5 Steps To become a Software Developer

By Liezel Shaw

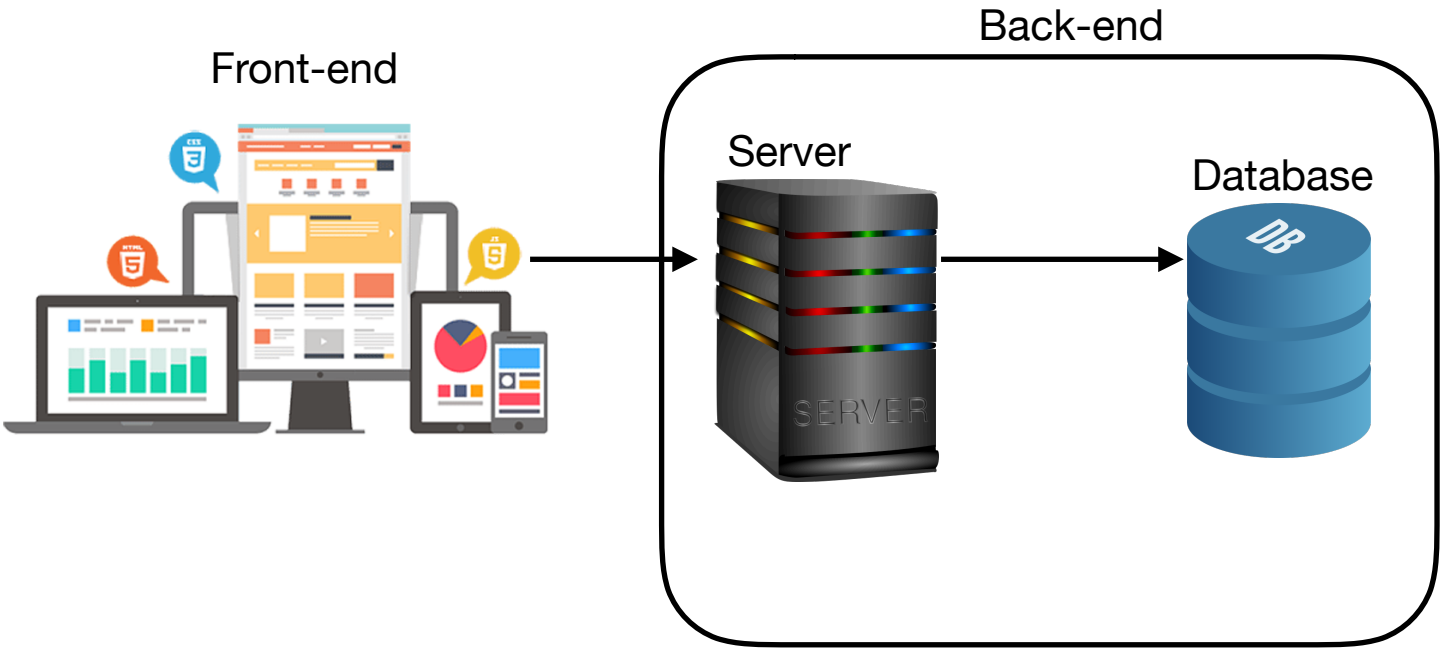
What is Software Development?

The Wikipedia definition for software development is as follows :

Software development is the process of conceiving, specifying, designing, programming, documenting, testing and bug fixing involved in creating and maintaining applications, frameworks, or other components.

From this definition you can see that software development is so much more than just programming/coding. Software development includes the full development lifecycle, from defining the specifications for an application to the eventual implementation of the fully-tested system.

A software application can roughly be divided into two parts, namely the Front-end and the back-end of an application. This diagram will give you a better overview.



The front-end is the part that the user sees and interacts with. It needs to be functional and user friendly to make it a pleasant experience for the user. A font-end can be a web-frontend/internet page, a mobile application or a desktop application.

The back-end consists of the server, which is the brain of the application. This is where all the back-end coding and logic takes place. The 'server' code is not necessarily a distributed system running on a server, it can also be the back-end of a desktop application that just runs on the desktop machine.

You would normally connect the front-end with the backend, either through RESTFul Web services, with server-side templates like Thymeleaf, if its a distributed system. The other part of the back-end is the Database where all the data for the application gets stored.

What does a software developer do?

You have probably heard the term 'Full Stack Developer' before. So, what is the difference between a full stack developer and a software developer? Well, there is none. A full stack developer would be someone who is proficient in both the front-end and back-end of an application. A software developer, by definition is a full stack developer.

A software developers might find that they might end up more proficient in one area of software development than another, and that is totally normal. No-one can be an expert in everything. The important thing is to have a basic knowledge of all parts of an application and then expert knowledge in a certain part that you then 'specialise' in.

If this is what you want to do, then follow these steps to become a software developer:

1 Become a lifelong learner

Software development is such a fast moving industry, that you need to stay on top of the industry trends, especially in the technologies that you are actively working with. You won't be able to stay on top of everything, but at least keep reading up on some of the latest trends, so you have a basic knowledge of what is happening in the software development world.

It is about the small steps, don't think about everything that you still need to learn, focus on one thing you want to learn and do it. After that pick another one and continue like that. Before you know it, you will have learnt a lot without even thinking about it.

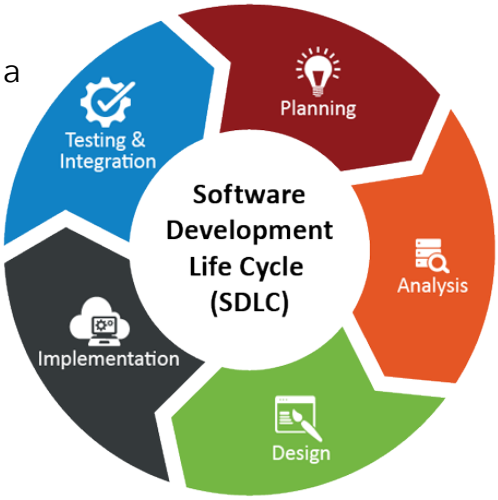
The best way to learn a new technology is by doing it. Pick a project you want to do and then start doing it. Learn, research and implement it as you go along. There is a lot of information on the internet that you can use to help.

If you want to fast-track your learning, you can also enrol on a course or a one-on-one training with a mentor to help you if you get stuck.

The best developers are the ones who keep learning new things on top of their normal work.

Albert Einstein said, *"Once you stop learning, you start dying." This is especially true in the software development world.*

2 Learn SDLC and Design Principles



The Software Development Life Cycle (SDLC) form a crucial part of a software developer's life. You need to know what it is and how to use it.

The Software Development Life Cycle is a process by which you develop high-quality software applications. It normally consists of 5 or more stages/ phases that you can implement in various ways.

The stages of the SDLC are as follows:

- Planning - this is where you find the the detailed requirements for the task/ project ahead. Useful tools for planning is a requirements specification and use cases.
- Design - here you design the system by looking at the architecture, design the front-end, choose tech to use, design back-end and data structures. Make sure to keep good design principles and patterns in mind during your design. Useful tools here would be UML Class diagrams, relationship and dependancy models, Sequence Diagrams, etc.
- Implementation - this is where you start coding your solution. Stick to the design created in the previous phase as much as possible. It does sometimes happen that you hit a snag in the design that no-one thought of before. It is then important to update your design first and then continue coding. Plan your code and then code your plan.

- Testing - this is where you test your code to make sure it works. If you are doing test-driven development (TTD) you are testing at the same time as you are coding. TTD means that you write a test first and then write the code to get the test to succeed.
- Installation/Deployment - Once everyone is happy with the application and have thoroughly tested it, to make sure there are no problems or bugs, it will be deployed to the production environment for the client to start using.

There are different SDLC models that companies use to implement the SDLC. Here are some of the more popular ones:

- Waterfall model
- Incremental Approach
- Agile Model
- Spiral Model

More on these models will be explained at a later stage.

Planning and design is a very important stage in the development process and should not be ignored. Don't be tempted to just start coding immediately.

"First solve the problem, then write the code" - John Johnson

3 Learn a back-end programming language

Most of your time will be spend doing 'back-end' code, unless you are specialise in front-end development.

Choosing which language you want to learn will largely depend on what kind of development you want to do and also personal preference. You will probably find that as you continue on your journey as a software developer, that you will eventually learn more than just one programming language.

If you are not sure which one to choose first, it would be a good idea to start with one of the mainstream languages first.

Here are some of the more popular server/back-end languages:

- Java - still one of the most popular languages, especially with big enterprises. As an Object Oriented Language, it is easy to learn and helps keep systems modular, extensible and flexible. It is platform independent, which means that it can run on any operating system. It also has lots of open-source libraries and frameworks to make your life easier. Java is by far the most versatile language
  - Uses:
    - Desktop, mobile and web applications
    - Embedded systems, ranging from tiny chips to specialised computers
    - Web and Application Servers (Enterprise applications)

- Scientific Applications for writing applications involving scientific and mathematical operations.

- Python - a general purpose and high level programming language. It is gaining hugely in popularity and is easy to learn. It is versatile and powerful and often used for Artificial Intelligence and Machine Learning.
  - Uses:
    - Desktop and web applications
    - Scientific and numeric computing
    - Support language for software developers
- PHP - a server-side scripting language. A lot of websites nowadays are using PHP for server-side programming.
  - Uses:
    - Web applications
    - Command line scripting
    - Server-side scripting

Other than just the language to write your code in, you should also know RestFul Web services as a means for the front-end to connect to the back-end of an application.

4 Learn about databases and data structures

A database is an organised collection of data stored and accessed electronically from a computer system. It is usually controlled by a database management system (DBMS).

Types of databases:

- Relational Databases - Data is organised as a set of tables with columns and rows. It provides the most efficient and flexible way to store and access structured information.
  - MySQL
  - Oracle DB
- NoSQL/Non-Relational databases - A non-relational database that allows unstructured or semi-structured data to be stored and manipulated.
  - MongoDB
  - Couchbase
- Object-oriented databases - Information is stored and represented in the form of objects as in object-oriented programming.
  - Cache
  - WakandaDB

The way you structure your data as well as which database to use, will largely depend on your expertise and the application you are writing. When designing your database, it is a good idea to have your database design to mimic your application's data model as closely as possible.

As a software developer, it is a good idea to familiarise yourself with all the different database types.

Relational databases are still the most popular by far. Structured Query Language (SQL) is the language you use to communicate with a relational database. You can ask for certain data, add data records or create databases and tables with it, amongst other more complicated queries . It is a good idea to learn SQL as most developers will work with relational databases frequently.

5 Learn Web/front-end languages

Front-end development is the development of the interface that the client uses to communicate with the system/application. It needs to be user-friendly and easy to use. The front-end code usually runs on a client's web browser or computer.

A front-end can be one of the following:

- A website or webpage connecting to a web server
- The user-interface of a desktop application
- A mobile application connecting to a web server

Front-end developers, however, mostly work with webpages and will use HTML, CSS and Javascript to write the webpages of a website, according to a design.

Even if you want to focus on back-end development, it is still a good idea to get to know and have a basic knowledge of these languages, so that you are able to create a website or web front-end when needed.

Here is a list of the web technologies that you will want to learn and familiarise yourself with:

- HTML - Hyper Text Markup Language. This is the language that you write a web page in
- CSS - Cascading Stylesheets. This is to tell the webpage what it should look like. It makes your website look pretty instead of just words on a blank page.

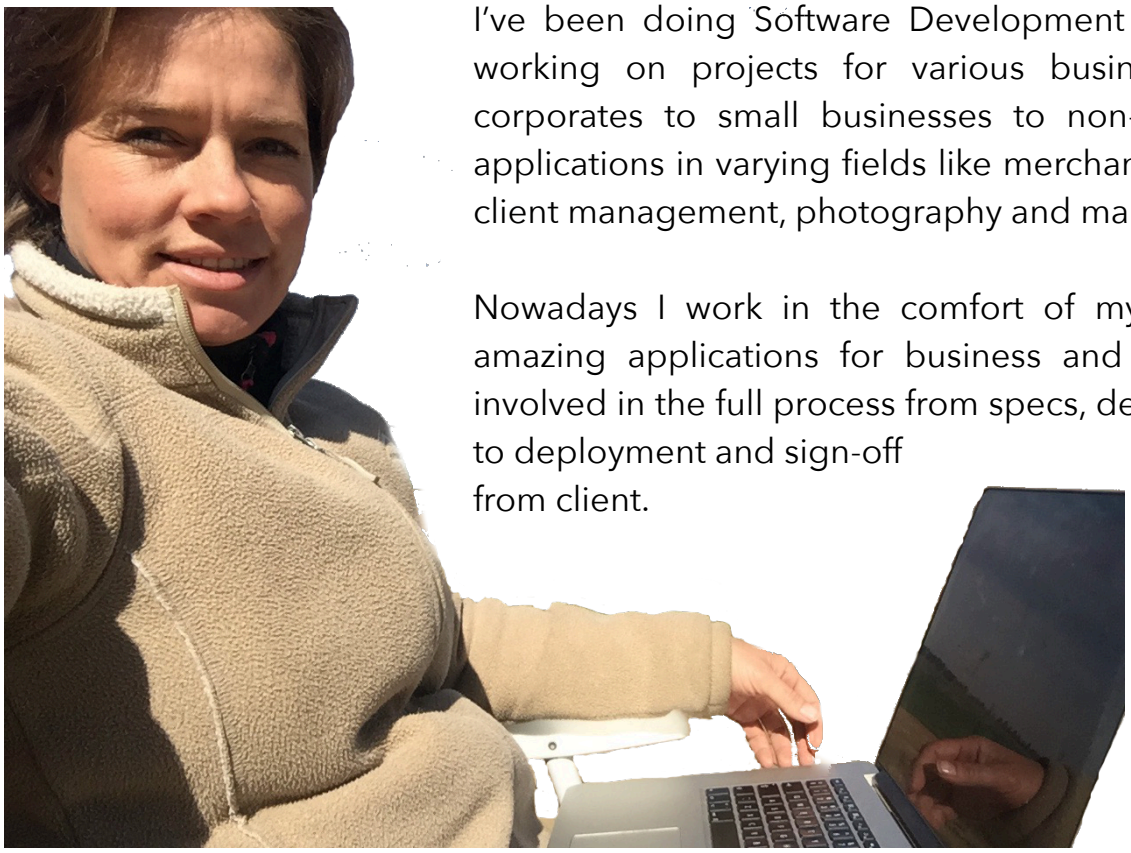
- Bootstrap - a CSS Framework for developing responsive and mobile-first websites
- Javascript - scripting language to make your webpage dynamic. Any moving parts, buttons, animations, etc. is thanks to Javascript.
- JavaScript libraries and frameworks like JQuery, AngularJS and ReactJS are popular and help you develop your javascript code faster and easier.

Conclusion

Now that you have all the steps to become a software developer, it's time to start working! Don't get overwhelmed, just keep taking little steps towards your goal. Keep moving forward. Learn a little bit every day and write code every day. Before you know it, you will be there!

About the Author

Liezel Shaw



I've been doing Software Development for the past 20 years working on projects for various business and clients from corporates to small businesses to non-profits. I've created applications in varying fields like merchant banking, accounting, client management, photography and marketing.

Nowadays I work in the comfort of my own home, creating amazing applications for business and private clients, being involved in the full process from specs, design and development to deployment and sign-off
from client.

In addition to these development projects, I also teach Java Software Development to new aspiring developers using HTML, CSS, Bootstrap, Java, Spring-boot and MongoDB or Mysql as well as all the other tools needed to help you with your software development journey.

I am passionate about helping people take the next step on their journey and teach them to invest in themselves and further their knowledge in the software development field.