MIS – Unit 4

Topic: Detailed System Design

Ques: What is Detailed System Design ?

Ans: Detailed system design is where you create specific blueprints for how the system will look, what it will do, and how it will do it. It's like drawing a detailed map for a treasure hunt, so you know exactly where to dig to find the treasure.

During this phase, you're answering questions like "How do we build this system?", "What does each part of the system do?", and "How will people use it?" It's like making sure every puzzle piece fits perfectly, so you end up with a complete and functional picture. Once this phase is done, you're ready to build and use your MIS effectively. **Here's the main stages we go:**

1. Create System Specifications: This is like making a detailed plan. You write down exactly what the MIS should do, what it should look like, and how it should work.

2. Design Data Storage: Think of this as figuring out where to store all the information the MIS will use. It's like deciding where to put your books, but in a very organized way so you can find any book quickly.

3. Plan Data Processing: This step is about deciding how the MIS will work with data. It's like having a recipe for how to cook a meal. You list all the steps and ingredients to make sure it turns out right.

4. Create User Interfaces: Imagine the MIS as a car. User interfaces are like the dashboard and controls inside the car. You design how people will interact with the system, like steering the car or pressing the pedals.

5. Develop Software: This is where you build the actual system. It's like constructing a house after you have the architectural plans. You write the code that makes the MIS work.

6. Plan System Testing: Before you drive a car, you want to make sure it's safe and works correctly. The same goes for the MIS. You plan how to test it to catch any issues before it's in full use.

7. Conduct User Training: Just like you'd learn how to drive a car, people using the MIS need to know how it works. Training ensures they can use the system effectively.

8. Document the System: You create documents to explain how the MIS works. Think of it like having an instruction manual for the car. These documents help users and future developers understand the system.

9. Prepare the Detailed Design Report: You gather all the information and decisions made during this phase into a report. It's like having a final document that sums up how the MIS is designed and built.

Explain "Aim of Detailed Design" ?

The aim of Detailed Design is to make a detailed plan for how to build a system after having the initial big-picture idea. It's like creating a step-by-step instruction manual for building something, where each part is carefully detailed and explained. This phase makes sure that all the small parts of the system are well-thought-out, planned, and documented. The goal is to make sure that everyone involved in building the system understands what needs to be done, how to do it, and what challenges might come up so they can be prepared. Ultimately, it's about turning the big idea into a practical and detailed plan for actual development.

Here's more description:

- Breaking down the overall system concept into smaller, more specific components.
- Creating comprehensive blueprints, diagrams, and documents detailing how each part of the system will be constructed, integrated, and tested.
- Ensuring that the planned system can be built using available resources and technology.
- Identifying potential risks that might arise during system development and creating strategies to overcome these challenges.

- Providing clear and understandable documentation to all involved parties, ensuring that developers, testers, and managers have a thorough understanding of the system's design and implementation plan.

Explain “Detailed Sub System” ?

A "Detailed Subsystem" refers to a specific part or component within a larger system that is examined and designed in-depth during the detailed design phase.

In a complex system, breaking it down into smaller, more manageable parts or subsystems is common. A Detailed Subsystem is one of these smaller parts. It's a more detailed and focused examination of a particular functional area or component within the larger system.

For instance, imagine a company's overall computer network system. Within this network, there might be subsystems like the email server, file storage system, or security protocols. Each of these smaller systems, such as the email server, can be further broken down into detailed subsystems - like email authentication, storage management, and delivery protocols.

Define Input, Output and Process Design in Detailed Design Process with help of examples.

1. **Input Design:** This aspect involves planning how data will be collected or entered into the MIS. It focuses on designing methods and interfaces for efficient, accurate, and secure data input.

   This includes:

- **Data Collection Methods:** Planning methods for gathering data, such as manual entry, automated sensors, or direct database integration.
- **Forms and Interfaces:** Creating user-friendly forms or interfaces for data entry, ensuring accuracy and ease of use.

- **Data Validation:** Implementing checks to verify the accuracy and reliability of entered data.
- **Security Measures:** Incorporating measures to ensure data security and prevent unauthorized access.

  Example: In a healthcare MIS, the input design might include user-friendly interfaces for medical staff to enter patient information. It could involve electronic forms or systems that collect patient demographics, medical history, and current symptoms.

2. **Output Design:** This encompasses planning how information will be presented or delivered to users or stakeholders. It involves designing reports, summaries, or presentations that convey processed data in a useful and understandable format.

   This includes:

- **Report Generation:** Designing reports, summaries, or dashboards to display processed data.
- **Formatting and Visualization:** Choosing suitable formats and visualizations for data representation (graphs, charts, tables, etc.) for better understanding.
- **Customization:** Allowing users to customize outputs based on their preferences or specific needs.
- **Distribution Methods:** Determining how and to whom the information will be distributed (via email, printed reports, online portals, etc.).

  Example: In an educational institution's MIS, the output design might involve generating performance reports for students. These reports could include grades, attendance records, and additional insights into individual student progress, presented in an easily interpretable format for teachers and parents.

3. **Process Design:** This involves planning how data will be processed, manipulated, or transformed within the MIS. It includes designing

algorithms, workflows, or procedures to ensure efficient data processing and management.

This includes:

- **Workflow Design:** Structuring workflows or sequences of operations for data processing.
- **Algorithm Development:** Creating algorithms or rules for data processing, such as calculations or decision-making logic.
- **Error Handling:** Implementing procedures to handle errors or exceptions during data processing.
- **Integration:** Ensuring seamless integration with other systems or databases for smooth data flow.

Example: In a healthcare MIS, process design may involve developing algorithms for patient diagnosis based on symptoms input, utilizing predefined medical knowledge and treatment protocols.

Explain "Hardware and Software Selection" ?

"Hardware and Software Selection" involves choosing the appropriate technology components necessary for the implementation and operation of the system. This phase is crucial as it determines the infrastructure and tools needed to support the system's functionality.

1. **Hardware Selection:** This involves choosing the physical components or devices required to support the system's operation. It includes:
    - **Computing Devices:** Selecting servers, workstations, networking equipment, storage devices, etc., based on the system's processing and storage needs.
    - **Peripheral Devices:** Choosing printers, scanners, sensors, or other devices necessary for data input/output or system integration.

2. **Software Selection:** This involves choosing the necessary programs, applications, or software components required for the system's functioning. It includes:
    - **Operating Systems:** Selecting the appropriate OS that supports the system's requirements and is compatible with the hardware.
    - **Application Software:** Choosing specific software applications or programs that align with the system's objectives. This could include database management systems, programming languages, analytical tools, etc.
    - **Integration Tools:** Identifying tools that facilitate seamless integration between different software components or interfaces.

How the software hardware selection is done in detailed design process ?

1. **Understanding System Requirements:** First, the design team thoroughly understands what the system needs to do. They analyze what functions it should perform, how many users it will have, and how much data it will handle.
2. **Identifying Technology Needs:** Based on the system requirements, the team figures out what kind of hardware (like computers, servers, etc.) and software (applications, operating systems, etc.) will be needed to make the system work effectively.
3. **Research and Evaluation:** The team then researches different hardware and software options available in the market. They compare various options, looking at factors like features, compatibility, reliability, performance, and cost.
4. **Testing and Prototyping:** Sometimes, they may create prototypes or conduct tests with different combinations of hardware and software to see how well they work together and if they meet the system requirements.
5. **Selection Process:** After thorough evaluation and testing, the team selects the hardware and software that best fit the system's needs and budget.

They make decisions based on what will work well together, be reliable, and perform efficiently.

6. **Documentation and Planning:** Finally, the chosen hardware and software are documented, and plans are made for their acquisition, installation, and integration into the system.

Explain show how Detailed Design is helpful in system design ?

1. **Making the Big Idea Practical:** Detailed Design takes the big idea for a system and breaks it down into smaller, more manageable parts. It plans out every detail, like how data will be entered, how it will be processed, and how the results will be shown. This makes the big idea practical and doable.
2. **Avoiding Surprises:** It helps avoid surprises during the actual building phase. By planning everything beforehand, it helps the designers and developers anticipate potential issues or challenges and create solutions for them in advance.
3. **Clear Instructions:** It's like having a clear set of instructions. Just like a recipe tells you what ingredients to use and how to cook a dish, Detailed Design tells the designers and developers exactly what to do and how to do it.
4. **Better Communication:** It improves communication among the team members. Everyone involved in building the system understands what needs to be done because Detailed Design provides clear and detailed information.
5. **Efficiency and Accuracy:** It promotes efficiency and accuracy in building the system. Since everything is planned in detail, the chances of mistakes are lower, and the system can be built faster and more accurately.

How Conceptual Design becomes inputs to detailed design phase ? What are its phases ?

Think of Conceptual Design as drawing the blueprint or sketch of a house you want to build, and Detailed Design as planning every room, door, window, and detail before actually building it.

1. **Translating Big Ideas to Details:** Conceptual Design outlines the big ideas and overall structure of the system. It identifies the system's purpose, main functions, and how it might look in broad strokes, like a rough sketch.
2. **Detailed Design as the Blueprint:** Detailed Design takes these big ideas from the Conceptual Design and creates a detailed blueprint or plan for building the system. It's like taking that rough sketch and deciding precisely where every wall, window, and door goes in every room.
3. **Building on Conceptual Foundations:** Detailed Design uses the concepts and ideas from the Conceptual Design as a foundation. It takes those big ideas and figures out the nitty-gritty details, like how data will flow, what the screens will look like, and how users will interact with the system.
4. **Adding Specifics and Technical Details:** Conceptual Design is about the 'what' and 'why,' while Detailed Design is about the 'how.' It converts the conceptual ideas into specific technical plans that developers and designers can use to actually build the system.

   (At last write phases/process of Detailed System Design)

What is Design Documentation ? Explain, Write the important contents present in detailed design documentation ?

Design Documentation is like a detailed instruction manual for building something, explaining how a system should be created, organized, and function. It includes all the essential information needed for developers and designers to understand and construct the system correctly.

Important Contents in Detailed Design Documentation:

1. **System Overview:** A brief description of the system, its purpose, and the problem it solves.
2. **Detailed System Architecture:** Detailed diagrams showing how different parts of the system interact, like flowcharts or diagrams displaying data flow.
3. **Data Design:** Details about how data will be stored, structured, and managed within the system, including database schema, data models, etc.
4. **Input and Output Specifications:** How data will be collected or entered into the system (Input) and how information will be presented or provided to users (Output).
5. **Process Design:** Explanation of how data will be processed or transformed within the system, including algorithms or workflow descriptions.
6. **Hardware and Software Requirements:** Specific details about the necessary hardware devices and software components required for the system.
7. **Interface Design:** Descriptions or mock-ups of user interfaces, showing how users will interact with the system.
8. **Security Measures:** Details about the security features and measures implemented to protect data and the system from unauthorized access.
9. **Error Handling and Recovery:** Procedures and strategies for dealing with errors or system failures to ensure smooth operation.
10. **Testing Plans:** Strategies and methods for testing the system to ensure it works as intended.

What is Internal Output" ?

Internal Output refers to the information produced by a system that is intended for use within the organization or by its employees, rather than being directed to external parties like customers or suppliers.

1. **For Internal Use:** Internal Output is generated by a system and is meant to be used by people within the organization. It includes reports, summaries, or data that are useful for employees, managers, or departments within the company.
2. **Helps Decision-Making:** This type of output often includes detailed reports, analysis, or performance metrics that assist employees or management in making decisions related to operations, strategies, or resource allocation.
3. **Examples:** Internal Output might include sales reports for the sales team, financial performance reports for managers.

Differentiate between "Verification" and "Validation" ?

Verification:

- **Meaning:** Verification confirms whether the product or system meets its initial requirements and specifications. It ensures that the system is built correctly according to the predefined plans.
- **Focus:** Verification focuses on ensuring that the development team is developing the product right by adhering to the specified requirements.
- **Methods:** It involves reviews, inspections, and walkthroughs to check documents, plans, and code against the specified requirements.
- **Objective:** The primary goal of verification is to make sure that the system is being developed in line with what was planned initially.

Validation:

- **Meaning:** Validation confirms whether the final product or system meets the actual needs and expectations of the users or customers. It ensures that the end product works effectively in the real world.
- **Focus:** Validation concentrates on checking if the system satisfies the user's real requirements and whether it fits into their environment.
- **Methods:** It involves actual testing and assessment to ensure that the system functions correctly and meets user expectations.

- **Objective:** The main aim of validation is to make sure that the right product is being built, one that fulfills the user's actual needs and expectations.

In essence, verification checks if the system is being built correctly, following the specified plans, while validation checks if the correct system is being built that satisfies the user's actual needs. Both processes are important in ensuring the quality and success of the system.

Difference between "Utility Software" and "Application Software" ?

Utility Software:

- **Purpose:** Utility software focuses on managing and improving the performance of the computer system.
- **Functionality:** It provides tools to handle various system-related tasks like virus protection, disk cleanup, system optimization, and data backup.
- **Examples:** Antivirus software that protects against viruses, disk cleanup tools that remove unnecessary files, and backup software for data protection.
- **Usage:** Utility software runs in the background, ensuring the computer works smoothly without interruptions.

Application Software:

- **Purpose:** Application software is designed to perform specific tasks catering to the needs of users.
- **Functionality:** It includes software for word processing, spreadsheet calculations, gaming, graphic design, and browsing the internet.
- **Examples:** Word processors (like Microsoft Word), spreadsheet software (like Excel), web browsers (like Chrome), video editors, and games.
- **Usage:** Users directly interact with application software to accomplish tasks like creating documents, editing photos, or playing games.

Topic: System Testing Below:-

Unit - 4

Date: / /
Page:

XTRA EDGE

Testing :- In software development, testing is a crucial step in ensuring that the system or application being developed meets the requirements and specifications defined in the design phase. Testing can be defined as a process of evaluating the system's performance or behaviour against the expectations and requirements of the user. It involves executing the system with the intent of finding errors, bugs or faults, and verifying that the system meets the intended functionality.

Testing is usually conducted in a staged manner, starting with unit testing, followed by integration testing, system testing, and acceptance testing. The testing process is usually conducted by a team of testers who are responsible for identifying and reporting any issues or bugs in the system.

Why do we Test System? How is important testing?

Testing is a crucial phase in the System Development Life Cycle (SDLC). It is the process of identifying defects, errors in the system and ensuring that it meets the specified requirements. The primary goal of testing is to validate that the system is working as per the client's requirements and meets the business objectives.

Testing is essential for several reasons, some of which are:

1. Identify and correct defects:- The primary goal of testing is to identify defects or bugs in the system and correct them before the system goes live. By detecting issues early, it is possible to correct them with minimum cost and effort.

2. Ensure system reliability:- Testing ensures that the system is reliable and stable. It helps to identify errors that could cause the system to crash or fail in the production environment.

3. Validate system functionality:- Testing ensures that the system meets the functional requirements specified by the client. It validates that the system is performing as expected and can handle different scenarios.

4. Enhance system security:- Testing can also help to identify security problems in the system. By identifying and correcting these problems, the system's security can be enhanced, and the risk of cyber-attacks can be reduced

5 Ensure user satisfaction:- The ultimate goal of testing is to ensure that the system meets the user's expectations. By validating that the system is functioning correctly, it helps to ensure that users can use the system efficiently and effectively.

6 Quality Assurance:- Testing helps to ensure that the system is of high quality, meets user needs, and is reliable and efficient.

7 Cost - Effective:- Testing helps to save costs by detecting errors or defects early in the development process, reducing the likelihood of costly rework or system failure after deployment.

8 Improve System performance:- Testing helps to identify performance issues, bottlenecks, and other inefficiences, allowing developers to optimize the system and improve its performance.

9 Enhance system usability:- Testing can also help to improve the usability of the system by identifying areas that are difficult or confusing for users.

10 Reduce maintenance costs:- Testing helps to identify and fix problems early in the development process, reducing the likelihood of expensive

maintenance and rework later on.

Levels of Testing :-

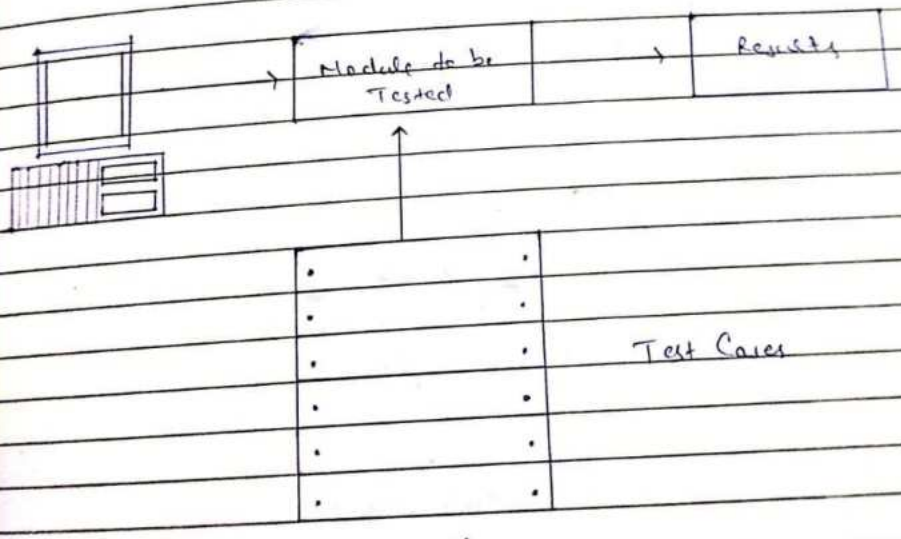
1. **Scaffolding** :- It refers to the creation of temporary structures or tools that help support the testing process. Scaffolding can be used to simulate various system conditions and test scenarios, and is typically used in the early stages of system testing. Scaffolding is an important part of system testing because it helps identifying potential issues early in the testing process, before they become more difficult and expensive to fix. Additionally, it can help reduce the time and effort required to test the system, which can be particularly important for large and complex systems.

2. **Unit Testing** :- Unit Testing is a level of system testing that involves the testing of individual units or modules of code. In this testing, each module is tested in isolation from other modules, and the objective is to ensure that each module functions correctly according to its design.

Unit testing is typically carried out by developers during the coding phase of SDLC. It involves the creation of test cases that cover all

possible scenarios and conditions that the module may encounter during its execution.

Unit testing provides several benefits to the system development process. Firstly, it helps to identify and eliminate defects early in the development process, which reduces the overall cost of development. Secondly, it helps to improve the quality of the code by ensuring that each module works correctly in isolation.

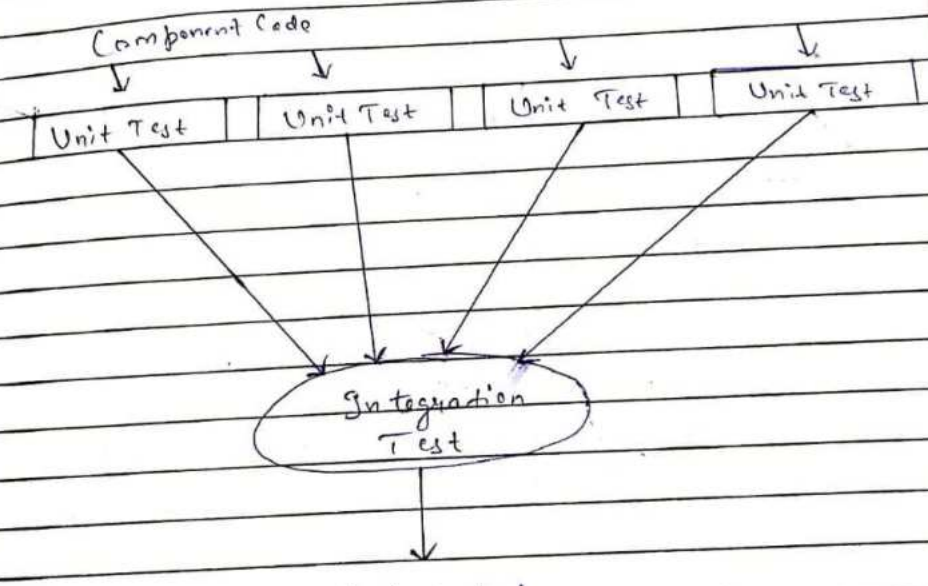


Diag: Unit Testing

3. Integration Testing: It is a level of system testing that is performed after unit testing. In this testing, individual modules that have been unit tested are combined and

Date : ../../..
Page: .....
XTRA EDGE

tested as a group. The main purpose of integration testing is to identify defects in the interactions b/w different modules when they are integrated. During integration testing, test cases are designed to test the interactions b/w different modules. Integration testing helps to ensure the system functions as a whole, and it helps to reduce the risk of defects in the final product.



Diagr Integration of individual Modules

Integration testing can be further classified as:

3.1 Top-down Integration Testing: In this, the higher level modules are tested first and the lower

level modules are gradually integrated and tested. This approach is useful when the higher level modules are more critical to the system.

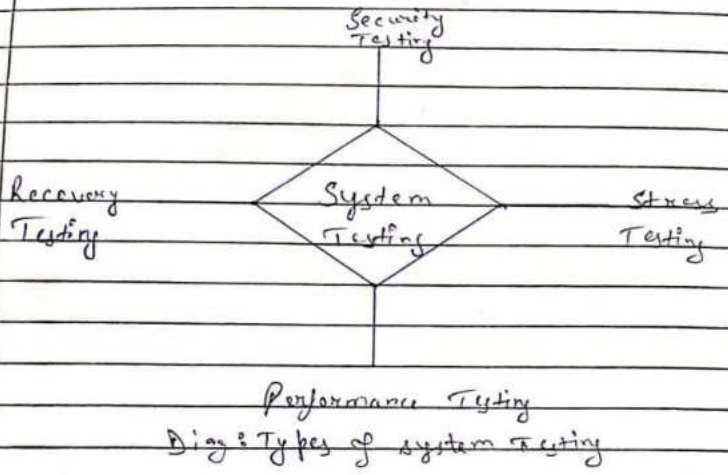
3.2 Bottom-up Integration Testing :- In this, the lower level modules are tested first and the higher level modules are gradually integrated and tested. This approach is useful when the lower level modules are more critical to the system.

3.3 ~~Reg~~ Regression Testing :- It is a type of testing that ensures that changes made to the system, such as bug fixes or new feature implementations, do not introduce new errors into the system or cause previous errors to reappear.

3.4 Smoke Testing :- It is a type of testing that is performed to check if the system is stable and functional enough to proceed with further testing. Its purpose is to identify major ~~errors~~ defects early in the testing process.

#4 System Testing :- It is a level of software testing where the entire system is tested as a whole to ensure that all its components are working together as expected and meeting the specified requirements. The primary objective of system testing is to verify that the system meets the functional,

performance and security requirements, and it is capable of operating under normal and abnormal conditions.



Diag: Types of system Testing

4.1 Recovery Testing:- It is a type of system testing that evaluates the system's ability to recover from an unexpected failure, such as hardware or software failure. The primary objective of this type of testing is to ensure that system can return to its normal state after a failure.

4.2 Security Testing:- It is a type of system testing that aims to identify vulnerabilities in the system that can be exploited by unauthorized users or hackers. This testing involves testing the system against different types of attacks.

4.3 Stress Testing :- Stress testing is a type of testing that evaluates the system's behaviour under extreme conditions, such as heavy load, high traffic or limited resources. Its primary objective is to identify the system's breaking point and ensure that the system can handle a sudden surge in traffic or demand.

4.4 Performance Testing :- It is a type of testing that evaluates the system's response time, and other performance related metrics under various load conditions. Its primary objective is to ensure that the system can handle the expected level of load and provide a smooth user experience.

5 Acceptance Testing :- It is the process of verifying that a system meets its requirements and is ready for ~~devel~~ deployment to the end-users or customers. It is performed to ensure that the system meets the specified business requirements and user needs. Acceptance testing can be performed manually or automatically, and it is typically the responsibility of the end-users or customers to perform this type of testing.

5.1 Alpha Testing :- It is the first phase of testing which is carried out internally in the development team. It is done before the software released to the customer. It is done to identify and fix bugs or defects in the system.

Differentiate between Unit and Integration Testing ?

1. **Scope:**
    - **Unit Testing:** It checks small individual parts (like functions or modules) of the software in isolation, making sure each piece works correctly on its own.
    - **Integration Testing:** It examines how these individual parts come together and function as a whole system, ensuring they work together smoothly.
2. **Testing Level:**
    - **Unit Testing:** This is like testing each brick individually before building a house, verifying that every small part performs its job as expected.
    - **Integration Testing:** Once the bricks are tested, integration testing checks how these bricks fit together, ensuring the entire structure is stable.
3. **Dependencies:**
    - **Unit Testing:** It tries to test each part separately, sometimes using fake or simulated elements to avoid relying on external parts.
    - **Integration Testing:** It looks at how real parts interact and work together, considering how these pieces depend on one another.
4. **Isolation:**
    - **Unit Testing:** Focuses on one piece at a time, ignoring the rest to make sure that individual piece functions properly.
    - **Integration Testing:** Checks how these pieces combine, ensuring they work well when connected together in the larger system.
5. **Granularity:**
    - **Unit Testing:** Zooms in on the smallest details, examining tiny parts like specific lines of code or functions.

- **Integration Testing:** Looks at the bigger picture, testing how these parts work together in broader scenarios.

6. **Timing in Software Development Life Cycle (SDLC):**
    - **Unit Testing:** Happens early on during development when coding, ensuring each small part is working before assembling them.
    - **Integration Testing:** Takes place a bit later, ensuring the integrated units function properly together as a cohesive system.

Differentiate testing and debugging with the help of examples.

1. **Purpose:**
    - **Testing:** Checks if the software works as expected by running different scenarios. *Example:* Testing an email feature to ensure sending and receiving emails works properly.
    - **Debugging:** Identifies and fixes specific issues found during testing. *Example:* Fixing a bug where the email system crashes when a user sends an attachment.
2. **Focus:**
    - **Testing:** Ensures the software meets requirements and functions correctly. *Example:* Verifying if a calculator app correctly performs addition, subtraction, etc.
    - **Debugging:** Concentrates on solving a particular problem or malfunction in the software. *Example:* Investigating and rectifying why the calculator app shows incorrect results for multiplication.
3. **Phase in Development:**
    - **Testing:** Happens continuously during development to find problems early. *Example:* Checking code as it's written to avoid errors.
    - **Debugging:** Occurs after testing when issues are found and need to be fixed. *Example:* Addressing problems reported by users or testers.

4. **Process:**
    - **Testing:** Involves running tests and comparing expected versus actual results. *Example:* Confirming if a game character moves correctly in response to user commands.
    - **Debugging:** Involves finding and solving specific issues. *Example:* Finding why the game character suddenly freezes in one level.
5. **Goal:**
    - **Testing:** Ensures the software works correctly to meet user needs. *Example:* Checking if a social media app allows users to post comments without errors.
    - **Debugging:** Aims to fix identified problems to make the software error-free. *Example:* Resolving an issue where comments on the app disappear after posting.

The End

Mitrooo…..