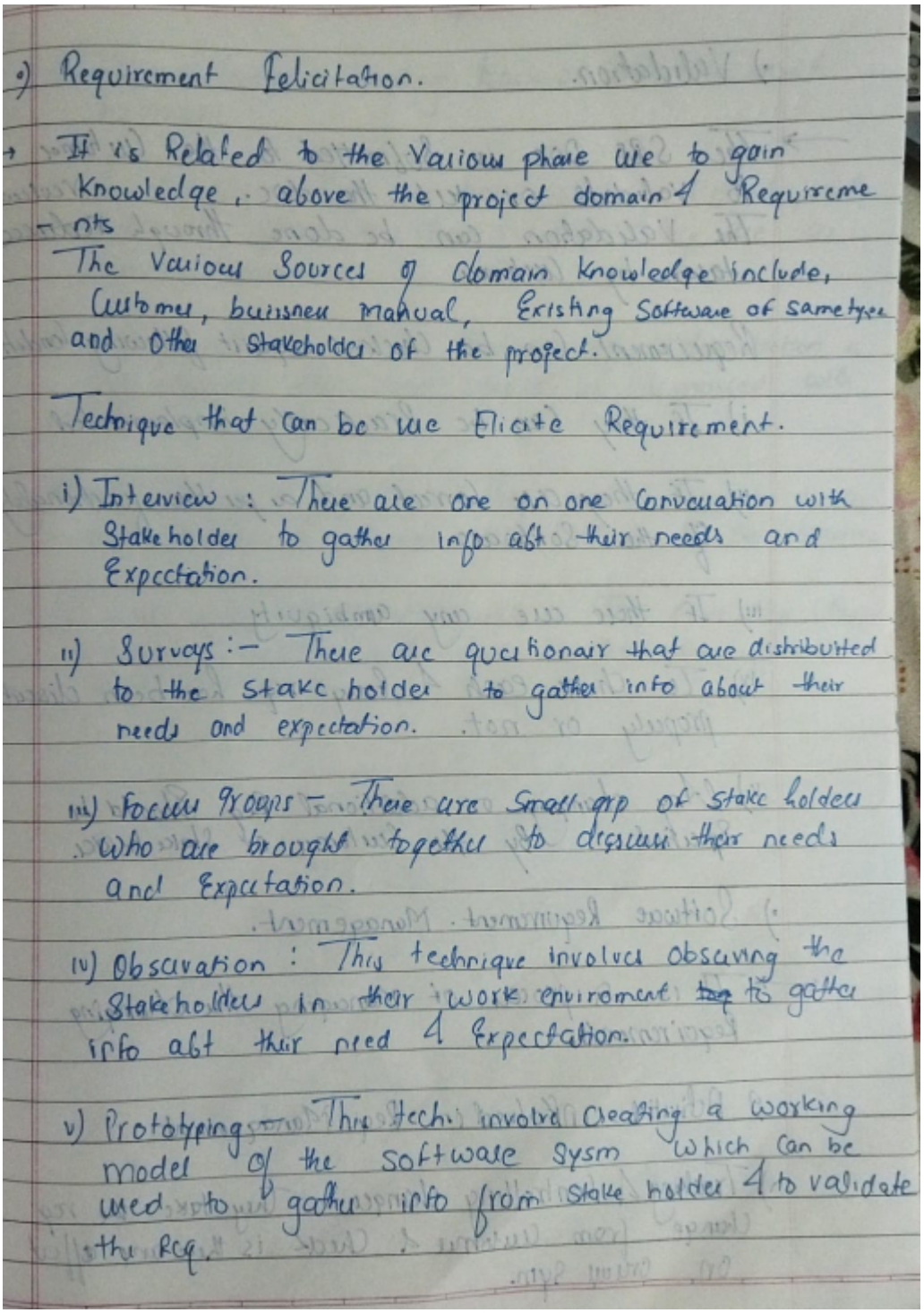


Importance of Requirement Management:

- **Consistency:** Ensures that all project activities are aligned with the requirements, reducing the risk of misunderstandings and misalignments.
- **Quality Control:** Helps in maintaining the quality of the final product by ensuring that all requirements are met and validated.
- **Change Control:** Provides a structured approach to managing requirement changes, preventing scope creep and ensuring that changes are evaluated and documented.
- **Stakeholder Satisfaction:** Ensures that stakeholder needs and expectations are met, leading to higher satisfaction and acceptance of the final product.
- **Project Success:** Enhances the likelihood of project success by ensuring that the project delivers the intended value and meets its objectives.

18. Illustrate Requirement Elicitation and Analysis in Detail

- **Requirement Elicitation:**Requirement elicitation is the process of gathering requirements from stakeholders.
- It involves identifying stakeholders, understanding their needs, and documenting those needs in a clear and detailed manner



a) Requirement Elicitation.

→ It is Related to the Various phase we to gain knowledge, above the project domain & Requirements

The Various Sources of domain knowledge include, Customer, buisness manual, Existing Software of same type and other stakeholder of the project.

Technique that can be use Elicite Requirement.

i) Interview : There are one on one conversation with Stakeholder to gather info abt their needs and Expectation.

ii) Surveys :- There are questionair that are distributed to the stake holder to gather info about their needs and expectation.

iii) Focus groups - There are Small grp of stake holders who are brought together to discuss their needs and Expectation.

iv) Observation : This technique involves observing the Stakeholders in their work enviroment to gather info abt their need & Expectation.

v) Prototyping - This tech. involve creating a working model of the software sysm which can be used to gather info from stake holder & to validate the Req.

19. Discuss Good Requirement Characteristics

Good requirements have specific characteristics that ensure they are clear, complete, and actionable. These characteristics are often referred to using the acronym SMART or similar frameworks.

Characteristics of Good Requirements:

1. **Specific:**
   - **Description:** Requirements should be precise and unambiguous, clearly stating what is needed without leaving room for interpretation.
   - **Example:** "The system shall allow users to reset their passwords using their registered email address."
2. **Measurable:**
   - **Description:** Requirements should be quantifiable, providing criteria that can be used to measure whether the requirement has been met.
   - **Example:** "The system shall process 100 transactions per second."
3. **Achievable:**
   - **Description:** Requirements should be realistic and achievable within the project's constraints, including time, resources, and technology.
   - **Example:** "The system shall support up to 1,000 concurrent users."
4. **Relevant:**
   - **Description:** Requirements should be relevant to the project's objectives and add value to the end-users or stakeholders.
   - **Example:** "The system shall generate monthly sales reports for the management team."
5. **Time-bound:**
   - **Description:** Requirements should include a timeframe or deadline for when they need to be achieved.
   - **Example:** "The system shall be fully operational by the end of Q3 2024."
6. **Complete:**
   - **Description:** Requirements should include all necessary information and context to ensure they are fully understood and actionable.
   - **Example:** "The system shall include a user authentication module that supports login, logout, and password recovery functionalities."
7. **Consistent:**

- **Description:** Requirements should be internally consistent, not conflicting with other requirements or project constraints.
- **Example:** "The system shall encrypt all user data in transit and at rest, consistent with the company's data security policy."

8. **Traceable:**
   - **Description:** Requirements should be traceable, with a clear link to their source and related project artifacts.
   - **Example:** "Requirement ID: 123, Source: User interview on 01/01/2024."
9. **Verifiable:**
   - **Description:** Requirements should be testable, providing a clear way to verify that they have been met.
   - **Example:** "The system shall generate an error message if the user enters an invalid email address during registration."
10. **Understandable:**
    - **Description:** Requirements should be clear and easily understood by all stakeholders, including non-technical users.
    - **Example:** "The system shall provide an intuitive user interface with clear navigation and help options."

20. Importance of Requirement Gathering

Requirement gathering is a critical phase in the software development process, as it lays the foundation for the entire project. Its importance can be highlighted through the following points:

1. **Ensures Alignment with Stakeholder Needs:**
   - **Description:** Requirement gathering ensures that the project team understands and aligns with the needs and expectations of stakeholders, including users, clients, and regulatory bodies.
   - **Impact:** This alignment helps in delivering a product that meets stakeholder expectations, leading to higher satisfaction and acceptance.
2. **Reduces Project Risks:**
   - **Description:** By identifying and understanding requirements early in the project, potential risks and challenges can be addressed proactively.
   - **Impact:** This reduces the likelihood of project delays, cost overruns, and failure, ensuring a smoother development process.

3. **Facilitates Accurate Planning and Estimation:**
    - **Description:** Detailed and well-understood requirements provide a solid foundation for project planning, including time, cost, and resource estimation.
    - **Impact:** This helps in creating realistic project plans and budgets, reducing the risk of scope creep and ensuring efficient resource utilization.
4. **Improves Communication and Collaboration:**
    - **Description:** The requirement gathering process involves continuous interaction with stakeholders, fostering clear communication and collaboration.
    - **Impact:** This shared understanding helps in preventing misunderstandings and misalignments, ensuring that all stakeholders are on the same page.
5. **Enhances Product Quality:**
    - **Description:** Clear and detailed requirements provide a blueprint for designing, developing, and testing the system.
    - **Impact:** This helps in creating a high-quality product that performs reliably and meets specified performance criteria, leading to higher user satisfaction and fewer defects.
6. **Supports Change Management:**
    - **Description:** A structured requirement gathering process includes mechanisms for managing changes to requirements.
    - **Impact:** This ensures that changes are evaluated, approved, and documented, maintaining project scope and preventing uncontrolled modifications.
7. **Facilitates Requirement Traceability:**
    - **Description:** Requirement gathering includes documenting and linking requirements to their sources and related project artifacts.
    - **Impact:** This traceability helps in ensuring that all requirements are addressed throughout the project lifecycle, supporting validation and verification activities.