1

September 9, 2025

[ ]:

```
```

1 SDLC

Software
Development
Life
Cycle

Planning tech busi legal aspects Reqirement Analysis fr nfr Designing hld lld Coding
Testing
Deployment
Maintenance

The Software Development Life Cycle (SDLC) is a structured process for designing, developing, testing, and deploying software. It ensures high-quality software is delivered efficiently and meets business needs.

Phases of SDLC with Examples: 1. Planning Objective: Define the project scope, objectives, and feasibility. Activities: Identify business requirements. Perform feasibility studies (technical, economic, legal, etc.). Create a project plan. Example: A company wants to build an e-commerce platform. In this phase, stakeholders decide on features like product catalogs, payment integration, and delivery tracking. A feasibility study determines if the existing infrastructure can support such a system.

2. Requirement Analysis Objective: Gather and analyze software requirements. Activities: Consult stakeholders to document functional and non-functional requirements. Create a Software Requirement Specification (SRS) document. Example: Functional Requirement: The system must allow users to search for products by name or category. Non-functional Requirement: The system should load pages in less than 2 seconds.

3. Design Objective: Architect the system and prepare technical designs. Activities: High-Level Design (HLD): Overall system architecture. Low-Level Design (LLD): Detailed design of individual modules or components. Example: HLD: Decide on a microservices architecture with separate services for user management, inventory, and payments. LLD: Design a database schema with tables like Users, Orders, and Products.

4. Implementation (Coding) Objective: Convert design into working software by writing code. Activities: Developers code each module or component. Use version control (e.g., Git) to manage code. Example: Developers write the backend logic in Java (Spring Boot) and the

frontend in React to build the e-commerce platform. APIs are developed to fetch product details or process payments.

5. Testing Objective: Verify the software meets requirements and is free of defects. Activities: Perform different testing types: unit testing, integration testing, system testing, user acceptance testing (UAT). Fix identified bugs. Example: Unit Testing: Ensure the "Add to Cart" functionality adds the correct product. System Testing: Verify the platform handles concurrent user logins during a sale event. UAT: Stakeholders test the platform and confirm it works as expected.
6. Deployment Objective: Release the software to a production environment. Activities: Deploy the software to servers. Perform post-deployment checks. Example: The e-commerce platform is deployed on AWS using CI/CD pipelines for automated deployment. DNS configurations are updated so users can access the platform via a domain name like www.myecommerce.com.
7. Maintenance Objective: Ensure the software continues to function correctly post-deployment. Activities: Monitor system performance. Fix bugs or issues that arise. Implement enhancements or updates as user needs evolve. Example: After deployment, customers report that the "Order History" feature isn't displaying correctly. Developers fix the bug and release a patch update. New features like "Wishlist" are added based on customer feedback.

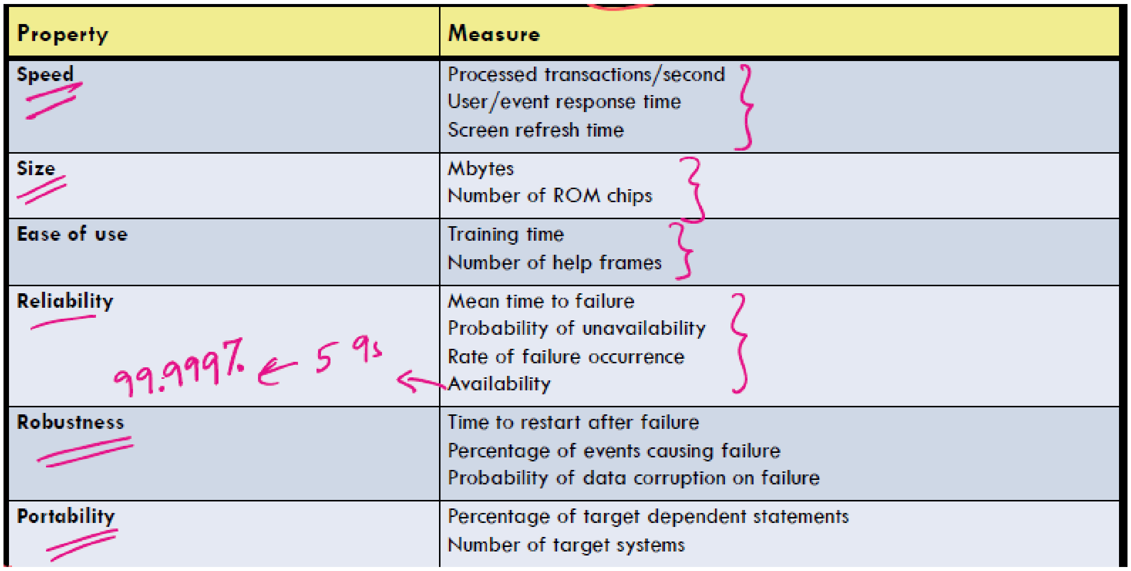
Can you explain the phases of the SDLC with examples?

Verification and Validation (V & V)

Functional Requirements Describe functionality or system services

These define system properties and constraints e.g. reliability, response time and storage requirements.

Constraints are I/O device capability, system representations, etc.



Property	Measure
Speed	Processed transactions/second User/event response time Screen refresh time
Size	Mbytes Number of ROM chips
Ease of use	Training time Number of help frames
Reliability 99.999% ← 5 9s ←	Mean time to failure Probability of unavailability Rate of failure occurrence Availability
Robustness	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Portability	Percentage of target dependent statements Number of target systems