PER SCHOLAS

Software Development Cycle Workbook

Software Development Lifecycle (SDLC)

The Software Development Life Cycle (SDLC) is a process the software industry uses to produce high-quality systems that meet or exceed the customer’s expectations.

Phases of SDLC

1a. Business Analysis 1b. Requirements Analysis
2. Design
3. Development/Coding
4. Testing
5. Deployment
6. Maintenance

1A— Business **Analysis**	Current state - so we know where we are now. Understanding the current state allows to identify the needs that need to be fulfilled by our project.
1B— Requirements Analysis	Customer describes what they want and how they want it used. Failure in this stage will result in incomplete or misdirected work in all the following stages. We need to have a clear understanding of: ● What we **want** to have (desire to have - initial plans). ● What we **must** have (absolutely necessary). ● What is just **gravy** (extra stuff - not expected enhancements). We need to have a picture in mind. “WANTS vs. NEEDS” and “CAN or WILL PAY FOR”
2— Design	How do we build it? Architectural Blueprints are created with the following considerations: ● Hardware: The physical components required for the software to operate. ● Software: The application will use operating systems, databases, and other software tools. ● Service/Product Availability: Ensuring the software or product is accessible when needed. ● Service/Product Capacity: Guaranteeing the software can handle the expected load or number of users. ● Security: Implementing measures to protect the software and data from unauthorized access or breaches. ● Environmental Considerations: Addressing any physical or regulatory environmental concerns affecting the software's operation or deployment. High-Level Design (HLD) vs. Low-Level Design (LLD) ● HLD outlines the functional aspects of various modules and the final result, setting the stage for how different system parts will interact. ● The LLD is responsible for defining the logic and execution plan for each module that has been outlined in the HLD.

3— Development Coding	Programming tools like compilers, interpreters, and language such as COBOL, C, and Java are used to code with respect to the application type. Team of programming experts to begin coding.
4— Testing	The testers test the quality of the software 1. Unit testing 2. Regression testing- tests the features unrelated to the new code after the new code is introduced 3. Smoke testing- run a sampling of the tests that cover the major parts of the app (but do not cover any part thoroughly) 4. Performance testing 5. Load testing 6. Security 7. Compatibility Testing 8. Functionality Testing
5— Deployment	Deployment moves the product/service into the “live” environment. Once deployed successfully, customers and users can be allowed to begin utilization of the functionality provided.
6—Maintenance	The maintenance phase is an ongoing process and includes the maintenance of hardware and software, updates, upgrades, and new features.

High-level Design	Lists the functional aspects of various modules, along with the final result. (More of the architectural aspect? What is it supposed to do? Does it meet the customer’s goals? General)
Low-level Design	Details logic and execution of each module in a high-level design (HLD) (More so of the coding aspect. All the tasks that are involved in the design. Granular)

User Experience Testing	How easily can users utilize the functionality provided by the software?
Functionality Testing	Type of testing that bases its test cases on the specifications of the software component under test. Does the software perform the tasks it was designed to do?
Security Testing	A process to determine that an information system protects data and maintains functionality as intended. Does the release meet the customer's acceptable risk profile, considering potential security vulnerabilities and data integrity issues?
Load Testing	The testing technique that puts demand on a system or device and measures its response. Will this service be able to support (x) number of users and maintain acceptable levels of performance?
Performance Testing	Functional testing is conducted to evaluate the compliance of a system or component with specified performance requirements. Will performance in a live environment meet the planned performance from the blueprint? Can the software support the expected number of users while maintaining acceptable performance levels? This is particularly important for applications that handle high volumes of concurrent users or transactions.
Compatibility Testing	Evaluate the effect of the software on the live environment, including any potential adverse impacts on existing services or systems.

What is the importance of ensuring users are properly trained?

Is a huge risk to successful implementation if users are not trained in *how* to interact with new systems prior to new rollout

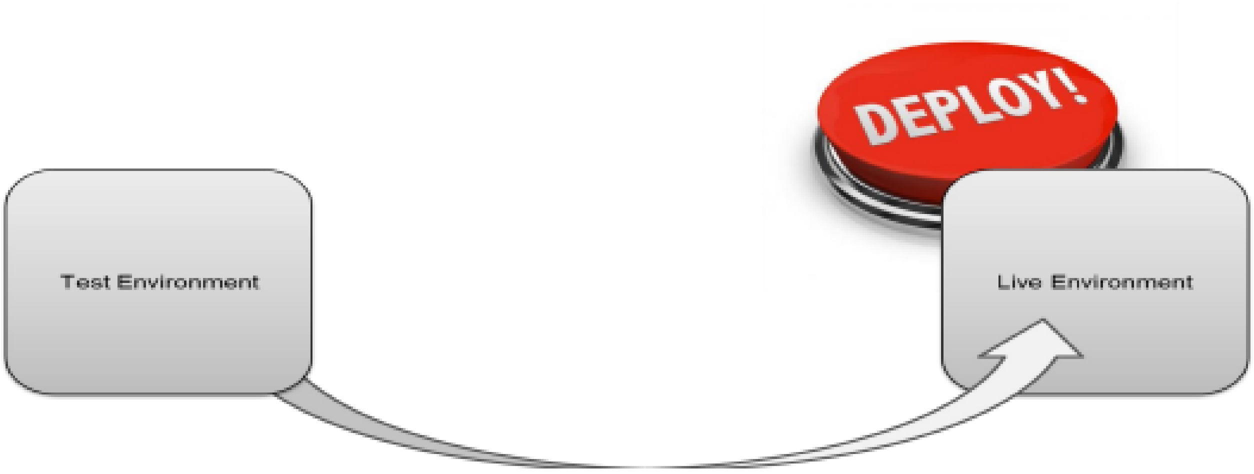
Preparing end-users to interact with the new system through training and familiarization is crucial to ensuring a smooth transition and mitigating implementation risks.

Why does this training need to occur before rollout?

To ensure end users are aware of how to utilize implementation before deployment and to answer any questions if need be.

What is meant by “deploy into the live environment''?

Making the product accessible to the customers and users. Once deployed successfully, customers and users can be allowed to begin utilization of the functionality provided.



What steps are necessary after deployment into the live environment to allow users to utilize the new functionality?

Regularly maintain and update the software to ensure its stability.