Full Stack iOS Development with Swift and Vapor

Full stack iOS development made easy

Hem Dutt

www.bpbonline.com

First published: 2023

Published by BPB Online
WeWork
119 Marylebone Road
London NW1 5PU

UK | UAE | INDIA | SINGAPORE

ISBN 978-93-55518-323

www.bpbonline.com

Dedicated to

This book is dedicated to students and software engineers embarking on their journey as full-stack developers or exploring the dynamic iOS domain. May it inspire and empower you to unlock your full potential in this exciting field.

Together, let's embrace the world of full-stack iOS development and strive for excellence.

About the Author

Hem Dutt started his software engineering career in 2010 as a macOS (OS X) and iOS application developer and thereafter designed and developed numerous native macOS and iOS applications for various clients across the globe while working in multiple MNCs. With more than a continuous decade of experience working in macOS and iOS, Hem Dutt has developed and managed applications in multiple domains, including healthcare, insurance, VPN clients, publishing, IOT, and VoIP. His passion for designing and developing secure, reliable, and modular software is evident from his blogs, client awards/recommendations, and open-source projects. Prior to this book, he authored "Interprocess Communication with macOS: Apple IPC Methods," cementing his expertise in the Apple ecosystem.

Acknowledgements

I have to start by thanking my beloved wife, Payal Bhardwaj, for keeping me motivated throughout my journey as a writer. She is always by my side and supports me to fulfill my dreams, however impossible they seem to others. Thank you a ton, my dear, for being the pillar of my strength. You are a superwoman.

I must offer a very special thanks to my parents, who nurtured my childhood and, despite their limited means, provided me with the best they could possibly do and shaped my character.

I would also like to thank BPB Publications for giving me this opportunity, and a big thanks to technical reviewers and editors for helping me in shaping the chapters and content of this book.

I would also like to thank my colleagues and friends, who always believed in me and encouraged me.

Finally and most importantly, I would like to acknowledge my two beautiful children, Adwit and Anika. Thanks, Adwit, for being such a sweetheart and adorable baby and filling the environment with unconditional love, and Anika, for being such a powerhouse and my super girl!!. I love you both so much.

Preface

Welcome to "Full Stack iOS Development with Swift and Vapor." In this book, we embark on an exciting journey that combines the power of Swift programming language, Vapor framework, and iOS development to delve into the realm of full-stack iOS development.

In today's interconnected world, the demand for versatile developers who can seamlessly bridge the gap between the backend and frontend is skyrocketing. As the boundaries between server-side and client-side become increasingly blurred, mastering full-stack development has become a valuable skill set.

This book is designed to cater to a wide range of readers, from aspiring developers and students to seasoned iOS professionals seeking to expand their expertise. Whether you are taking your first steps in Swift or are already well-versed in the language, this book equips you with the knowledge and tools to navigate the world of full-stack iOS development with confidence.

We begin by laying the foundation, exploring the essentials of Vapor, Swift, and iOS app development. From there, we delve into backend development, covering topics such as persisting data, working with models, and integrating APIs. Simultaneously, we dive into frontend development, unraveling the intricacies of creating compelling user interfaces, networking, and authentication.

Throughout this journey, we emphasize best practices, security considerations, and performance optimization techniques to ensure that you not only build functional applications but also create robust, secure, and high-performing ones.

Real-world projects and hands-on exercises will guide you, allowing you to apply your newly acquired knowledge in practical scenarios. You will witness the power of integrating Swift and Vapor, leveraging their synergistic potential to develop cutting-edge full-stack iOS applications.

I invite you to embark on this transformative journey of becoming a full-stack iOS developer.

Chapter 1: Full-stack Development Overview – This chapter aims to give a basic understanding of the term Full Stack Development, a brief history of the term, and the concept of a minimum viable product. We will also explore the problems and advantages of Full stack development and provide a brief introduction to Swift on the server.

Chapter 2: Setting Up the Environment – This chapter aims to give a basic understanding of tools and SDKs to start with Vapor and iOS development. In this chapter, we will cover the installation of Xcode, Vapor Toolbox, and starter projects in Vapor as well as for iOS.

Chapter 3: Routing, MVC and JSON in Vapor – This chapter aims to give a basic understanding of creating Routes for the server application, a brief understanding of the MVC design pattern, and creating Controllers in a Vapor application. We will also explore JSON format and handling JSON in a Vapor app and extend this discussion, and we will also cover the Postman app for testing the Routes.

Chapter 4: Async and HTML Rendering in Vapor – In continuation of the last chapter, This chapter aims to extend the basic understanding of Async, Logging, Capturing Errors and Stack Traces, and finally, handling HTML rendering in a Vapor project. In this chapter, we will implement a small part of the code to showcase HTML rendering on a webpage using Leaf and Vapor routes.

Chapter 5: PostgreSQL Integration in Vapor – In this chapter, we will study the integration of PostgreSQL with Vapor. PostgreSQL is an open-source, relational database system that focuses on extensibility and standards. It is designed for enterprise use and also has native support for geometric primitives, such as coordinates which comes in handy working with Fluent, which also supports these primitives and saves nested types, such as dictionaries, directly into PostgreSQL.

Chapter 6: Building User Interfaces for iOS – The aim of this chapter is to understand the basic building blocks of iOS UI development and complete the circle of Full Stack Development with Swift.

Chapter 7: Data Persistence with Core Data and SQLite in iOS – Implement data persistence on iOS using Core Data with SQLite as a persistent store. In this chapter, we will write our very first Core Data implementation for storing data in an iOS app. After reading this chapter, readers will be able to Model data using Xcode's model editor, Add new records to Core Data, Fetch a set of records from Core Data, Display the fetched records, and learn the basics of Networking.

Chapter 8: Full Stack Implementation – We implemented small sample codes to understand working on Vapor and iOS app. All these samples were discussed in isolation to make it simple for you to grab specific concepts without worrying about the larger picture. In this chapter, we will specifically look into the larger picture and will look into Full Stack implementation of an app.

Chapter 9: Advanced Full-stack Concepts – In this chapter, we will explore some advanced topics related to full stack, which are very important with respect to overall system design and system architecture. These concepts are a must for commercial application development.

Chapter 10: Deploying iOS and Vapor Applications – The objective of this chapter is to study and understand the deployment process/es for our iOS and Vapor apps to the public. For Vapor apps, we will study deployment through Heroku and Docker, whereas for iOS, there is only one way, and that is through App Store, which will also be covered in this chapter.

Code Bundle and Coloured Images

Please follow the link to download the
Code Bundle and the ***Coloured Images*** of the book:

https://rebrand.ly/yqsj4yl

The code bundle for the book is also hosted on GitHub at **https://github.com/bpbpublications/Full-Stack-iOS-Development-with-Swift-and-Vapor.** In case there's an update to the code, it will be updated on the existing GitHub repository.

We have code bundles from our rich catalogue of books and videos available at **https://github.com/bpbpublications**. Check them out!

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Table of Contents

CHAPTER 1 Full-stack Development Overview

Introduction

This chapter aims to give a basic understanding of the term *full-stack development*, a brief history of the term, and that of a minimum viable product. We will also explore the problems and advantages of full-stack development and provide a brief introduction to Swift on the server.

Structure

In this chapter, we will cover the following topics:

- Introduction to full-stack development
- Brief history
- Minimum viable product (MVP)
- Problems with full-stack development
- Advantages of full-stack development
- Swift on server and Vapor
- Swift packages for back-end development

Introduction to full-stack development

We have heard the term full-stack developer in the software industry, typically referring to a Web developer who can build the front-end and back-end for a Web app. Instead of specializing, a full-stack developer is able to work across the back-end and front-end spectrum of app development.

It is an already established fact that being a specialist in one field or technology and gaining mastery in that particular aspect of technology has distinct advantages, but in the modern world, as technology is rapidly changing and evolving, many companies are seeking talented developers who are able to understand and work on the entire spectrum of the front and back-end technologies and are able to create a usable end product. Hacker Rank's survey on the most sought-after talent pool in 2020 provides a good insight into the demand for full-stack developers.

As per Hacker Rank Report: Across company sizes, hiring managers agree that full-stack developers are a top priority: 38% of hiring managers say it is the #1 role to fill in 2020. Back-end developers and data scientists were ranked second and third priorities, respectively.

The emphasis on full-stack developers was most pronounced in small companies (1–49 employees), 43% of which ranked the role as their top priority.

Though the qualities that define a *full-stack developer* are a subject of debate, most agree that they should have a basic understanding (or better) of all layers of a tech stack and should be able to generate a minimum viable product on their own. It is why they are especially important in small organizations, where fewer employees often have to do the job of many.

See the following figure (source: **https://info.hackerrank.com/rs/487-WAY-049/images/HackerRank-2020-Developer-Skills-Report.pdf**).

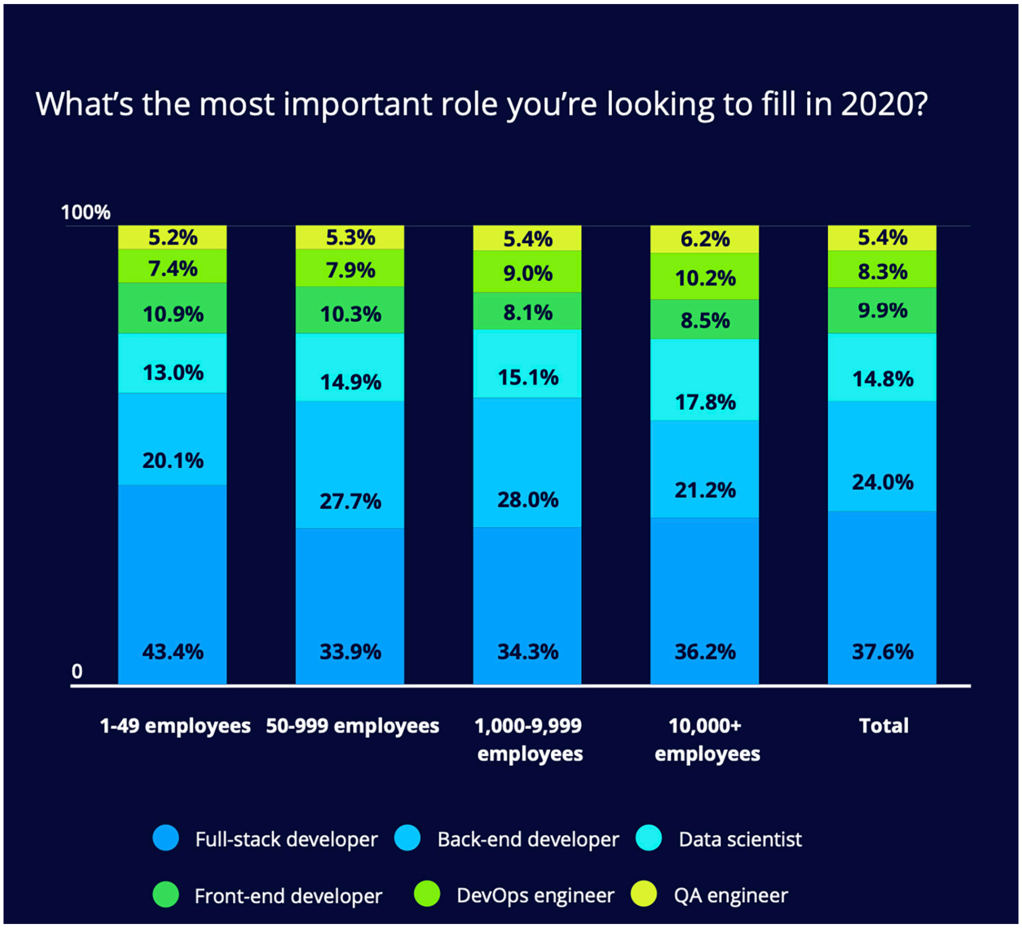

Figure 1.1: *The 2020's most in-demand talent pool*

As is clear from the report, these developers, also known as *full-stack developers*, are *once again* in demand. Does this mean this is not a new phenomenon? Yes, this role has a long history and has had its share of ups and downs, as well as arguments and disagreements from all kinds of people about what full-stack developer really means and what should be the level of expertise of the developer in different aspects of the stack.

Full-stack developers are useful as generalists who can quickly come up with a **minimum viable product** (**MVP**) on their own. They can also be very helpful in providing insight into the entire application infrastructure and contributing to all its parts. It is a sought-after ability for many roles in the software development industry.

Brief history

If we look at it from a high level, full-stack development has been part of the programming world since the very beginning, but it was not understood in its current context before.

The full stack development in the public domain only came to light in 2008, when designing for the Web as well as mobile became mainstream. Earlier, this term was used with varying understandings regularly in the 1970s as well as the 80s.

The main reason for this was that, at that time, there was not much difference between a back-end programmer and a front-end programmer. Slowly, with time, the distinction between front-end and back-end became defined, and two different streams of application development came into existence, that is, front-end and back-end development. In 2008, the term full-stack Web development gained momentum, and with passing years it has come to become one of the most in-demand job roles of present times.

According to Stack Overflow's 2021 developer survey, over 49.47% of developers describe themselves as full-stack. See the following figure (source: **https://insights.stackoverflow.com/survey/2021# developer-profile-developer-roles**):

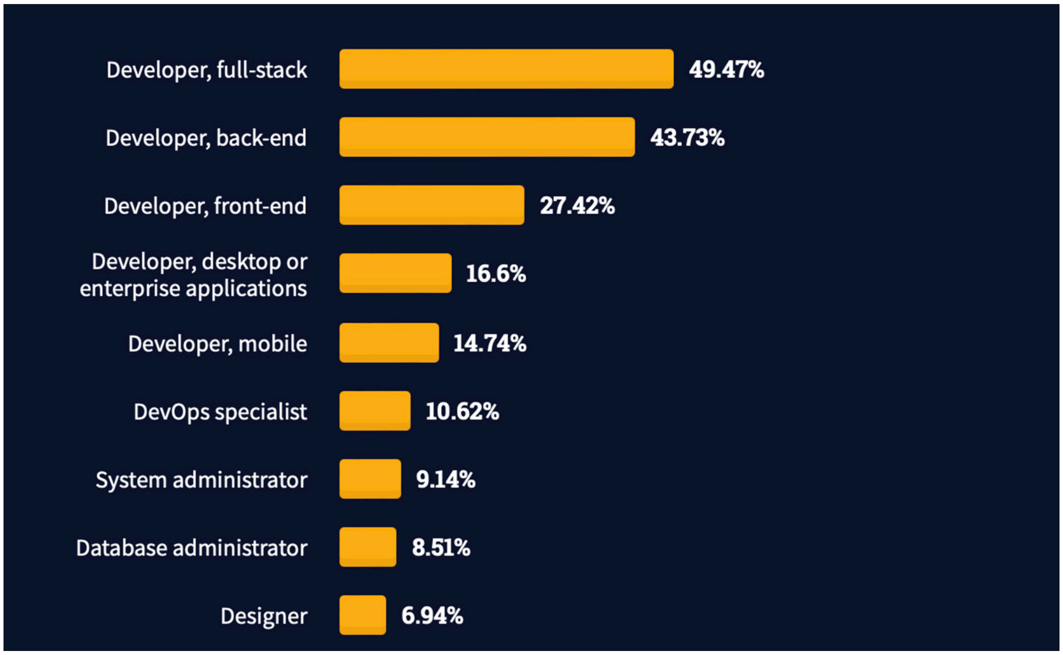

Figure 1.2: *Developer roles*

While during all these times, the term *full stack* has gained traction in the Web developer community, an obvious question is whether it can be applied to mobile application development. It is an interesting question, what a full-stack mobile app developer would mean?

As we know that mobile app developers work on the client side of the application or, in loose terms, front-end, and therefore, it might look perfectly sensible to assume that a mobile app developer simply needs the skill to develop a back-end to be a full-stack developer.

But this is not as simple as it looks, and we are going to explore why it is a lot more complicated in the context of an iOS developer.

Full-stack: what does it mean?

The term stack here refers to the collection of technologies needed to build an application. For example:

LAMP (**Linux, Apache, MySQL, and PHP**) or **MEAN** (**MongoDB, Express, Angular, and NodeJS**) or **MERN** (**MongoDB, Express, ReactJS, and NodeJS**), and so on are technology stacks having all the parts needed to build a minimum viable product of Web app.

To understand the term full-stack in terms of iOS development, let us use the MERN preceding example and substitute React with Swift to replace the front-end part in a Web app stack with native Swift. Therefore, a full-stack on iOS will look something like **MESN** (MongoDB, Express, Swift, and NodeJS).

Minimum viable product (MVP)

As discussed in previous sections, full-stack developers are useful as generalists who can quickly come up with a **minimum viable product** (**MVP**) on their own. Let us understand what an MVP is.

A minimum viable product, or MVP, is a product with only enough features to onboard initial targeted customers and validate a product-market fit for a business idea early in the product development cycle. In the software industry, the MVP can actually help the product team receive early user feedback and make it possible to iterate and improve the product.

The basic idea of agile methodology is built on a process for validating and iterating products based on short user input cycles, and so the MVP plays a central role in agile development.

MVP can be understood as the initial version of a new product that allows a team to get the maximum feedback and customer validation from customers with the least amount of effort.

A company might decide to develop and release a minimum viable product because of the following:

- The company wants to release the product to the market as quickly as possible with basic features to gain an early-mover advantage.
- The company wants to test the idea with real target customers before committing a large budget to the product's full development.

MVP has the following two distinct features:

- It has enough features for consumers to purchase the product.
- It has a feedback mechanism for users so that the company can collect real data for product-market fit.

If you are still wondering what this would look like in the real world? Let us go through the stories of a couple of brands that launched successful MVPs.

Airbnb

With no money to build the business, the founders used their own apartments to validate their idea of creating a market offering for peer-to-peer rental housing online. They created a minimalist website, did marketing about their property, and found several customers almost immediately.

Foursquare

The location-based social network Foursquare started with just a one-feature MVP, that is, offering only check-ins and gamification rewards. Foursquare's development and product team then added recommendations, city guides, and other features until they validated the idea with an ever-growing user base.

Problems with full-stack development

One of the problems with the term *full-stack* is that it does not exactly define the skill level needed from the developer across the stack. For example, how can we gauge the threshold skill needed from a full-stack iOS developer to develop a website at a bare minimum? A full-stack iOS developer should know how to put together a simple static website using HTML and CSS, let us say, for playing a YouTube video URL within the App.

But if the developer is working on a complex social networking app that will require an admin portal to control user's permissions based on various parameters and which will also require a lot of other complex user flows such as authentication, data storage, and APIs.

Both of these scenarios will need a huge shift in terms of expertise needed in various stacks. Generally speaking, the expectation from a *full-stack* iOS developer is to have deep expertise in the iOS domain and basic knowledge of how to put together simple Web apps using HTML and CSS.

At the other end of iOS app development, there are hybrid app developers who use frameworks like React Native and Flutter to develop Web and mobile apps. It seems much easier to earn the title "full-stack" going the hybrid way, but native iOS app development has its own merits, and hybrid and native app developers are generally not the same.

We also need to understand that, in practice, a full-stack iOS developer might not complete a real project on his/her own. Although theoretically possible, an individual developing all parts of a project means a lot of risks. In practice, a full-stack iOS developer is a generalist who has a deep understanding of one or two components of the full-stack and a high level of knowledge of the rest. This makes a full-stack iOS developer suitable for creating minimum viable projects, proof of concepts, and leading an overall project from a high level.

The fact that there is no well-defined and concrete definition of a full-stack developer and the role requires continuous juggling of technologies is validated by Hacker Rank's survey 2020. As per the survey, full-stack developers are required to learn new skills most often.

As per Hacker Rank Report: Full-stack developers may be in the highest demand, but their role is also one of the most professionally demanding. Sixty percent of full-stack developers were required to learn a completely new framework or platform in the last year—more than any other role polled.

Full-stack developers also have to learn the most languages: 45% reported that they had to pick up a new one within the last year. Their peers have to learn more about theoretical concepts; data scientists and DevOps engineers were required to learn new concepts most often (33%).

With expertise that spans front-end, back-end, and more (depending on whom you ask), full-stack developers have one of the more nebulous job descriptions in the technical world. The relative flexibility of their role—and the breadth of

technologies they have to keep up with as a result—means learning on the job never stops.

See the following figure (source: **https://info.hackerrank.com/rs/487-WAY-049/images/HackerRank-2020-Developer-Skills-Report.pdf**):

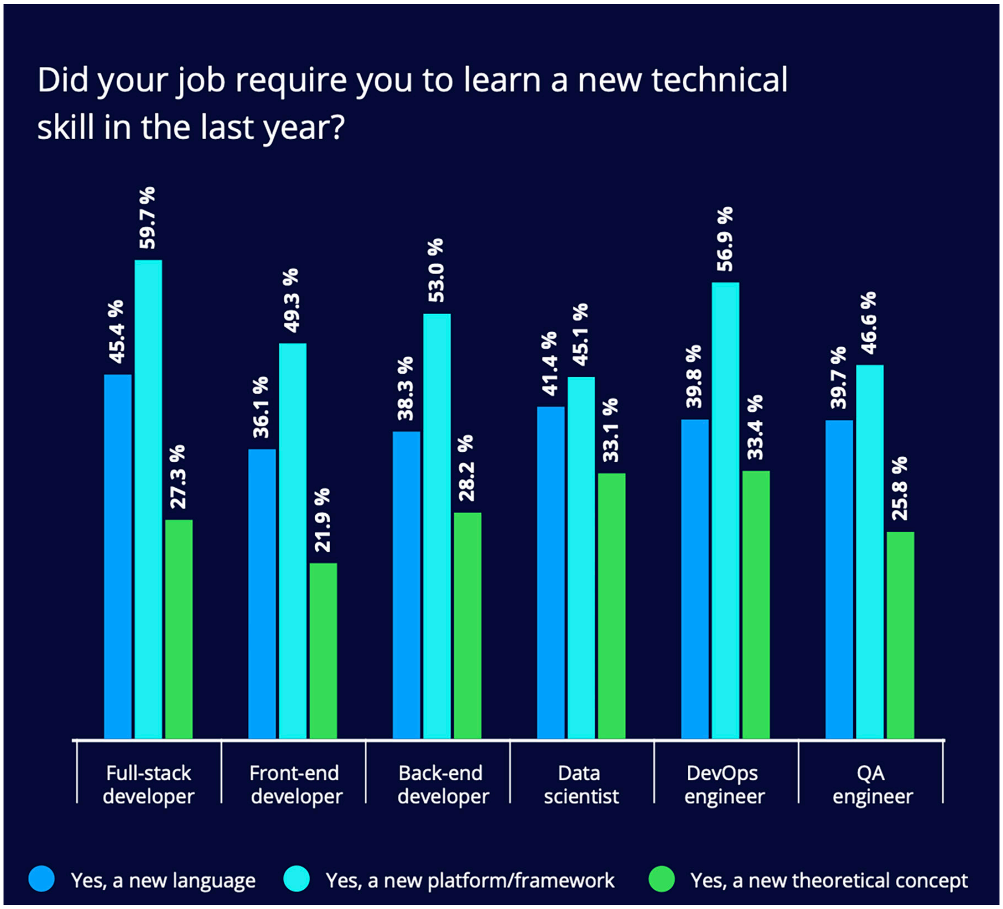

Figure 1.3: *Full-stack developers are required to learn new skills most often*

As is evident from the data, full-stack development is gaining traction, but it has its own unique problems, even more aggravated in the case of Full-stack mobile development. So, as it always happens in such cases, the industry is divided on this.

People on the anti-full-stack developer side are raising their voices with an argument for what does or does not constitute full-stack development. The anti-full-stack argument is pivoted around the idea that a full-stack developer should have the ability to easily navigate between the back-end and front-end development with a high level of expertise.

The anti-full-stack argument says that to be really effective, full-stack developers should be able to:

- Write high-quality code for the client side and should be at par with a senior client-side developer.
- Write equally high-quality code for the server side and should be at par with a senior server-side developer.
- Manage the infrastructure and deployment on the server side.
- Manage client application releases (on App Store in case of iOS app).

And while many developers can do some work that covers both disciplines, very few can do both well.

So, against full-stack development, the argument is that a truly full-stack developer is almost impossible to find, and while too many people boast of themselves as full-stack developers, they do not have full-stack qualifications in reality.

In a way, it seems that it is an unrealistic demand. A true full-stack developer should have *dual mastery* of both the client and server side, which is almost impossible, given the speed at which new technology is evolving. The against full-stack developer argument is that this encourages wide breadth, shallow depth knowledge and does not allow an individual to attain expertise.

Advantages of full-stack development

As discussed in the previous section, there are visible problems in understanding full-stack development and what the expectation should be from a full-stack developer. For all practical purposes, we can understand full-stack development with a broader interpretation of the term. The idea that a full-stack developer has to be an expert in every layer of the tech stack is an expectation, and instead, if they have working knowledge of the entire stack, with expertise in only a few layers, this should be good enough for all practical implications.

We can see the definition of *full-stack* development for a less strict set of requirements, described as follows:

- Comfortable with writing both client-side and back-end code with moderate expertise in one and deep expertise in another.
- Can Generate a minimum viable product (MVP) with minimal support from others.
- Provide expert-level specialty in either client or server side.