1. What are requirements? Bring out the different ways of documenting software project requirements.

Ans:- **requirements** are descriptions of the system's functionalities and constraints. They define what the software system should do, how it should behave, and the criteria it must meet. Requirements act as a bridge between the clients' needs and the final product developed by the software team.

Different Ways of Documenting Software Project Requirements:

1. Functional Requirements:

- **Definition:** Functional requirements describe what the system should do. They specify the system's behavior under certain conditions.
- **Example:** "Users must be able to add products to the shopping cart and proceed to checkout."

2. Non-Functional Requirements:

- **Definition:** Non-functional requirements specify the quality attributes of the system. They describe how the system performs a specific function rather than the function itself.
- **Example:**: "The website should load within 3 seconds to provide a responsive user experience."

3. Business Requirements:

- **Definition:** Business requirements describe the high-level needs and goals of the organization initiating the project. They provide context for the project and help align it with business objectives.
- **Example:** "Increase online sales by 20% within the next fiscal year."

4. User Requirements:

- **Definition:** User requirements specify the needs of end-users. They describe the interactions users have with the system.
- **Example:** "Users should be able to reset their password through a 'Forgot Password' link on the login page."

5. System Requirements:

- **Definition:** System requirements define the technical specifications and constraints for the software system. They detail the hardware, software, and network configurations necessary for the system to function.
- **Example:** "The system should be compatible with Windows 10, macOS 11, and the latest versions of popular web browsers."

2. Differentiate b/w functional & non-functional requirements.

FUNCTIONAL REQUIREMENTS	NON-FUNCTIONAL REQUIREMENTS
THEY DEFINE HOW THE SYSTEM SHOULD WORK	THEY DEFINE THE QUALITY ATTRIBUTES OF THE SYSTEM
THEY ARE DESCRIBED IN THE SOFTWARE REQUIREMENT SPECIFICATION	THEY ARE DESCRIBED IN THE SOFTWARE ARCHITECTURE DOCUMENT
THEY EXPLAIN THE BEHAVIOR OF A FUNCTION OR FEATURE	THEY EXPLAIN THE OPERATING BEHAVIOR OF AN ENTIRE SYSTEM OR SYSTEM COMPONENT, NOT A SPECIFIC FUNCTION
THEY ARE MANDATORY FOR FULFILLMENT	THEY ARE NOT MANDATORY FOR FULFILLMENT
THEY DERIVE FROM THE SOFTWARE REQUIREMENTS	THEY DERIVE FROM THE FUNCTIONAL REQUIREMENTS
THEY FORM THE SKELETON OF SOFTWARE SYSTEM IMPLEMENTATION	THEY COMPLEMENT THE SOFTWARE SYSTEM
THEY CAN EXIST WITHOUT A NON-FUNCTIONAL REQUIREMENT	THEY CAN'T EXIST WITHOUT FUNCTIONAL REQUIREMENTS
THEY PROVIDE SPECIFIC INFORMATION ABOUT THE FEATURE	THEY CAN HAVE MANY ATTRIBUTES OF NON-FUNCTIONAL REQUIREMENTS
THEY ARE DEFINED AT A COMPONENT LEVEL	THEY ARE APPLIED TO A WHOLE SYSTEM
THEY ARE USUALLY EASY TO DEFINE	THEY ARE NOT SO EASY TO DEFINE

3. Explain User and System requirements. Write sample User and system requirements for Mentcare System.

Ans :- User Requirements:

Definition: User requirements describe the functionalities and features of the software system from an end-user perspective. They focus on what the users need the system to do for them.

Sample User Requirements for Mentcare System:

1. **User Authentication:**
2. **Patient Profile Management:**
3. **Appointment Scheduling:**
4. **Secure Messaging:**
5. **Mood Tracker:**
6. **Prescription Management:**
7. **Notifications:**

System Requirements:**Definition:** System requirements detail the technical aspects of the software system, including hardware, software, and network configurations necessary for the system to function efficiently.

Sample System Requirements for Mentcare System:

1. **Platform Compatibility:**
2. **Database:**
3. **Security:**
4. **Server Requirements:**
5. **Notification Service:**

4. Explain how to validate non-functional requirements ?

Ans:- **Validating non-functional requirements** can be a complex task, as they often deal with aspects of the system that are not directly observable or quantifiable through typical functional testing. Non-functional requirements include aspects such as performance, reliability, security, and usability.

1. Performance Requirements:

- **Load Testing:** Simulate expected user loads and observe system behavior. Measure response times, resource utilization, and throughput under various loads.

2. Reliability Requirements:

- **Failure Mode Testing:** Intentionally introduce failures (such as server crashes or network interruptions) to observe how the system recovers and maintains functionality.

3. Security Requirements:

- **Penetration Testing:** Ethical hackers attempt to exploit system vulnerabilities to uncover potential security risks. This helps identify weak points in the system's security measures.

4. Usability Requirements:

- **User Testing:** Conduct usability testing with representative users. Observe user interactions and gather feedback on the system's interface and overall user experience.

5. What are the activities involved in Requirements Engineering Process.

Ans:- 1. Elicitation:

- **Definition:** Elicitation involves gathering requirements from stakeholders. It includes interviews, surveys, workshops, brainstorming sessions, and observations to collect information about the system's functionalities and constraints.

2. Analysis:

- **Definition:** Analysis involves understanding the gathered requirements in depth. It includes identifying inconsistencies, ambiguities, and incompleteness in the requirements.

3. Specification:

- **Definition:** Specification involves documenting the requirements in a clear, concise, and unambiguous manner. It includes creating requirement documents that serve as a reference for the development team.

4. Validation:

- **Definition:** Validation ensures that the documented requirements accurately represent stakeholders' needs and expectations. It involves verifying the requirements for correctness and completeness.

5. Verification:

- **Definition:** Verification involves ensuring that the specified requirements can be tested and validated. It includes defining acceptance criteria and metrics for measuring the requirements' success.

6. Communication:

- **Definition:** Effective communication ensures that all stakeholders have a clear understanding of the requirements. It involves facilitating discussions, resolving conflicts, and keeping all stakeholders informed.
- **Key Tasks:** Organizing regular meetings, providing status updates, and facilitating communication channels between stakeholders.

6. What are the ways of specifying system requirements specification? Explain any 2 types with example.

    1. Ans:- Use Cases: A use case is a description of a system's behavior as it responds to a request from one of its users. Use cases can be used to represent the functional requirements of a system.
    2. User Stories: A user story is a short, simple description of a feature written from the perspective of the user. User stories can be used to represent both functional and non-functional requirements of a system.
    3. Requirements Specification: A requirements specification is a detailed document that describes the requirements for a system. It can include functional and non-functional requirements, as well as constraints, assumptions, and dependencies.
    4. Requirements Traceability Matrix: A requirements traceability matrix is a table that shows the relationships between different requirements in a system. It can be used to track the progress of the development of a system and ensure that all requirements have been addressed.
    5. Prototypes: Prototypes are simplified versions of a system that are used to test and refine the requirements. Prototypes can be used to represent both functional and non-functional requirements.

7. What are software artifacts? Explain.

Ans:- **Software artifacts** are the tangible products and documents created during the software development process. These artifacts serve various purposes, such as communication, planning, design, and documentation. They provide a structured way to capture, organize, and represent different aspects of a software project. Here are some common software artifacts and their explanations:

1. Requirements Document:

- **Explanation:** A document outlining the functional and non-functional requirements of the software system. It describes what the system should do and how it should behave.
- **Purpose:** Provides a clear understanding of user needs and expectations, guiding the development process.

2. Use Cases and User Stories:

- **Explanation:** Use cases and user stories describe specific interactions between users and the system. They detail how users interact with the system to achieve specific goals.
- **Purpose:** Captures functional requirements from an end-user perspective, guiding the development of features and functionalities.

3. Design Documents:

- **Explanation:** Documents detailing the system architecture, components, data structures, algorithms, and user interfaces. Design documents provide a blueprint for the development team.
- **Purpose:** Guides developers in implementing the system, ensuring consistency and adherence to the planned structure.

4. Codebase:

- **Explanation:** The actual source code files written by developers. The codebase includes scripts, classes, functions, and modules that make up the software system.
- **Purpose:** The foundation of the software system, executing the defined functionalities and algorithms.

5. Test Cases:

- **Explanation:** Test cases specify scenarios and conditions to validate the software. They outline steps, inputs, and expected outputs for different aspects of the system.
- **Purpose:** Guides the testing team in verifying that the software behaves as expected and meets the requirements.

6. Database Schema:

- **Explanation:** Describes the structure of the database, including tables, fields, relationships, and constraints. It defines how data is organized and stored.
- **Purpose:** Guides database administrators and developers in creating and managing the database that the software uses.