Building Your First AI-Powered Web Application Using LLMs

Welcome to this hands-on workshop! We'll guide you through creating a functional AI web application from scratch using Large Language Models.

This roadmap breaks down the entire process into manageable steps. By the end, you'll have a working application you can share with others.

By Atef Masmoudi

Understanding LLM Fundamentals

1

What are LLMs?

Large Language Models are AI systems trained on vast text data. They can understand and generate human-like text.

2

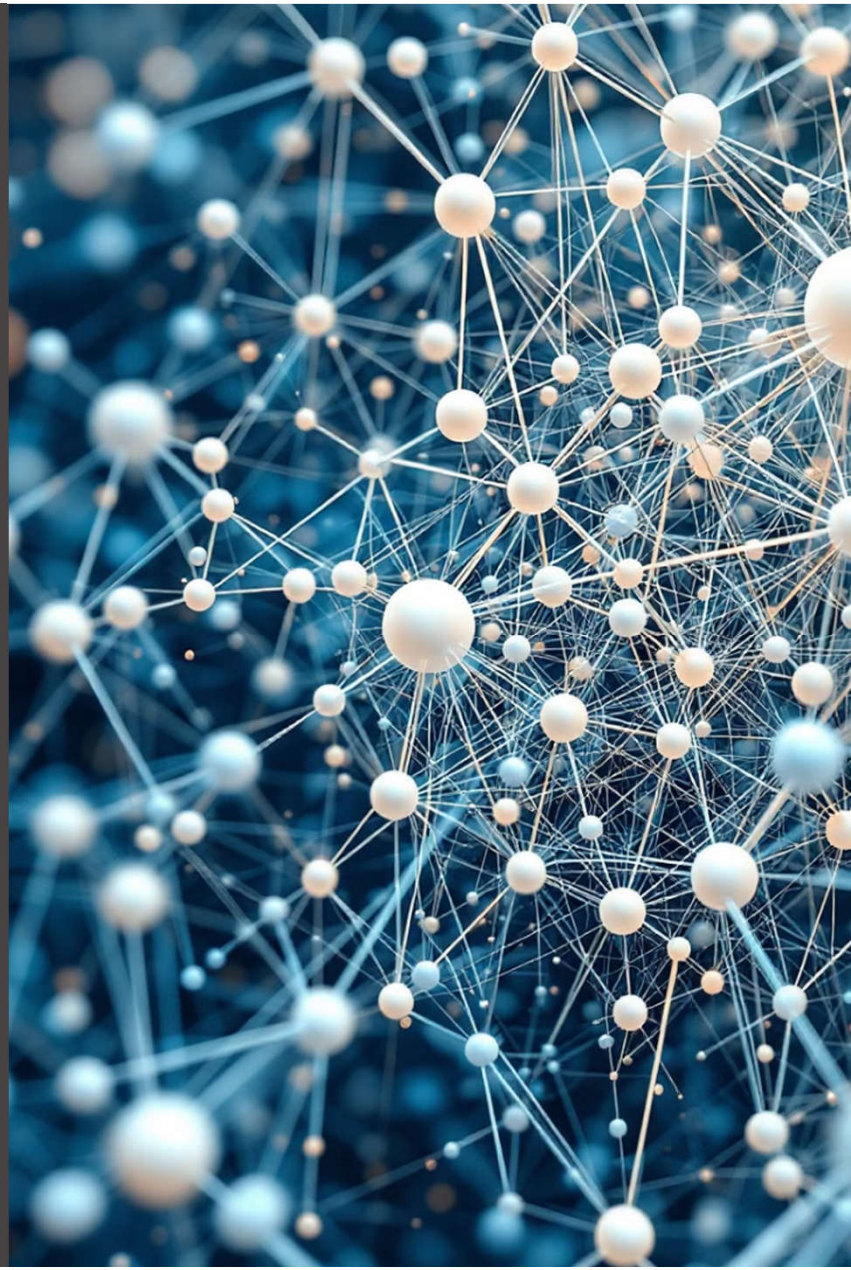
How They Work

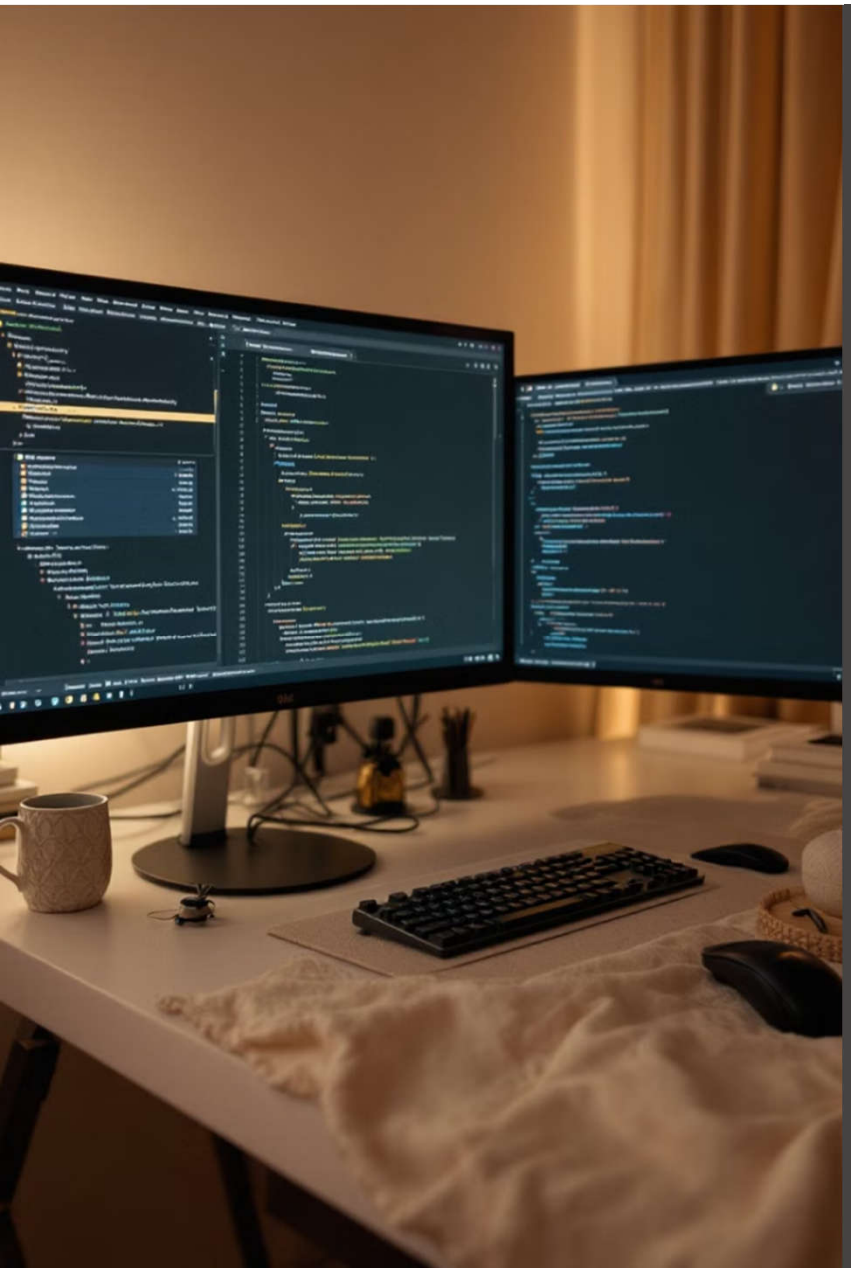
LLMs process input text through neural networks. They predict the most likely next words based on training data.

3

Common Use Cases

Text generation, summarization, translation, and question-answering are popular LLM applications.





Setting Up Development Environment

Install Python

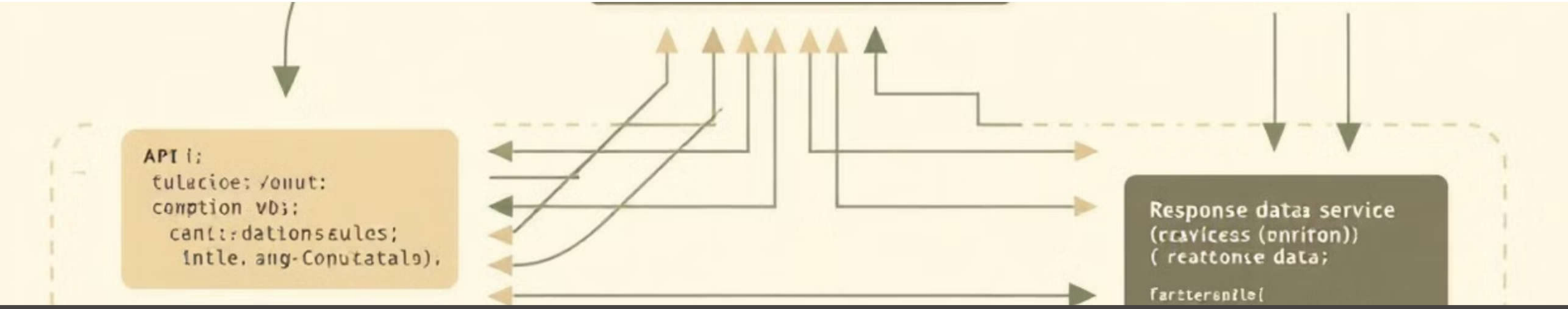
Download and install Python 3.8+ from python.org. Verify installation with 'python --version' in your terminal.

Virtual Environment

Create isolation with 'python -m venv venv'. Activate it with 'source venv/bin/activate' (Mac/Linux) or 'venv\Scripts\activate' (Windows).

Required Packages

Install dependencies with 'pip install openai streamlit python-dotenv ...'.



Working with LLM APIs

Choose an LLM Provider

Select GroqCloud as your LLM provider,

Register for API Access

Create an account with your chosen provider. Generate an API key from your account dashboard.

Make Your First API Call

Write Python code to send requests to the API. Process the responses to extract generated text.

Developing Web Scraping Techniques

Select Data Sources

Identify websites with relevant information for your application.

Set Up a Web Scraper with LangChain

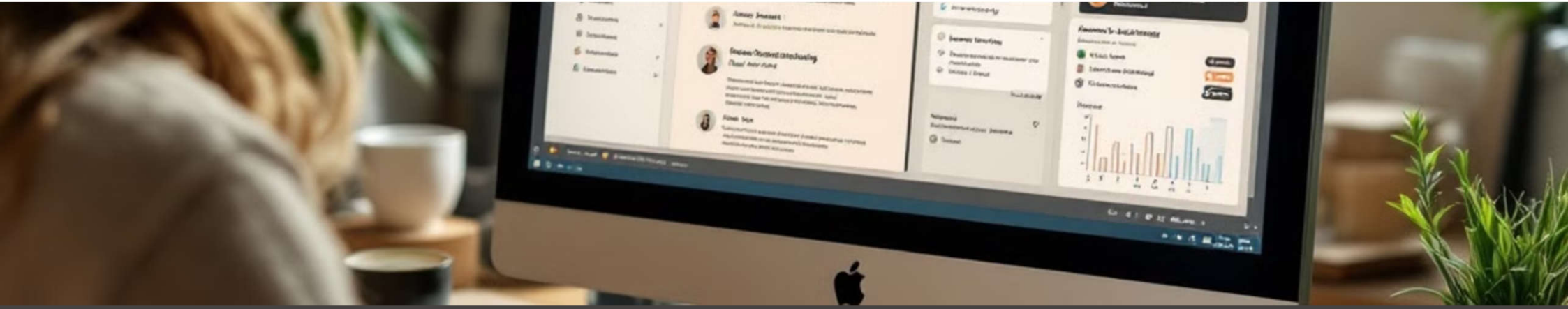
Use LangChain's **WebBaseLoader** or a custom scraper to fetch webpage content.

Clean and Process Data

Remove unnecessary elements and format data for LLM input.

Parse with BeautifulSoup

Extract specific elements like text, links, or tables.



Crafting Effective Prompts

Define the Prompt Structure

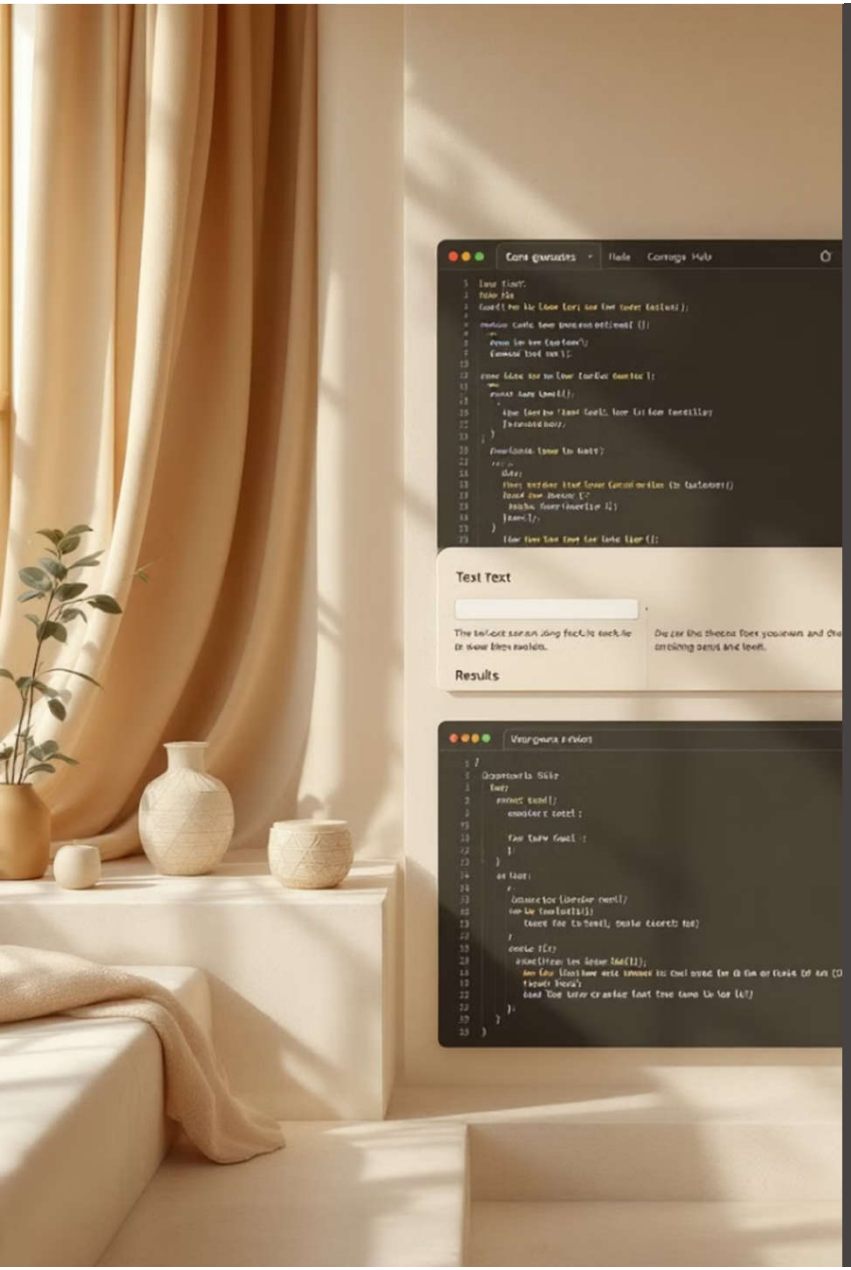
Use LangChain's PromptTemplate to create a reusable prompt structure.

Customize the Prompt for Your Use Case

Tailor the prompt to your application's needs (e.g., summarization, question-answering, or data extraction).

Integrate the Prompt with Your Application

Pass user inputs or scraped data into the PromptTemplate to generate dynamic prompts.



Building with Streamlit

1

Create App Structure

Write a main.py file with basic Streamlit commands. Import required libraries at the top.

2

Design User Interface

Add input widgets like text areas and buttons.

3

Connect LLM Functionality

Integrate API calls with user inputs. Display generated results in an organized layout.

4

Test Locally

Run 'streamlit run main.py'. Verify all features work correctly before deployment.

Securing API Keys

1 Never Hardcode Keys

Avoid placing API keys directly in your code. This creates security risks if code is shared.

2 Use Environment Variables

Store sensitive information in a .env file. This file should never be committed to version control.

3 Load with python-dotenv

Import and use the dotenv library. Access keys safely with os.getenv() in your application.

4 Configure Deployment Security

Set environment variables in your hosting platform. Many services offer secure storage options.

Deploying Your Application

1

Choose Platform

Streamlit Cloud, Heroku, or Railway offer simple deployment options for Python web apps.

2

Prepare Files

Create requirements.txt and README files. Ensure .gitignore excludes sensitive information.

3

Deploy

Connect your repository to your chosen platform. Configure environment variables for API keys.

4

Share

Get a public URL for your application. Share it with friends or professional contacts.