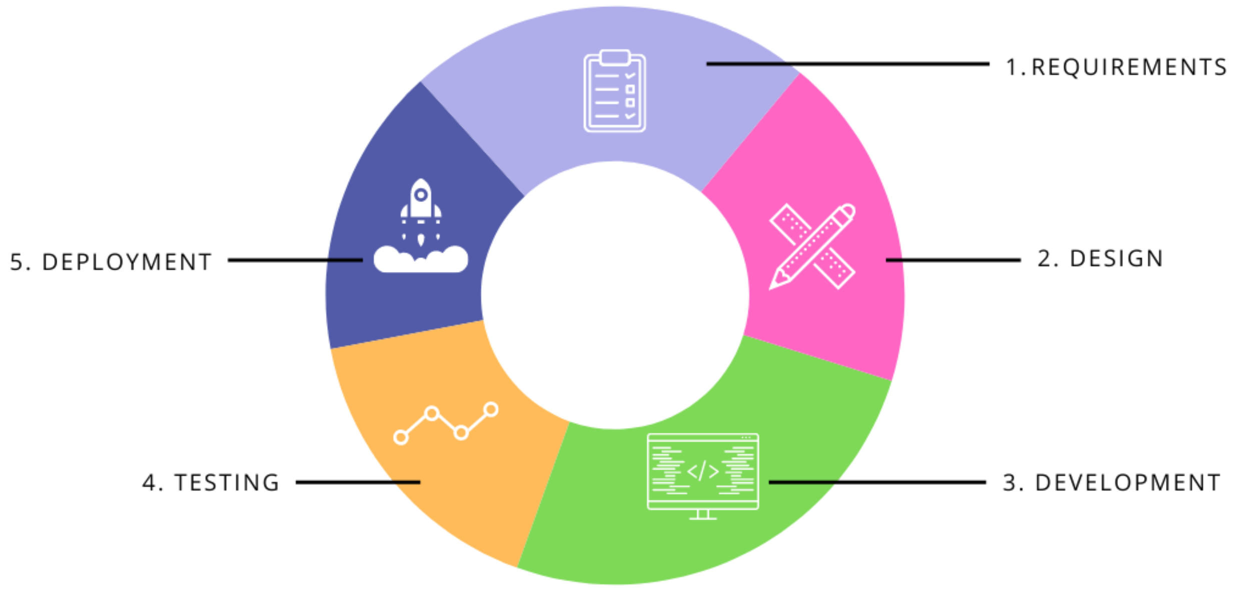
The Software Development Cycle

The Software Development Life Cycle (SDLC) is a series of stages one might complete while working on software. Each of these stages are vital to the process of creating a robust, well-maintained piece of software.

Read on to learn about all of the steps in the SDLC!


1. REQUIREMENTS
2. DESIGN
3. DEVELOPMENT
4. TESTING
5. DEPLOYMENT

1. Requirements

What is the Requirements stage?

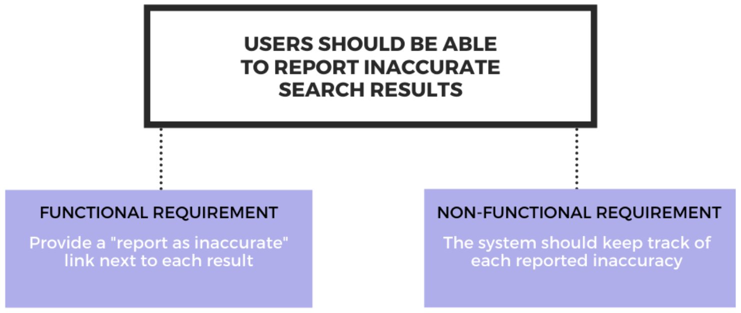
This is the stage where the clients and developers join forces to discuss what features to implement. “Client” is a term used to refer to the people who will be using your software or code. Clients can be external companies and other third parties, but they can also be other developers.

Given a client's wants, your task as an engineer is to translate them into technical requirements. Requirements can be either functional or non-functional.

- Functional requirements specify the features of a project and its components.
- Non-Functional requirements ensure the quality of the project and its components.

Say you're working with a client to build a search engine. Let's take a look at a sample client ask and the translation of that ask into technical requirements.

- Client Ask: "Users should be able to report inaccurate search results"
- Technical Requirements:
    - Functional Requirement: Provide a "report as inaccurate" link next to each result
    - Non-Functional Requirement: The system should keep track of each reported inaccuracy


USERS SHOULD BE ABLE TO REPORT INACCURATE SEARCH RESULTS
FUNCTIONAL REQUIREMENT
Provide a "report as inaccurate" link next to each result
NON-FUNCTIONAL REQUIREMENT
The system should keep track of each reported inaccuracy

For example, say a small business hires a tech consulting firm to create a mobile app for them. In this case, the small business is the client. Consider another scenario, in which a team of developers in a large tech company creates an API on the server side to be consumed by another team of developers on the front end side.

In most cases, the clients will explain what they or their users want to accomplish with the software, and the developers will determine whether that feature is feasible or not.

Another Meaning for "Client"
The term "client" can also be used to refer to the client-side (aka front-end) of an application. The opposite of client-side is server-side (aka back-end).

Who's usually in charge of this stage?

As mentioned above, this stage includes clients and developers. However, this stage can include additional parties like designers, product/project managers, business analysts, and any other interested parties (also called stakeholders).

The task of translating these client asks is usually performed by Business Analysts or Project Managers. As a software engineer, your job will be to implement the requirements and get the job done!

2. Design

What is the Design stage?

The design stage can entail two different meanings—it all depends on the type of development you're doing!

In front-end development, the design stage focuses on the aspects of the software that the user will see and interact with directly. During the design stage for a front-end developer, the user interface, user experience, aesthetics, accessibility, and other cosmetic features are at the forefront of this stage for front-end developers. Wireframes and prototypes make their appearances during this stage.

As for back-end developers, design refers to the architecture of the server-side and database of the software. Systems design, design patterns, and database architecture are three of the key fields of interest when it comes to design on the back-end. UMLs, design docs, and tech-stack consideration comes into play.

Who's usually in charge of this stage?

Both front-end and back-end developers are involved in the Design stage. Software Engineers in general are heavily involved in the design stage.

To get more technical, front-end design is usually handled by front-end developers, fullstack developers, and UI/UX engineers. On the other hand, back-end design is handled by back-end developers, fullstack developers, and database architects.

3. Development

What is the Development stage?

The development stage is the stage that coders can really sink their into! This stage is all about putting the wheels in motion and implementing the requirements and designs established in the previous stages.

Who’s usually in charge of this stage?

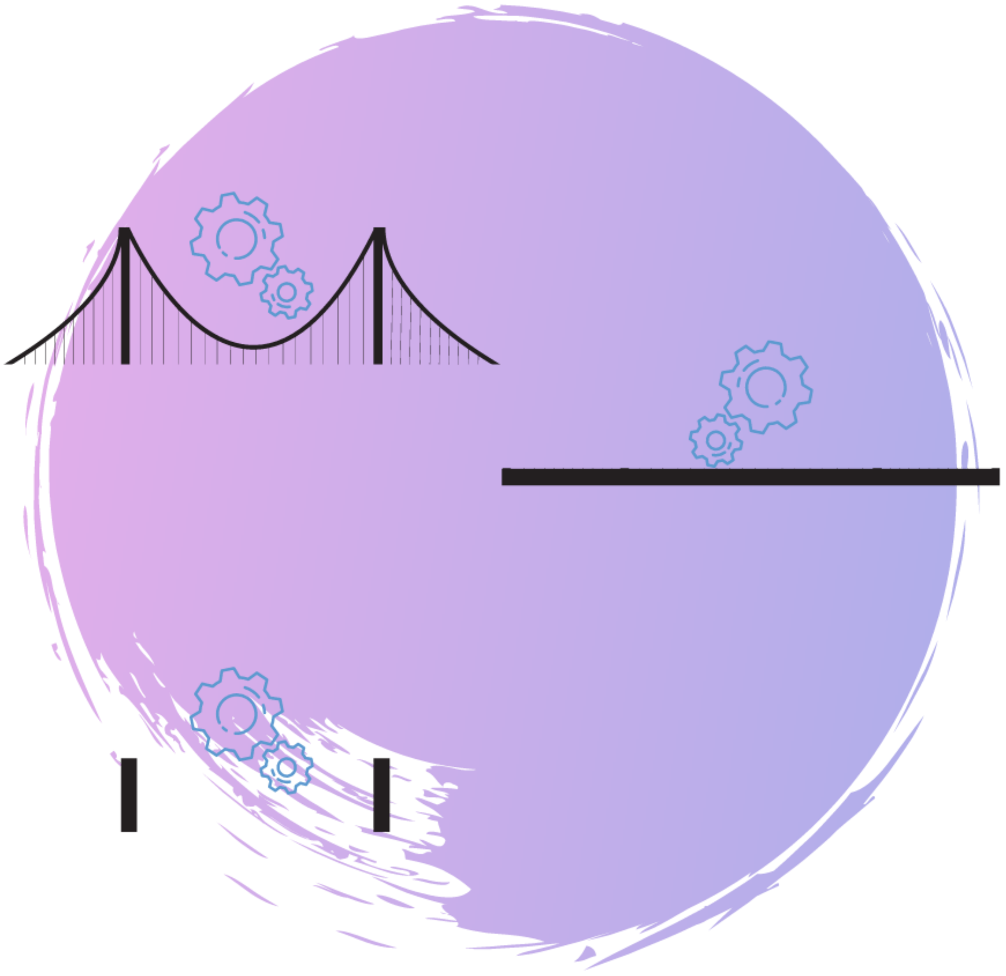
Software developers and engineers are tasked with writing all of the code necessary to get the requirements fulfilled.

Engineer vs Developer
Although the terms "software engineer" and "software developer" are used interchangeably, engineers are much more involved in the design stage than developers are. However, the duties for these roles vary from company to company.

4. Testing

What is the Testing stage?

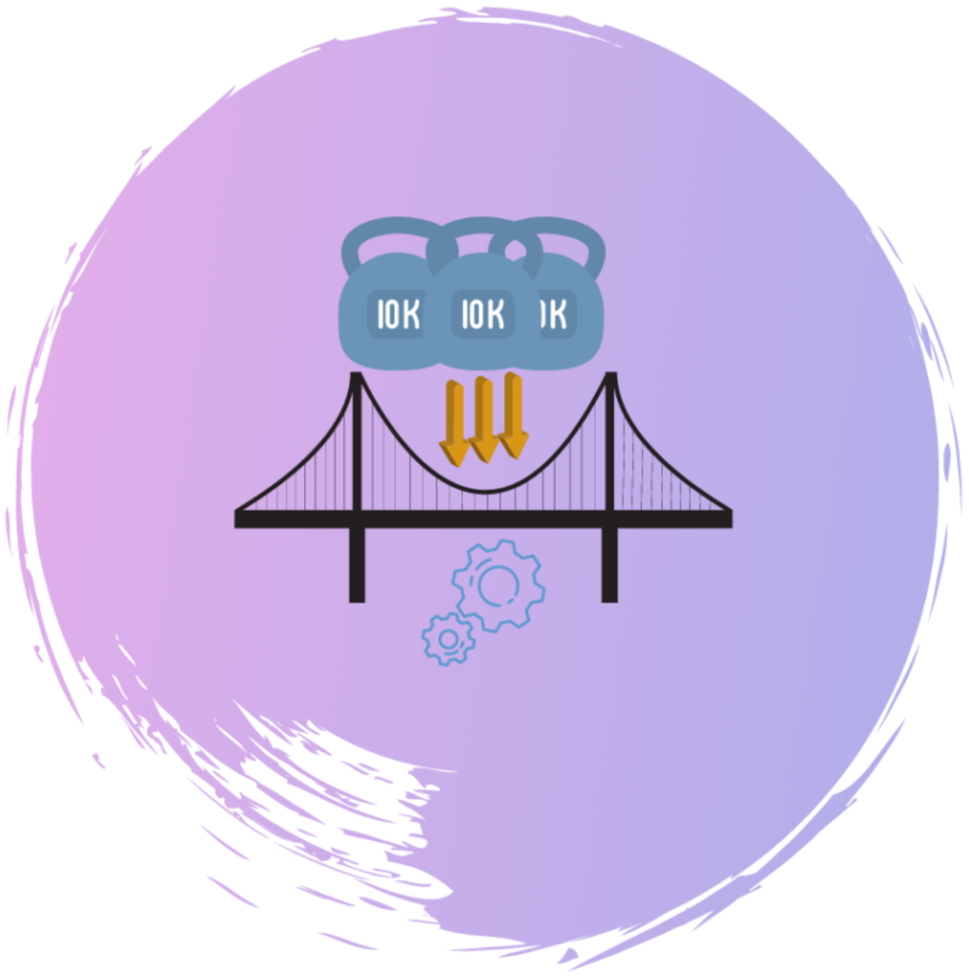
Arguably one of the most underrated stages in the SDLC, the testing stage is all about ensuring quality. There are many types of tests to keep in mind not only in the testing stage, but in the development stage as well. Here’s a quick overview of some of the types of tests you might encounter:



Unit Testing: The three units of this bridge are tested individually.

10K

Integration Testing: The bridge is tested as a whole.


10K
10K
10K

Load Testing: The bridge is placed under loads of weight to make sure it's safe for cars and foot traffic!

Who’s usually in charge of this stage?

This depends on the company. Some companies have designated roles (like Software Engineer in Test and Quality Assurance Engineers) that are dedicated to writing integration and load tests. Other times, Software Engineers/Developers are expected to write all types of tests.

In both cases, Software Engineers/Developers are expected to write their own unit tests.

5. Deployment

What is the Deployment stage?

Deployment covers all of the tasks necessary to get the software ready to be released to the client! The tech stack, platform (cloud, on-premise, serverless, etc), and release pipeline are all key components of the deployment stage.

Deploying to the cloud is the most popular form of deployment these days. Cloud-based platform as a service (PaaS) solutions provide tools for developers and engineers to build, run, and operate applications and software in the cloud.

Let’s break this down, shall we?

- platform as a service (PaaS): In the world of cloud computing, companies offer their computing powers to others as a service. Other “as a service”s include software as a service (SaaS), Blockchain as a Service (BaaS) and more!
- build: Building a project refers to the compilation of the source code into executable code.
- run: Running a project is exactly what it sounds like! It runs the executable code created in the build process and runs the application locally (only on the computer that’s running it)
- operate: If we want our application to run for everyone, and not just locally, then you have to buy a URL and get a host for your application. Deployment may require establishing a URL, web hosting, database hosting, error logging, and more.

Who’s usually in charge of this stage?

Underneath the general umbrella of “Software Engineer” lies roles such as Reliability Engineer, Release Engineer, and the more vague DevOps Engineer. All of these roles may be responsible for deploying the code that is written in the Development stage.

BONUS! Continuous Integration

What is Continuous Integration?

A major part of software engineering is establishing a developer “flow”. This is a series of steps you’ll take whenever working on code for a specific project or feature during the development stage. Continuous Integration helps teams facilitate these steps.

Continuous Integration, or CI, is a practice in which code changes are committed to a shared repository several times a day. Committing a change is like hitting the save button on Microsoft Word while you’re writing an essay

Just like how Microsoft Word highlights misspellings and provides tools for fixing formatting and improving word choice as you type, continuous integration builds the project every time you commit and push (aka upload) a code change and makes sure that all appropriate tests are passing.