RADemics

Creating AI Powered Web Applications with Flask Django and Streamlit

Archana Anandrao Nikose, Sneha Sujit Dhande, B. Persis Urbana Ivy
PRIYADARSHINI BHAGWATI COLLEGE OF ENGINEERING, MOTHER TERESA INSTITUTE OF ENGG AND TECHNOLOGY

Creating AI Powered Web Applications with Flask Django and Streamlit

[1]Archana Anandrao Nikose, assistant professor, computer science and engineering, Priyadarshini Bhagwati College of Engineering, Nagpur, Mobile number: 824 800 2831, Mail id: nikose.ar@gmail.com.

[2]Sneha Sujit Dhande, Assistant Professor, Computer Science and Engineering, Priyadarshini Bhagwati College of Engineering, Nagpur, Mobile number: 824 800 2831, Mail id: snehadhakl16@gmail.com.

[3]B. Persis Urbana Ivy, Dean (CSE & Allied branches), Mother Teresa Institute of Engg and Technology, Melumoi (Post), Palamaner - 517408, Mail id: urbana23@gmail.com.

Abstract

The convergence of artificial intelligence (AI) and web development has catalyzed a new generation of intelligent applications capable of real-time decision-making, personalized interaction, and dynamic data visualization. Despite significant progress in AI model training and evaluation, the deployment of these models into production-ready web systems remains a complex and underexplored domain. This book chapter presents a comprehensive investigation into the deployment strategies of AI models using three prominent Python-based frameworks—Flask, Django, and Streamlit. Each framework offers a unique set of capabilities tailored to different stages of AI application development, ranging from lightweight microservices to scalable full-stack platforms and interactive analytical dashboards. The chapter examines critical deployment metrics such as latency, throughput, and scalability, providing an in-depth comparative analysis of how each framework performs in operational environments. Real-world use cases—including natural language processing, image classification, recommendation systems, and decision support—are explored to illustrate practical implementation patterns. In addition, the chapter highlights architectural trade-offs, integration workflows, and security considerations essential for reliable and maintainable AI-powered web systems. Through this research-driven synthesis, the chapter addresses a key gap in the literature by bridging the development-to-deployment divide, offering actionable insights for researchers, developers, and industry practitioners seeking to operationalize machine learning models effectively.

Keywords: AI deployment, Flask, Django, Streamlit, web applications, model integration

Introduction

The rapid proliferation of artificial intelligence (AI) technologies has significantly transformed the digital landscape, driving a paradigm shift from static data processing to intelligent, adaptive, and context-aware applications [1]. This evolution has created unprecedented opportunities for integrating machine learning (ML) models into web-based platforms, enabling real-time inference, personalized content delivery, and intelligent automation [2]. considerable advancements in the development of sophisticated AI models, their deployment into production-ready web systems

remains a critical bottleneck [3], [4]. Bridging this gap requires not only technical acumen in AI but also deep expertise in web development frameworks that can support scalable, maintainable, and secure AI application infrastructures [5].

The demand for deploying AI models in live environments has necessitated the exploration of various Python-based web frameworks that align with different use cases and system requirements [6]. Flask, Django, and Streamlit have emerged as three of the most prominent tools in this domain, each catering to a distinct subset of application needs [7]. Flask is lauded for its lightweight nature and modular design, making it suitable for microservices and custom API endpoints [8]. Django, with its comprehensive ecosystem, is preferred for building full-stack, scalable applications with robust security and database management [9]. Streamlit, in contrast, is optimized for rapid development of interactive dashboards and data exploration tools, tailored specifically for data scientists and AI researchers. Understanding the strengths, limitations, and appropriate contexts for each of these frameworks is imperative for effective AI deployment [10].

In this context, performance metrics such as latency, throughput, and scalability are central to evaluating deployment strategies [11]. AI web applications, particularly those offering inference services, must be capable of handling concurrent requests with minimal delay [12]. Latency affects user experience, especially in applications involving real-time predictions such as image classification, fraud detection, or recommendation engines [13]. Throughput is critical in high-volume systems such as health diagnostics or financial forecasting platforms, where rapid response across numerous sessions is mandatory. Scalability ensures that the deployed AI services can adapt to increasing workloads without degradation of performance [14]. Thus, framework selection must be aligned with these metrics to ensure efficient, user-centric AI services that can evolve with changing operational demands [15].