Faysal Aziz

Rapid Prototyping with Streamlit & Gradio

Turn Python code into interactive AI apps in hours, not weeks

FA-0395 · two-day

Software Development · DevOps, Cloud & Infrastructure

Introduction

The gap between building a model and delivering something people can actually use is where many AI and data projects stall. Streamlit and Gradio solve this problem by allowing developers to create interactive, web-based applications directly from Python code, without traditional front-end complexity. These tools have become essential in modern AI teams for rapid prototyping, internal demos, proof-of-concepts, and even production-ready tools.

This two-day course focuses on practical, industry-driven workflows that show how professionals rapidly move from notebooks and scripts to usable applications. The instructor brings over 30 years of industry experience and teaches proven patterns used in real AI teams, avoiding academic abstractions in favor of what actually works under real constraints.

Learning outcomes

By the end of this course, learners will be able to:

- Understand the role of rapid prototyping in AI and data product development.
- Build interactive data and AI applications using **Streamlit**.
- Create lightweight ML and AI demos using **Gradio**.
- Design user inputs, outputs, and layouts for non-technical users.
- Integrate Python models, scripts, and pipelines into interactive apps.
- Handle basic state, performance, and user interaction challenges.

- Decide when to use Streamlit vs. Gradio based on use case.
- Deploy and share prototypes efficiently with stakeholders.

Prerequisites

Participants should have:

- Solid fundamentals in Python (functions, data structures, basic modules).
- Basic familiarity with data manipulation or machine learning workflows.
- Experience running Python scripts and installing packages.

Training outline

1. 1. Why Rapid Prototyping Matters in AI Projects

- The gap between models and usable tools
- Prototyping vs. production systems
- Common AI project failure points
- Where Streamlit and Gradio fit in modern workflows

2. 2. Foundations of Interactive Python Apps

- Event-driven thinking for Python developers
- Inputs, outputs, and reactive execution models
- Stateless vs. stateful application behavior
- Common UX mistakes in technical prototypes

3. 3. Getting Started with Streamlit

- Streamlit application architecture
- Running and structuring a Streamlit app

  Core UI components
  - Text, numbers, sliders, buttons
  - File uploaders and selectors

  Layout and organization
  - Columns, containers, and sidebars

4. 4. Building Data & AI Prototypes with Streamlit

Displaying data

- Tables, dataframes, and summaries

Integrating Python logic

- Data processing pipelines
- Model inference workflows

Visualization basics

- Charts and plots

- Handling user-driven recomputation

5. 5. State, Performance & Usability in Streamlit

- Understanding reruns and execution flow
- Managing state across interactions
- Caching strategies
- Performance considerations for larger datasets and models
- Structuring code for maintainability

6. 6. Introduction to Gradio

- Gradio’s design philosophy
- Component-based interface building

  Inputs and outputs

  - Text, images, audio, files

- Function-first development model
- Quick demos vs. structured applications

7. 7. Building AI Demos with Gradio

- Wrapping ML and AI functions
- Creating interactive model demos
- Handling preprocessing and postprocessing
- Managing multiple inputs and outputs
- Customizing layouts and behavior

8. 8. Streamlit vs. Gradio: Choosing the Right Tool

- Strengths and limitations of each framework
- Typical industry use cases

  Decision criteria
  - Audience
  - Complexity
  - Deployment needs
- Combining both tools in a workflow

9. 9. Deployment & Sharing Prototypes

- Local vs. cloud-based deployment concepts
- Sharing apps with teams and stakeholders
- Security and access considerations
- Common deployment pitfalls

10. 10. Industry Patterns & Best Practices

- Designing prototypes for feedback, not perfection
- Making demos understandable to non-technical users
- Avoiding overengineering
- Transitioning from prototype to production systems

© 2026 Faysal Aziz · Kuala Lumpur