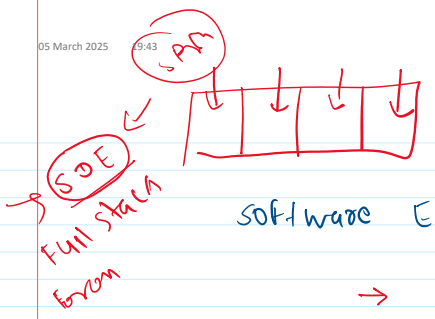
05 March 2025 19:43



Introduction to Software Engineering

AlmaBetter

Software Engineering disciplined approach

- → Designing
- → developing
- → testing
- → deploying

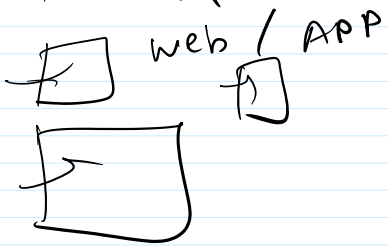
} maintaining software.

- Analysis : User needs and business goals.
- Design : Plan the software architecture and user interface
- Coding → implement the design with proper prog.
- Testing :- Ensure the software works.

Deployment & maintenance : update.

Amzon : mange millions of trascations

reliable
scalable
platform



2. Software Engineering Vs programming

Software Engineering :

- focuses the entire lifecyle : planning, designing coding, testing, maintenance.

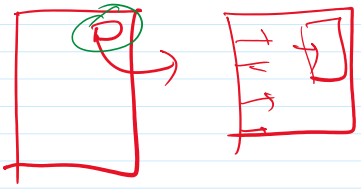
programming : writing code. slove problems.

3. (SDLC) software Development Life

Software Development Life Cycle (SDLC)

- **Phases of SDLC:**
  - **1. Analysis:** Gather and understand requirements.
  - **2. Design:** Create detailed system blueprints (architecture, UI, etc.).
  - **3. Development:** Write the actual code.

LMS



1. **Analysis:** Gather and understand requirements.
2. **Design:** Create detailed system blueprints (architecture, UI, etc.).
3. **Development:** Write the actual code.
4. **Testing:** Identify and fix defects.
5. **Deployment:** Release the product to users.
6. **Maintenance:** Update and improve the system over time.

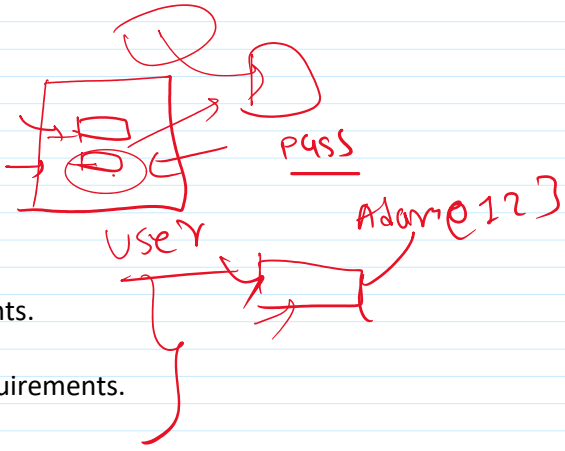
- **Real-World Implication:**
  Amazon follows a robust SDLC to continuously roll out updates without disrupting user experience, ensuring high performance and reliability even under heavy traffic.

Software Requirements Engineering

- **Purpose:**
  To understand and document what the stakeholders need from the software.
- **Types of Requirements:**
  - **Functional Requirements:** What the software must do (e.g., search, purchase, recommendations).
  - **Non-Functional Requirements:** How the software performs (e.g., speed, security, usability).
  - **User & System Requirements:** Direct needs from end-users and technical/environmental needs.
- **Techniques for Eliciting Requirements:**
  - Interviews, surveys, observation, and document analysis.
  - **Example:** Amazon uses user behavior data to refine its recommendation engine.

5. Software Design Principles and Architectural Patterns

- **Key Design Principles:**
  - **Modularity:** Breaking the system into independent modules.
  - **Encapsulation:** Bundling data and functions together.
  - **Abstraction:** Hiding complex details to simplify the system.
- **Architectural Patterns:**
  - **Client-Server:** Separates user interface from back-end processing.
  - **Model-View-Controller (MVC):** Divides the system into three parts for easier management.
  - **Layered Architecture:** Organizes the system into layers with specific responsibilities.
- **Design Notations:**
  - **UML Diagrams:** Standardized diagrams (e.g., class, sequence diagrams) for visualizing system design.
  - **Flowcharts:** Represent process flows and decision points.
- **Real-World Example:**
  Amazon's architecture uses a mix of patterns to ensure that their website remains responsive, secure, and easy to update as new features are added.



Software Testing and Test-Driven Development (TDD)

- **Importance of Testing:**
  - Ensures quality and reliability by catching defects early.
  - Reduces risks and saves cost by avoiding rework.
- **Types of Testing:**
  - **Unit Testing:** Tests individual pieces of code.
  - **Integration Testing:** Checks interaction between components.
  - **System Testing:** Validates the complete system.
  - **Acceptance Testing:** Confirms the software meets user requirements.
- **Test-Driven Development (TDD):**

1. Write a test for a specific function.
2. Run the test (it should fail initially).
3. Write the minimal code needed to pass the test.
4. Run all tests to ensure nothing breaks.
5. Refactor code and repeat.

- **Interactive Activity (True/False):**
  - *"Software engineering involves applying engineering principles to create high-quality software."*

(True)

- *"Non-functional requirements focus on the specific functions and capabilities of the system."* (False – they focus on performance, security, etc.)

Software Maintenance and Evolution

- **Key Concepts:**
  - **Bug Fixing:** Correcting errors to ensure smooth functioning.
  - **Enhancements:** Updating the software to add new features or improve performance.
  - **Version Control:** Managing changes through systems like Git for collaboration and tracking revisions.
- **Real-World Implication:**
  Amazon continuously updates its platform to enhance security, add features, and improve performance, all while maintaining backward compatibility and user trust.

Software Project Management

- **Project Planning and Estimation:**
  - Define project goals, scope, deliverables, and timelines.
  - Estimate the required resources using historical data and expert judgment.
- **Roles and Responsibilities:**
  - **Project Manager:** Oversees the project, manages timelines and resources.
  - **Developers, Analysts, QA Engineers, UI/UX Designers:** Each contributes to different aspects of the project.
  - **DevOps, Database Administrators, Technical Writers:** Support development, maintenance, and documentation.
- **Project Tracking Tools:**
  - **Gantt Charts:** Visualize task timelines and dependencies.
  - **Agile Boards (Scrum/Kanban):** Track work progress in iterative cycles.

Software Quality Assurance (QA)

- **Purpose of QA:**
  To ensure that the final product meets the required quality standards through continuous testing and process improvements.
- **Quality Attributes:**
  - **Reliability, Usability, Efficiency, Security, Maintainability, Scalability.**
- **Techniques for Quality Assurance:**
  - Test planning, test case design, regression testing, automated testing, code reviews.
- **Real-World Example:**
  High standards in QA help Amazon maintain its reputation by ensuring that their platform is reliable, secure, and user-friendly, even under high load.

Ethics and Professionalism in Software Engineering

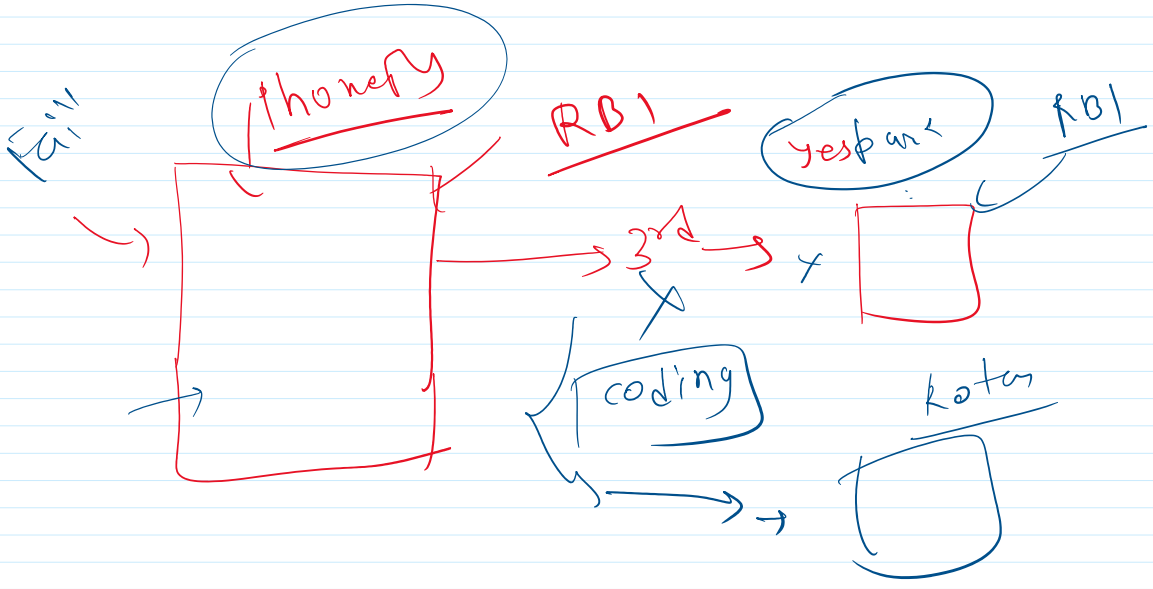
- **Ethical Considerations:**
  - Protecting privacy and data, ensuring accessibility, and avoiding biases.
  - Considering environmental impacts and responsible AI practices.
- **Professional Codes of Conduct:**
  - Honesty, transparency, confidentiality, continuous learning, and legal compliance.
  - Respect for intellectual property (copyrights, patents, trade secrets).
- **Software Licensing:**
  - **Proprietary vs. Open-Source:** Defines how software can be used, modified, and distributed.
  - Violating licenses can result in legal consequences.

SDLC Models

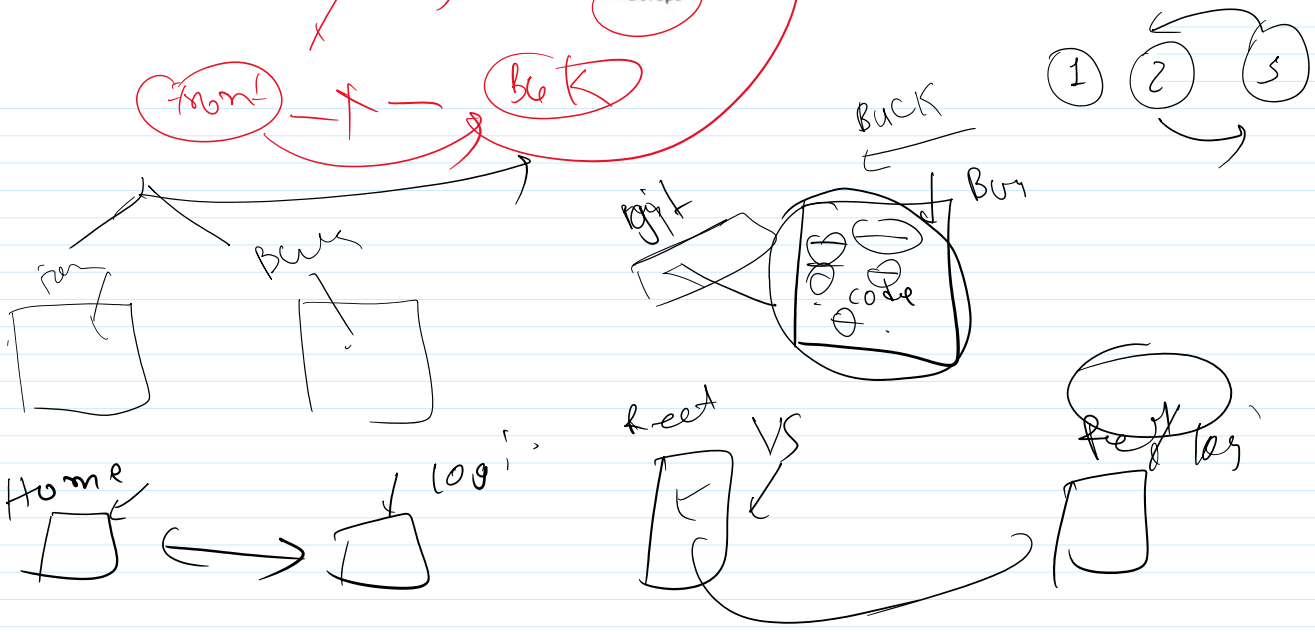
- **Waterfall Model:**
  - A sequential, linear approach.
  - **Use Case:** When requirements are well-defined and unlikely to change.
- **Agile Model:**

- A sequential, linear approach.
- **Use Case:** When requirements are well-defined and unlikely to change.

- **Agile Model:**
  - An iterative, flexible approach with short development cycles ("sprints").
  - **Use Case:** Projects with evolving requirements.
- **Iterative Model:**
  - Development through repeated cycles with continuous improvements.
- **V-Model:**
  - Emphasizes a parallel relationship between development and testing phases.
- **Spiral Model:**
  - Combines iterative development with risk analysis.
- **RAD (Rapid Application Development):**
  - Focused on quick prototyping and user feedback.
- **Real-World Example:**
  Companies like Amazon may use a hybrid of these models (often leaning towards Agile) to quickly adapt to market changes while ensuring robust quality control.
- **Interactive Activity (Matching):**
  Match the testing type with its description:
  1. Verifies interactions between modules – **Integration Testing**
  2. Tests the complete system – **System Testing**
  3. Checks individual units of code – **Unit Testing**
  4. Confirms software meets user expectations – **Acceptance Testing**
  5. Approach where tests are written before code – **Test-Driven Development (TDD)**



6 Phases of the Software Development Life Cycle



QA Engineer

- process - softw
- ensuring quality built
- plannins
- improvements
- Autom

Tester

- test case
- find bug
- requi