ChatGPT

Comprehensive Resources for Next.js + FastAPI + LangChain Agent Stack

Tutorials and Courses

- **Full-Stack Blog Guides:** Step-by-step guides are available in developer blogs. For example, a LangGraph/FastAPI/Next.js agent tutorial on Dev.to walks through building "Post Generator" and "Stack Analyzer" agents with Next.js UI and FastAPI backend [1] [2]. Similarly, a Medium walkthrough shows how to scaffold a Next.js+TypeScript app (`create-next-app`) and add a FastAPI Python backend, including Vercel deployment notes (e.g. watch for Vercel's 100 MB limit on Python packages) [3] [4]. Tech blogs also cover integrating Python ML code into Next.js. FreeCodeCamp/YouTube courses on Next.js or FastAPI (though not agent-specific) can help fill gaps. In addition, **LangChain Academy** offers free self-paced courses (e.g. "Deep Agents with LangGraph" or "Ambient Agents") that teach advanced agent workflows with code labs [5].

- **Official LangChain Tutorials:** The LangChain documentation includes tutorials and "how-to" guides on chains, chatbots, RAG, and agents. Notably, the *Tutorials* section lists guides to build chatbots with memory and agents calling tools [6]. Key topics like memory, tools, and multi-step retrieval are covered in conceptual guides and examples. LangChain's **Academy** and the official **LangSmith** docs also provide structured learning paths on agent observability and deployment. For background on LangChain fundamentals (chains, prompts, memory, etc.), introductory tutorials and conceptual guides are listed on the LangChain docs site [7] [6].

Documentation and Official Guides

- **Next.js:** The official Next.js docs describe Next.js as a React framework for building *full-stack* web apps [8]. It includes guides on TypeScript, API routes, and deployment (including Vercel). The docs have sections on "Getting Started" and various guides (Pages vs App Router, routing, data fetching). The **Next.js Docs** explicitly invite community support via GitHub Discussions, Discord, Twitter, and Reddit [9]. Refer to the Next.js docs for best practices (e.g. API routes for BFF, `next.config.js` rewrites) and tutorials on SSR/SSG, React hooks, etc.

- **FastAPI:** FastAPI's official documentation calls it "a modern, fast (high-performance) web framework for building APIs with Python" [10]. It has an extensive user guide with examples on routers, dependencies, OAuth2, error handling, WebSockets, and streaming responses (e.g. Server-Sent Events). The **"Full Stack FastAPI Template"** docs outline a sample stack (FastAPI + React/TypeScript + Docker + Postgres + Auth) [11]. The FastAPI docs also highlight community resources: an official **Discord** chat for discussion and **GitHub Discussions** for Q&A [12]. Use the docs for reference on Pydantic models, async/await use, and built-in OpenAPI support.

- **LangChain and LangGraph:** The LangChain docs site provides a comprehensive reference for chains, agents, tools, and memory. It includes **conceptual guides** and how-to's for LLM apps (e.g. a RAG tutorial, SQL agents) [6]. LangGraph (LangChain's agent/memory framework) is covered under "Orchestration" tutorials (e.g. building agents with memory or tools). The docs list include "Agents" and "Chatbots" tutorials. Also check the LangChain **forum** and the Slack community (see Community section below) for clarifications.

Example Projects and Code Repositories

- **Next.js + FastAPI Starter:** The "Next.js + FastAPI: Chat with Your Website" repo is a template combining these technologies with LangChain tools [13]. It integrates a FastAPI backend as a Next.js API route (using `next.config.js` rewrites) so that locally FastAPI runs on `localhost:8000` and in production on Vercel serverless [13]. This repo demonstrates fetching and processing website content via LangChain and streaming answers back to the Next.js front-end.

- **CopilotKit/LangGraph Demos:** CopilotKit's example *open-gemini-canvas* repo shows a production-like stack: Next.js frontend, FastAPI backend, and LangGraph-based agents powered by Google Gemini/LLMs [2]. It includes two agents (a post-generator and a stack-analyzer) and uses environment variables for API keys. This project illustrates a clean separation of Next.js (UI) and FastAPI (agents) [2].

- **Assistant-UI + LangGraph:** The `assistant-ui-langgraph-fastapi` repo uses LangGraph with a modern React-based chat UI. It runs a LangGraph agent on FastAPI, streaming responses via **assistant-stream**, and uses `assistant-ui` in Next.js for a sleek chat interface [14]. Features include real-time streaming to the browser and integration of external tools/APIs. Studying this code shows how to set up a streaming SSE endpoint in FastAPI and hook it into a Next.js front-end [14].

- **AgentKit (BCG X):** AgentKit is an open-source LangChain-based starter kit for scalable agent apps. It uses Next.js 14 (with Tailwind, TypeScript, NextAuth) and FastAPI (with SQLModel, Pydantic 2.x) in a Docker-compose setup [15]. AgentKit provides a chat UI with streaming, table/code rendering, and includes readiness for auth, Redis/Celery queues, and monitoring [16] [15]. It exemplifies a production-ready architecture (with container deployment and tests) for LLM agents.

- **Streaming Demo (LangChain + FastAPI):** The `LangChain-FastAPI-Streaming` repo by Coding-Crashkurse demonstrates streaming chat with GPT-3.5 and FastAPI [17]. It shows how to use LangChain's async `astream` interface with FastAPI's `StreamingResponse` to send chunked LLM outputs to a browser in real time [17]. This is useful for learning how to implement SSE or chunked responses for live chat.

- **Full RAG Example (React + FastAPI):** The "LangChain-RAG-Pattern" demo by jonathanscholtes is a complete project using a React frontend and FastAPI backend with an Azure Cosmos DB vector store [18]. It walks through indexing data, spinning up FastAPI endpoints for retrieval, and building a React UI to ask grounded questions. This shows how to structure a Retrieval-Augmented Generation app across the full stack [18].

- **Other Boilerplates:** There are also boilerplate templates (e.g. Next.js + FastAPI + LangChain CLI) and community starter projects. For example, a Next.js + FastAPI "enterprise" template was extended with LangChain by UF-Hobi (see their GitHub). Searching GitHub for "FastAPI LangChain" or "Next.js LangChain" yields community projects and demo apps to study.

Video Tutorials and Conference Talks

- **YouTube Channels:** Many developers share free tutorials. Look for "LangChain agents" or "FastAPI streaming" on YouTube. Channels by CodewithAI, Dave Gray, or AI-focused educators often cover Next.js + LangChain apps. For example, Dave Gray has a video "Build an AI RAG App with LangChain & Next.js" (covering chat UI and vector DB). FreeCodeCamp or TechWithTim may have full-stack FastAPI/Next.js walkthroughs (though maybe without LangChain specifics).

- **Streaming and Integration Walkthroughs:** There are video series on LangChain that include streaming to the web (e.g. "LangChain JS Tutorial #3: Response Streaming with Next.js 13"). Likewise, look for tutorials on "FastAPI LangChain streaming" – some creators demonstrate SSE with LangChain in FastAPI.

- **LangChain Conferences:** LangChain's *Interrupt AI Agent Conference* (May 2025) has many recorded sessions. For instance, the LangChain website lists the 2025 recordings: it includes a "LangChain Keynote", "LangChain: Building Reliable Agents", an "Andrew Ng: State of Agents" talk, and other industry sessions [19]. These sessions (available on the LangChain site and YouTube) cover agent design, LangGraph, and production-scale deployment patterns. Watching these can give high-level insights into building robust agentic systems [19].

Books and Articles

- **"Learning LangChain" (O'Reilly):** This comprehensive book covers everything from LangChain basics to advanced agents. It includes chapters on building RAG pipelines, adding memory with LangGraph, and deploying agents [20]. It explicitly promises a step-by-step guide to "build a production-ready AI agent" using LangChain [20]. It's a recommended in-depth reference for LangChain users.

- **Agent Development Books:** Other recommended reads include *"Hands-On GenAI Agent Development with LangChain"* and *"AI Agents and Applications"* (Manning), which delve into multi-agent architectures and tooling. These aren't free, but they cover theoretical and practical aspects of agents.

- **Tutorial Articles:** Many blogs and Medium posts cover aspects of this stack. Besides the ones cited above, look for articles on *building a chatbot with LangChain*, *LLM retrieval systems*, and *Next.js + Python APIs*. For example, dev.to, Medium, and Hacker Noon often have case studies (like converting LLM outputs to streaming JSON, or using conversational memory). Curating bookmarks from these can help fill knowledge gaps.

Community and Support

- **LangChain Community:** Join the official LangChain Slack and forum for help and discussion. LangChain's website invites users to their community Slack (for design discussions) and points to the LangChain Forum for product support [21]. The Slack is active with channels on agents, memory, and integrations; the forum is good for tracking issues and examples. Also check StackOverflow (tag [langchain]) for Q&A.

- **FastAPI Community:** FastAPI has an active Discord server (invite link on the docs site) for chat [12]. Questions are often answered in GitHub Discussions (with FastAPI experts). The official "Help FastAPI" page links to their Discord and advises using GitHub Discussions for technical questions [12].

- **Next.js Community:** Next.js maintainers mention that help is available via GitHub Discussions, Discord, Twitter, and Reddit [9]. Subreddits like r/nextjs and r/FastAPI are good places to ask questions or find shared projects. For example, a post on r/FastAPI shares a starter project combining Next.js and LangChain [22]. Likewise, r/LangChain (if existing) or general ML/AI forums on Reddit can offer advice.

- **Other Resources:** Consider joining relevant Discord/Slack groups for React, Python, and LLM enthusiasts. Community-contributed templates (e.g. on GitHub) often have useful discussions in their issue trackers. Engaging with communities (StackOverflow, GitHub Discussions, r/AI or r/MachineLearning) can help solve specific integration issues (e.g. state management or deployment quirks).

Key Use-Case References

- **Conversational Agents with Memory:** LangChain's chatbot tutorial (using LangGraph) teaches adding conversation memory to an assistant [6]. The LangChain Academy course *"Deep Agents with LangGraph"* specifically focuses on long-term agent memory [5]. Studying these shows how to store conversation history (via BaseChatMessageHistory or StateGraph) and use it to maintain context.

- **Agents Calling APIs/DBs:** For agents that invoke external tools or APIs, see the LangChain "Build an Agent" tutorial (agents with tool functions) [6]. The AgentKit repo includes a library of common API tools ready to plug in [15]. The `assistant-ui-langgraph-fastapi` example also hints at integrating custom tools (it mentions "integrate external tools and APIs" [14]). Practically, you'd write LangChain tools (Python functions or API wrappers) and attach them to agents.

- **Real-Time Streaming Chat:** Use FastAPI's async streaming (via SSE or WebSockets) to push tokens to the frontend. The dev.to article shows using `StreamingResponse` and LangChain's `astream_log` for SSE [23]. The Crashkurse demo likewise streams chunks from OpenAI's GPT to the browser [17]. On the Next.js side, use libraries or custom logic to consume events. (Libraries like `EventSource` in the browser or packages like `assistant-stream` can help.)

- **State and Conversation History:** LangChain provides built-in memory classes (buffer, summary, vector). The best practices guide suggests using `ConversationBufferMemory` or newer

LangGraph memory for short-term context [6]. For longer context, combine RAG: store past interactions in vector DB and retrieve as context. The books above cover merging memory chains. In code, you might implement conversation state management in your FastAPI server (store history per user session or token). The `assistant-ui-langgraph-fastapi` project illustrates maintaining state across streams [14].

- **Deployment and Production:** For production readiness, plan for auth, error handling, and scaling. AgentKit is a prime example: it includes NextAuth integration, testing, and Docker-compose deployment [15]. If using Vercel/Render, remember their serverless limits (e.g. Python package size [4]). In practice, one may deploy the Next.js frontend on Vercel (or Render/Netlify) and run FastAPI separately (e.g. on Render or AWS with Docker). For containerization, use `uvicorn`/Gunicorn for FastAPI, and Dockerize both frontend and backend (AgentKit's demos can guide this) [15]. Also include logging/monitoring – for example, LangSmith or other observability tools (see LangChain docs on evaluation).

Each of these resources – from official docs to community repos – provides pieces of the puzzle. By combining LangChain's tutorials and courses [6] [5] with production-ready examples (AgentKit, CopilotKit, etc.), and leveraging community support (forums, Discord), you can learn to build and deploy a robust Next.js/TypeScript + FastAPI + LangChain AI agent system that meets your use cases.

Sources: Official docs and tutorials (Next.js [8] [9], FastAPI [10] [12], LangChain [7] [6]), example projects on GitHub [13] [2] [14] [16] [17] [18], and community-shared guides [1] [23] [4] [20] [19]. Each citation points to material covering the described topic.

---

[1] Here's How To Build Fullstack Agent Apps (Gemini, CopilotKit & LangGraph) - DEV Community
https://dev.to/copilotkit/heres-how-to-build-fullstack-agent-apps-gemini-copilotkit-langgraph-15jb

[2] GitHub - CopilotKit/open-gemini-canvas
https://github.com/CopilotKit/open-gemini-canvas

[3] [4] Boosting Your Full-Stack Workflow with Next.js, FastAPI and Vercel | by Kaveh Pouran Yousef, PhD | Medium
https://medium.com/@kaweyo_41978/boosting-your-full-stack-workflow-with-next-js-and-fastapi-and-vercel-3c7d3cd8220f

[5] LangChain Academy
https://academy.langchain.com/

[6] [7] Tutorials | LangChain
https://python.langchain.com/docs/tutorials/

[8] [9] Next.js Docs | Next.js
https://nextjs.org/docs

[10] FastAPI
https://fastapi.tiangolo.com/

[11] Full Stack FastAPI Template - FastAPI
https://fastapi.tiangolo.com/project-generation/

[12] Help FastAPI - Get Help - FastAPI
https://fastapi.tiangolo.com/help-fastapi/

[13] GitHub - mazzasaverio/nextjs-fastapi-your-chat: Chat with any website using FastAPI, Next.js, and the latest LangChain version for seamless integration.
https://github.com/mazzasaverio/nextjs-fastapi-your-chat

[14] GitHub - Yonom/assistant-ui-langgraph-fastapi
https://github.com/Yonom/assistant-ui-langgraph-fastapi

[15] [16] GitHub - BCG-X-Official/agentkit: Starter-kit to build constrained agents with Nextjs, FastAPI and Langchain
https://github.com/BCG-X-Official/agentkit

[17] GitHub - Coding-Crashkurse/LangChain-FastAPI-Streaming
https://github.com/Coding-Crashkurse/LangChain-FastAPI-Streaming

[18] GitHub - jonathanscholtes/LangChain-RAG-Pattern-with-React-FastAPI-and-Cosmos-DB-Vector-Store: Complete project (web, api, data) covering the implementation of the RAG (Retrieval Augmented Generation) pattern using Azure Cosmos DB for MongoDB vCore and LangChain. The RAG pattern combines leverages the new vector search capabilities for Azure Cosmos DB.
https://github.com/jonathanscholtes/LangChain-RAG-Pattern-with-React-FastAPI-and-Cosmos-DB-Vector-Store

[19] Interrupt - The AI Agent Conference by LangChain: 2025 Recordings
https://interrupt.langchain.com/video

[20] Learning LangChain [Book]
https://www.oreilly.com/library/view/learning-langchain/9781098167271/

[21] Join the Community
https://www.langchain.com/join-community

[22] Chat with a website using Next.js, FastAPI and LangChain : r/FastAPI
https://www.reddit.com/r/FastAPI/comments/1ahatcz/chat_with_a_website_using_nextjs_fastapi_and/

[23] Integrating LangChain with FastAPI for Asynchronous Streaming - DEV Community
https://dev.to/louis-sanna/integrating-langchain-with-fastapi-for-asynchronous-streaming-5d0o