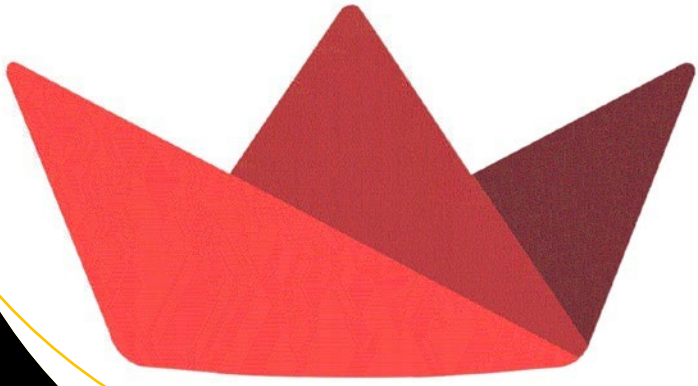


Web Application Development with Streamlit

Develop and Deploy Secure and Scalable Web Applications to the Cloud Using a Pure Python Framework

Mohammad Khorasani
Mohamed Abdou
Javier Hernández Fernández

Apress®

Web Application Development with Streamlit

Develop and Deploy Secure and Scalable Web Applications to the Cloud Using a Pure Python Framework

Mohammad Khorasani
Mohamed Abdou
Javier Hernández Fernández

Apress®

Web Application Development with Streamlit: Develop and Deploy Secure and Scalable Web Applications to the Cloud Using a Pure Python Framework

Mohammad Khorasani
Doha, Qatar

Mohamed Abdou
Cambridge, United Kingdom

Javier Hernández Fernández
Doha, Qatar

ISBN-13 (pbk): 978-1-4842-8110-9
ISBN-13 (electronic): 978-1-4842-8111-6
https://doi.org/10.1007/978-1-4842-8111-6

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: James Robinson-Prior
Development Editor: James Markham
Coordinating Editor: Jessica Vakili

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on the Github repository: https://github.com/Apress/Web-Application-Development-with-Streamlit. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

To my parents Yeganeh and Daryoush and to my departed grandparents Ghamar and Reza.

—Mohammad Khorasani

To my family, friends, and the open source community.

—Mohamed Abdou

To my family and friends for their support.

—Javier Hernández Fernández

Table of Contents

About the Authors

Mohammad Khorasani is a hybrid of an engineer and a computer scientist with a Bachelor of Science in Mechanical Engineering from Texas A&M University and a master's degree in Computer Science from the University of Illinois at Urbana-Champaign. Mohammad specializes in developing and implementing software solutions for the advancement of renewable energy systems and services at Iberdrola. In addition, he develops robotic devices using embedded systems and rapid prototyping technologies. He is also an avid blogger of STEM-related topics on Towards Data Science – a Medium publication.

linkedin.com/in/mkhorasani/

Mohamed Abdou is a software engineer with diverse academic and industrial exposure, a graduate of Computer Engineering from Qatar University, and currently an Software Development Engineer at Amazon. Mohamed has built a variety of open source tools used by tens of thousands in the Streamlit community. He led the first Google Developer Student Club in Qatar and represented Qatar University in national and international programming contests. He is a cyber security enthusiast and was ranked second nationwide in bug bounty hunting in Qatar in 2020 among under 25-year-olds.

linkedin.com/in/mohamed-ashraf-abdou/

Javier Hernández Fernández specializes in the area of technology innovation and brings over 20 years of practical experience in overseeing the design and delivery of technological developments on behalf of multinational companies in the fields of IT, telecom, and utilities. He publishes extensively, speaks at conferences around the world, and spends his days wading through piles of academic papers in the hope of finding something interesting. He holds master's degrees in both Energy Management and Project Management, in addition to a BSc in Computer Science from the Faculty of Engineering of the University of Ottawa.

linkedin.com/in/javier-hernandezf/

About the Technical Reviewers

Rosario Moscato has a master's degree in Electronic Engineering (Federico II University, Naples) as well as a master's degree in Internet Software Design (CEFRIEL, Milan). He also has a Diploma in Apologetics (Pontifical Athenaeum Regina Apostolorum, Rome) and a master's degree in Science and Faith (Pontifical Athenaeum Regina Apostolorum, Rome). Rosario has gained over 20 years of experience, always focusing his attention on the development and fine-tuning of the most innovative technologies in various international companies in Europe and Asia, covering various highly technical, commercial, and business development roles.

In recent years, his interest has focused exclusively on artificial intelligence and data science, pursuing, on one hand, the goal of enhancing and making every business extremely competitive by introducing and supporting machine and deep learning technologies and on the other hand, analyzing the ethical-philosophical implications deriving from the new scenarios that these disciplines open up.

Rosario has authored two books, and he is a speaker at international research centers and conferences as well as a trainer and technical/scientific consultant on the huge and changing world of AI.

Currently, he is working as Senior Data Scientist with one of the biggest multinational IT companies in the world.

Randy Zwitch is a data science and analytics professional with broad industry experience in big data and data science. He is also an open source contributor in the R, Python, and Julia programming language communities.

Acknowledgments

This undertaking would not have been possible without the support and efforts of a selfless few. Individuals and entities who in one way or another have made a contribution to the contents of this book are named as follows in no particular order:

- *Streamlit*: The folks who created the framework itself, empowering countless developers
- *Iberdrola*: Which provided the inspiration and time for us to put Streamlit to a very noble use
- *Iberdrola Innovation Middle East*: The folks who served as a test bed for our very first Streamlit ventures and had to put up with our constant pitching of Streamlit's formidability
- *Qatar Science & Technology Park*: Which has fostered an environment conducive to innovation and research
- *Daniel Paredes, Jerome Dumont, Ana Martos, and Gustavo López-Luzzatti*: For being our very first Streamlit users
- *Dr. Nikhil Navkar*: For being another trailblazing Streamlit user

In addition, a tangible part of our careers and personal endeavors would have simply been inconceivable without the spirit of the open source community. It is therefore in order to give a special tribute to Python and its respective developers, in addition to the multitude of other online forums who are silent heroes. Without their efforts, all-nighters would be every other night and our works not nearly as neat as they are.

Preface

It was an inconspicuous night like any other. I was about to doze off, but right before that happened, my phone buzzed, and being a millennial, I just could not resist the temptation to check. However, much to my disdain, it was just another pesky email advertisement recommending an online course in something called "Streamlit." I am a strong disbeliever in email lists, and, honestly, I would have investigated no further had it not been for the aesthetically pleasing Streamlit logo. In retrospect, I am glad I clicked on the ad. Since then, my programming life has one way or another been intertwined with a framework that I had been yearning for someone to create for years – the formidable Streamlit.

I noticed early on in my career that there is a plethora of seasoned Python developers that are adept with backend and server-side programming but cannot develop frontend user interfaces and client-side software to save their lives, myself included. While there were noble efforts made by the teams in Flask and Django, both frameworks solicit an abundance of exposure and technical know-how of HTML, CSS, and HTTP, which effectively rendered them as no-fly zones. I found myself making do with the likes of Tkinter and PyQt, thereby limiting myself to local desktop software with no means of deploying my work to the cloud. This is the predicament that I and a multitude of other programmers faced. What all of us Python loyalists needed was a pure Python web framework with an intuitive API that enabled the prompt creation and deployment of web applications while allowing the developer to focus on the backend. Similar perhaps to what ReactJS is to JavaScript. And when I clicked on that pesky ad mentioned earlier, that is exactly what I found. It was a sort of eureka moment!

Mind you, this anecdote of mine occurred in the summer of 2020, and Streamlit had only been released to the public in the fall of 2019. But less than a year of development by their team had rendered exactly the sort of framework and API that I had hoped for. And ever since then, this product has only been moving in one direction – upward, with a steep incline. For myself personally, I could not have discovered it at a more auspicious time. I had just been hired by Iberdrola and tasked with the audacious goal of creating and deploying a Python-based application to the Web. In a pre-Streamlit world, I would have fervently resisted the notion of deploying applications to the Web, but armed with my new friend, I found myself routinely advocating for the development of web applications while passionately brandishing Streamlit's untethering capabilities. Overnight, I had been transformed into a trailblazing member of sorts within our development team.

With all good things in the world, it just does not feel right to proceed without sharing the goodness with the world at large. Consequently, I have made it a subtle goal in life to inform the online and offline software development community of the empowerment that Streamlit ushers in. This book is the culmination of that effort, and more specifically it is intended for those who have faced the same hurdles as I have, and it will provide a holistic overview of Streamlit. This book will guide the reader through the life cycle of creating scalable web applications of their own, from the most basic use cases to crafting complex and distributed applications on the cloud.

In addition to learning all the ins and outs of Streamlit itself, after perusing this book, readers should be able to interface their web applications with robust server-side infrastructure like MongoDB, PostgreSQL, Linux, Windows Server, and Streamlit's own deployment platform. In a nutshell, you should be able to walk away from this book feeling empowered enough to unleash your ideas and to embody them on the Internet. Perhaps this could be the beginning of your next startup for the curious inner entrepreneur in you.

Mohammad Khorasani
September 2021

Acronyms

aaS As a service
API Application programming interface
BLOB Binary large object
CLI Command-line interface
CPU Central processing unit
CRUD Create, read, update, and delete
CSP Cloud service provider
CSRF Cross-site request forgery
CSS Cascading Style Sheet
DI Dependency injection, a coding pattern
DG Delta Generator, a core module in Streamlit
DOM Document Object Model
DTW Dynamic time warping
GPU Graphics processing unit
HTML Hypertext Markup Language
IDE Integrated development environment
ISP Internet service provider
JSON JavaScript Object Notation
JWT JSON Web Token
MLaaS Machine learning as a service
NAT Network address translation
ORM Object-relational model
OS Operating system
PID Process identifier
RCE Remote Code Execution
RDP Remote Desktop Protocol

REST Representational state transfer
SaaS Software as a service
SCADA Supervisory Control and Data Acquisition
SQL Structured query language
SQLI SQL injection
SSH Secure Shell
TPU Tensor processing unit
UI User interface
URI Uniform Resource Identifier
URL Uniform Resource Locator
UX User experience
VPN Virtual private network
XSS Cross-site scripting

Intended Audience

This book assumes that you have no less than a basic level of exposure and understanding in the following areas:

- Object-oriented programming
- Data structures and algorithms
- Python and the following bindings:
  - Pandas
  - Numpy
  - Plotly
- SQL (both relational and nonrelational databases)
- Git version control frameworks
- Cloud computing

In order to materialize on the concepts divulged in this book, it is imperative that you possess sufficient experience in programming. If you have little to no prior experience in the areas mentioned previously, then you should consider first enrolling in an online crash course of your choice that would offer at the least an introductory level of exposure. Other than that, by no means must you be an expert of any sort in the aforementioned concepts, although experts also stand to gain from the contents of this book. Even if you have the ability to develop applications with a more sophisticated stack, you may still appreciate the amount of time and energy that is saved by utilizing Streamlit. With Streamlit, you can render

a robust web application in hours what would have previously taken you weeks to produce in Flask or Django. In simpler terms, it offers a lot more bang for the buck.

Notwithstanding, it is important to clarify that for those who are looking for a means to deliver highly bespoke and tailored frontend user interfaces, perhaps Streamlit is not what you should be scouting for. Not that it will not address your needs someday, it is just that "someday" is not today. Streamlit is a work in progress, and while their team perseveres relentlessly, we should remain patient and expect a greater degree of customizability alongside a multitude of additional features to be released in the near future. Should you need something more amenable, then you may find that Django is a more suitable option. Mind you, Django lands you back in the realm of the predicament mentioned earlier, as you will need to be a more advanced programmer to create web applications with it.

Hopefully upon completion of this book, you should be equipped with the tools that you will need to produce a scalable web application deployed to the cloud from inception to deployment and operation. You will become confident in addressing the functional and performance requirements of developing both server-side and client-side software. You will be able to create both backend functionality and frontend user interfaces. In addition, you will learn to interface your software with relational and nonrelational database systems such as PostgreSQL and MongoDB in order to scale your application on demand. And finally, you will acquire the technical know-how to orchestrate and provision your scalable application on the cloud using Microsoft Server, Linux containers, and Streamlit's own cloud service.

While this book will go into great depth and breadth of the required concepts, a degree of self-learning and research is still expected of any reader. There will be gaps in tutorials, and perhaps some of the tools used will be deprecated or obsolete by the time you are reading this

book; therefore, a great deal of intuition and sound judgment is solicited. Furthermore, this book will not attempt to provide any explanation for the source code used by Streamlit or any other binding/API. Our scope will be solely limited to the application of the objects, classes, and functions included in each of the tools used. Each tutorial included will address a separate use case or application with the corresponding code provided. All original source codes published in this book are released under the MIT License and are open source. It is anticipated that you will utilize the whole or part of the methodology of the provided book to fulfill your own technical requirements.

Additional Material

This book is accompanied with an abundance of online material including repositories, datasets, libraries, APIs, and their corresponding documentation. Wherever necessary and possible, the URL to the material will be provided. All tutorials and source code included in this book will be available at the following repository: https://github.com/. Finally, any reference made to the Streamlit API can be additionally found on https://docs.streamlit.io/library/api-reference.

CHAPTER 1

Getting Started with Streamlit

With the inundation of data, and the pace at which it is created, traditional computing methods possess limited means to deliver results. On the other hand, cloud computing acts as an enabler, allowing one to overcome the limitations of the former. With increased scalability, reduced costs, and enhanced adaptability, cloud service providers, developers, and users alike stand to gain from the fruits of migrating to the cloud.

Given that Python is currently the scripting language of choice for most of the software development community, it is absolutely vital to provide a web framework for developers that would bridge their skills gap. While legacy frameworks such as Flask and Django solicit a firm understanding of HTML and CSS, Streamlit is the first major framework that is all pure Python, thereby reducing development time from weeks to hours.

1.1 Why Streamlit?

Limiting oneself to local computing options is simply a figment of the distant past, given that the cloud unlocks a host of advantages and allows developers to leave a tangibly greater impact on the world. It is for this very reason that an entire new echelon of developers are beginning to embrace everything cloud, and the rapid transition toward this new

M. Khorasani et al., *Web Application Development with Streamlit*,
https://doi.org/10.1007/978-1-4842-8111-6_1

paradigm of computing is a testament to that fact. And this is exactly where a pure Python web framework such as Streamlit can deliver a lot of value to developers who want to make the transition but who need an enabler to do so.

1.1.1 Local vs. the Cloud

The cloud is rapidly becoming synonymous with data itself. Quite literally, wherever there is an abundance of data, it is somehow intertwined with cloud computing. Put into layman's terms, it is simply inconceivable to harness the value of big data without utilizing the resources of the cloud. Gone are the days where one would brandish Microsoft Excel to create a 1990s style dashboard for their dataset. With the sheer magnitude of data exposed to us, local computing will not make the cut.

Even still, there are certain tangible benefits to local computing, namely, prototyping an idea is considerably faster, and latency between nodes and servers is several orders of magnitude smaller. This is why edge computing still holds an "edge" over the cloud, no pun intended. In addition, there are applications for which security is sacrosanct and/or regulations are inhibiting, and in that case, one may be better off disconnecting themselves from the world at large. Other than that, there is not much else that one can attribute to the list of benefits of local computing. However, there is a plethora of reasons why you should steer clear of it. To name a few, the bottom line will be quite lower with the overheads solicited to run the infrastructure at hand. There will also be very limited adaptability to fluctuating traffic; imagine the half-time surge in traffic during the Super Bowl, you simply will not be able to scale to meet demand.

On the other hand, cloud computing is cheaper to provision, exceedingly adaptable to demand, highly robust against failure, and supremely reliable to run. In addition, cloud computing platforms enable scalability in two dimensions – horizontal and vertical. Horizontal for

when you need multiple instances of the same computing resources and vertical for when you need bespoke resources, the likes of GPUs, TPUs, database systems, and more. Perhaps, the single most salient factor about the cloud is that it expands your horizons and allows you to offer your product as a service on the World Wide Web. The latest paradigm shift in tech is to offer quite literally anything as a service. The as-a-service model (aaS) can be applied to software (SaaS), machine learning (MLaaS), and any other product that can be offered as a web application on the Internet. And this is precisely where a web framework such as Streamlit fits the bill. Streamlit is a cloud enabler for those of us in the software development community that have been unable to deploy our value to the Internet due to a hefty skills gap that has thus far impeded us from doing so. Concisely put, Streamlit is a means of empowerment for developers at all levels.

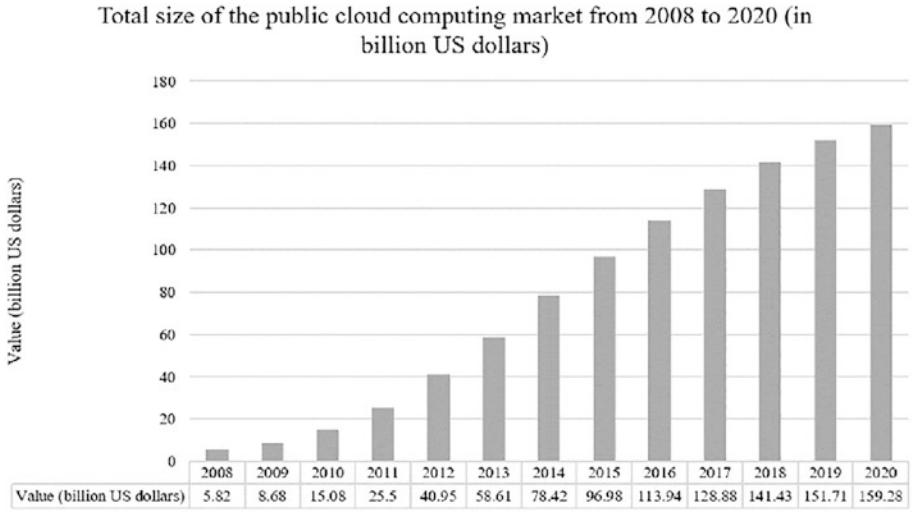
1.1.2 A Trend Toward Cloud Computing

Cloud computing is north, south, east, and west. Academia, the corporate world, governments, and even spy agencies are shifting from local to the cloud at a breathtaking pace. With legacy software providing limited means for growth and return on investments, organizations are increasingly migrating to cloud service providers who in turn offer agility, economies-of-scale, and advanced computing resources. Even CSPs themselves are jumping on the bandwagon, with the likes of Google and Microsoft offering their legacy applications on the cloud with Google G Suite and Microsoft Office 365.

Evidently, from a business perspective, the justification is even more so robust. Disruptive businesses are as pervasive as they have ever been and have wholly championed what others had so far been reluctant to accept. Therefore, embracing the cloud is no longer a subjective decision but rather a means for survival. With reduced lead time, scalability, limited capital investments, and increased innovation, businesses stand to gain a great deal. From a CSP perspective, this proposition is possibly even

more enticing. Equipped with the cloud, CSPs are able to pool resources together, increase resource elasticity, and reduce maintenance and overheads. And perhaps most importantly, from a consumer's point of view, the cloud is the best thing that ever happened since the Internet itself. As a consumer, the SaaS model offers unparalleled flexibility, price granularity, and plenty more bang for the buck. In other words, the cloud is win-win-win; everyone stands to gain.

While the trend toward cloud computing was already on a healthy trajectory, a once-in-a-century pandemic doubled as a booster toward everything cloud. The pandemic all but destroyed the stigma associated with remote learning, online examinations, working from home, and other dogma that decades of noble efforts by the tech community had failed to make a dent into. It is fair to assume that from here onward, the progression toward everything cloud will exceed the prediction of pundits. And if numbers alone can serve to indicate where we are heading, then one should feast themselves on the omen of Figure 1-1.



	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020
Value (billion US dollars)	5.82	8.68	15.08	25.5	40.95	58.61	78.42	96.98	113.94	128.88	141.43	151.71	159.28

Figure 1-1. *Growth of the public cloud computing market from 2008 to 2020. Source: statista.com [13]*

1.1.3 History of Web Frameworks in Python

Web development can be quite an arduous task, soliciting a multidisciplinary team with expertise in frontend, backend, and server-side software development. It is for this very reason that full-stack developers who brandish the know-how for the entire development process are in surplus demand and are often in receipt of a handsome compensation.

Traditionally, these developers would resort to using JavaScript, PHP, or Perl to develop web applications, with Python being sidelined as a local scripting language. This was due to the fact that Python is not inherently designed to run on the Internet and requires a web framework to interface with web servers and browsers. Over the years however, the community has developed several novel frameworks that allow Python to be utilized effectively for the Internet. And given Python's emphasis on simplicity, readability, rich ecosystem of libraries, and being open source, it has steadily transformed into one of the main web scripting languages of choice for many developers. Python has even gained enough traction to being adopted by heavyweights, the likes of Google and Instagram.

Generally, such web frameworks come in two types, full-stack and nonfull-stack. They manage everything from communications and infrastructure to other lower-level abstractions required by web applications. Nontrivial applications require a slew of functionalities including but not limited to interpreting requests, producing responses, storing data, and rendering user interfaces. For such applications, one would often utilize a full-stack framework that provides an in-house solution for all the technical requirements. This contrasts with nonfull-stack frameworks otherwise known as microframeworks that provide a bare minimum level of functionality, typically limited to routing HTTP requests to the relevant controllers, dispatching the controller, and subsequently returning a response. Such frameworks are usually stacked with other APIs and tools in order to create applications. Some of the most popular examples of each type are listed in the following sections.

1.1.4 Flask

Flask was developed in 2010 by Armin Ronacher from what was allegedly an April fool's joke to begin with. Flask is a nonfull-stack or microframework that provisions an application server without offering much else in terms of components. Flask is composed of two main elements: *Werkzeug*, a tool that lends support for HTTP routing, and *Jinja*, a template engine used to render basic HTML pages. In addition, Flask uses *MarkupSafe* as a string handling library and *ItsDangerous* as a secure data serialization library to store session data as cookies.

Flask is a minimalistic framework that is equipped with the bare minimum of the components required to render a web application. Consequently, the developer is afforded a great deal of autonomy and also responsibility to create their own application. As a result, Flask is best suited for static websites and for experienced developers who are able to provision most of their own infrastructure and render their own interfaces.

1.1.5 Django

Django was developed by a group of web programmers in 2003 who used Python to build web applications. It allows developers to create complex applications with relatively less overhead in comparison to Flask. Specifically, Django enables programmers to render dynamic content with enhanced scalability and with in-house capabilities to interface with database systems using object-relational mapping.

In addition, there are a host of other modules including but not limited to ecommerce, authentication, and caching packages that allow the developer to readily provision extended services. Bundled with a multitude of other third-party packages, Django allows the developer to focus mainly on the idea while not having to worry much about the implementation.