Providing On-Prem GenAI Inference Services to a Campus Community

Sarah Rodenbeck
Rosen Center for Advanced Computing
Purdue University
West Lafayette, IN, USA
snrodenbeck@gmail.com

Erik Gough
Rosen Center for Advanced Computing
Purdue University
West Lafayette, IN, USA
goughes@purdue.edu

Athreyan Mohana Krishnan Sangeetha
Rosen Center for Advanced Computing
Purdue University
West Lafayette, IN, USA
amohanak@purdue.edu

Ashish
Rosen Center for Advanced Computing
Purdue University
West Lafayette, IN, USA
ashish@purdue.edu

Mihir Ahlawat
Rosen Center for Advanced Computing
Purdue University
West Lafayette, IN, USA
mahlawat@purdue.edu

Vivek Karunai Kiri Ragavan
Rosen Center for Advanced Computing
Purdue University
West Lafayette, IN, USA
vkarunai@purdue.edu

Abhishek Muthukumar
Rosen Center for Advanced Computing
Purdue University
West Lafayette, IN, USA
muthukua@purdue.edu

Aanis Ahmad
Rosen Center for Advanced Computing
Purdue University
West Lafayette, IN, USA
aahmad31@purdue.edu

Abstract

The Rosen Center for Advanced Computing at Purdue University has recently released two Generative AI inference tools, AnvilGPT and Purdue GenAI Studio, to the research and campus communities. These services support over 1000 users who use 10+ open-source GenAI models to aid their work. Building on HPC's long history of using open-source tools, these services are based on customized open-source frameworks and hosted entirely on-prem. This paper argues that building custom GenAI services from open-source frameworks is a scalable and cost-effective solution for providing access to Generative AI models. This paper shares the methodology and resources required to develop and host these services and seeks to be a resource for other research computing centers that wish to leverage their HPC investment to create similar services.

CCS Concepts

• **Computer systems organization** → *Cloud computing*; • **Computing methodologies** → **Natural language generation**; • **Information systems** → **Language models**; • **Software and its engineering** → *Designing software.*

PEARC '25, Columbus, OH, USA

ACM ISBN 979-8-4007-1398-9/25/07
https://doi.org/10.1145/3708035.3736039

Keywords

Generative AI Inference Services

ACM Reference Format:
Sarah Rodenbeck, Erik Gough, Athreyan Mohana Krishnan Sangeetha, Ashish, Mihir Ahlawat, Vivek Karunai Kiri Ragavan, Abhishek Muthukumar, and Aanis Ahmad. 2025. Providing On-Prem GenAI Inference Services to a Campus Community. In *Practice and Experience in Advanced Research Computing (PEARC '25), July 20–24, 2025, Columbus, OH, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3708035.3736039

1 Introduction

Campus high-performance computing (HPC) centers must continuously look for ways to provide reliable, secure, and cost-effective computing services to meet the scientific computing needs of researchers and students. Over the past 20 years, a primary and critical service has been providing access to batch HPC systems for running large-scale computation and simulation [8]. More recently, centers have provided cloud-native scientific computing platforms for hosting science gateways and persistent services [4]. These resources are made possible by the well-established practice of utilizing a combination of commodity hardware, open-source tools, and staff expertise.

With the increasing maturity of Generative AI (GenAI), campus researchers, students, and staff increasingly seek to incorporate AI tools into their scientific workflows. At Purdue's Rosen Center for Advanced Computing (RCAC), there had been a surge in requests for help integrating GenAI into research pipelines. Many users asked for help running specific open-source models, resulting in unsustainable bespoke solutions, given the required requests and

resources. This necessitated a centralized approach, for which there were two fundamental options: build or buy.

Within a campus HPC environment, the build option is compelling. Commercial services charge usage- or seat-based fees associated with covering infrastructure costs—infrastructure HPC centers often already have. The deployment and management of HPC systems on campus have been consistently proven to be less expensive than paying for services from public cloud providers [14]. Building an on-prem platform allows universities to utilize their infrastructure investment, often at a fraction of the cost of going through a commercial service.

For these reasons, Purdue built AnvilGPT and Purdue GenAI Studio as LLM platforms for the national Anvil and Purdue campus communities, respectively. These on-prem services utilize open-source tools, run on approximately $100,000 (total) in hardware, and are provided at no cost to researchers, in contrast to the multi-million dollar deals other university systems have made with commercial platforms [15].

In the following sections, we will expand upon the requirements and customizations involved in building these services, emphasizing the minimum requirements to get started and recommendations for scaling.

1.1 User Requirements

The development of Purdue's GenAI services has been guided by the needs of the campus and Anvil user bases. Use cases fall into three general categories:

- Chat User: Most users use GenAI for one-off tasks, like helping write emails and debugging code. The only customization for these users is prompt engineering.
- RAG User: Users wishing to use retrieval-augmented generation (RAG) to supplement model knowledge in a particular domain form the second most common use case [3]. These users typically create RAG models based on research papers or other domain-specific documentation.
- API User: The most advanced use case involves integrating GenAI into larger workflows. These users typically use the API and may also be interested in either fine-tuning a foundational model or developing agentic workflows. This functionality is often used for integration into science gateways, like Nanohub, as the backend of a custom chatbot [9].

Beyond these user profiles, additional overarching requirements from the user base included:

- Data Privacy: Building a solution inherently allows for much more control over data privacy, a concern we observed being raised frequently by the community about commercial services. Keeping user data private to users enables the tool be used with proprietary data.
- Variety of models: Based on the specific use case, users often need access to different open-source models, so services must support using many different models.

1.2 Usage

AnvilGPT and Purdue GenAI Studio were released in the fall of 2024 and now have a combined user base of over 1000, with about 40% active every month. Each supports about 15 models, ranging from DeepSeek-r1:1.5B to llama3.3:70B. Users show a propensity towards the various 70B models, with these models accounting for 35-40% of usage between the two services, even as the default has been set to smaller models (Figure 1).

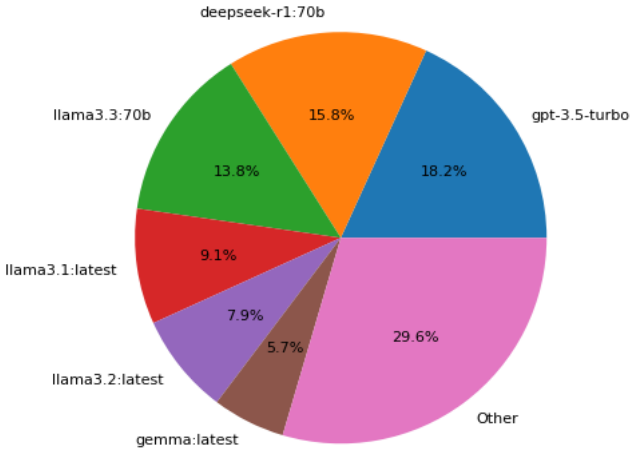

Figure 1: Model Usage

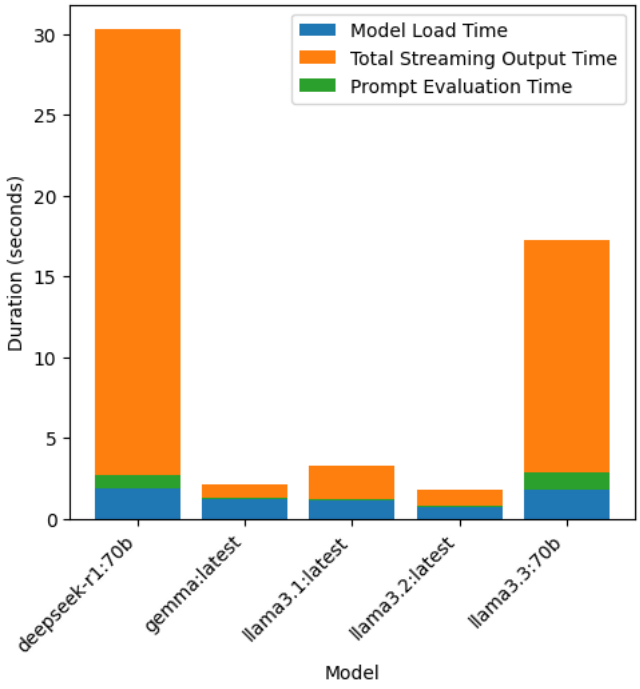

Figure 2: Performance Across AnvilGPT and GenAI Studio

2 Architecture

AnvilGPT and Purdue GenAI Studio are largely based on three open-source projects: Ollama, Open WebUI, and Postgres [11, 17].

2.1 User Interface

Open WebUI is designed as a self-hosted, web-based user interface that enables interaction with GenAI models. It forms the foundation of the UI and API capabilities and supports functionalities like user management. Open WebUI offers a modular architecture that integrates natively with Ollama, allowing users to select and switch between different hosted models. Users can also point Open WebUI to the OpenAI API for models to which they have paid access. Additionally, Open WebUI supports RAG and has database and vector store integration for data management [17].

Open WebUI has an active development community and many of the initial limitations we observed have since been fixed. Although we have chosen to build additional customizations to better support our user base and provide quality-of-life improvements, only a few of these should be viewed as required for rolling out a GenAI service based on Open WebUI to a campus community.

2.2 Model Serving

To process model inference requests, trained models must be deployed. Important characteristics of model serving are 1) latency - the time it takes for a response to be processed; 2) throughput - the number of tokens provided; and 3) flexibility - the number of models that can be hosted.

Many tools have been created to serve LLMs including vLLM, TensorRT-LLM, and Ollama [10, 11, 16]. We evaluated Ollama and vLLM to serve models and ultimately selected Ollama due to our user requirements. While vLLM has higher throughput (4-5x over Ollama, based on our testing), it offers less flexibility for serving multiple models. In contrast, Ollama is optimized for dynamic models loading on a single GPU.

2.3 Hardware Requirements

Hardware is required to host the models, database, and UI, but by far, most resources are required for Ollama/model serving. For model hosting on AnvilGPT and GenAI Studio, we focus on 4-bit quantized models to make the most efficient use of GPU memory. The minimum GPU requirement for a model-serving backend for hosting 4-bit quantized 70B parameter models is 40G vRAM, which either NVIDIA A100 or L40 GPUs can provide. AnvilGPT and GenAI Studio backends are each served by two NVIDIA GPUs (H100 or GH200) with 96GB of vRAM. To provide fast model load times, models are stored on SSDs. This hardware is sufficient to simultaneously serve both 70B parameter and smaller models. Figure 2 demonstrates average response times with this setup.

2.4 Load Balancing

To accommodate large numbers of inference requests, load balancing can be used to route queries to multiple model-serving backends. Load balancing can be implemented via simple DNS load balancing or a reverse proxy. However, most inference engines are readily available as container images that can be launched with container runtimes like Docker. Kubernetes is a natural fit for orchestrating these containers but is not a requirement. Our services run across three Kubernetes deployments: a GPU-backed Ollama deployment, and two CPU deployments for the Open WebUI front-end and database.

3 Customizations

Customizations we have made to the open source framework broadly fall into two categories: (1) initial changes required to make the service suitable to release to a large campus community and (2) additional features and quality of life improvements. As Open WebUI is still actively developing, we have omitted customizations we initially made that have been subsequently added/fixed.

3.1 Initial Customizations

3.1.1 Database Changes. We started by moving the database from SQLite to PostgreSQL to increase the robustness and scalability of Open WebUI for a large user base. We also moved from ChromaDB to PGVector for RAG to simplify database provider management [12]. It supports storing embedded sentences as vectors and similarity search between vectors. CRUD operations using PGVector can be tricky, especially when deleting embeddings since there is a lot of abstraction. We achieve this by keeping track of the vector IDs when the document is first inserted into the RAG and mirroring the movement of the document across both the vectorstore and the relational database.

3.1.2 Single Sign-On Integration. Adding an SSO for access management was crucial to offering the service across the community. Open WebUI supports any authentication platform supporting the OpenID Connect (OIDC) protocol, making integrating CILogon for SSO possible [2, 5].

3.2 New Features and Changes

3.2.1 Document Collection Management. While Open WebUI supports RAG functionality, it did not support the ability to organize documents. As users may want to develop RAG models on different topics, we added the concept of collections to collate and address documents together. This involved minor changes to the database structure to track document-collection association and tweaking how the RAG was ingesting data to avoid leakage across collections. Users can change a document's collection, allowing for more dynamic groupings according to shifting needs.

3.2.2 Custom Models and Secure Sharing. Users can modify base models like llama to create private custom models by changing the knowledge, prompts, and other parameters. While this capability is built-in, we implemented functionality to allow users to share (with inference-only or full edit access) these custom models with other users via their registered email IDs. This can facilitate sharing across members of a research group and inference-only model access, for example, as a TA for a course.

3.2.3 Document Parsing. One issue we encountered during our implementation was parsing documents, particularly PDF files. The default PDF parser used by Open WebUI is PyPDF. We found that this loader struggles with parsing font-specific ligatures (such as ff, fi, fl, and others), which are interpreted as null bytes (`\x00`), which led to issues with context building. To address this, we switched to a more mature library called PyMuPDF, which is more versatile and offers better parsing speed [13].

3.2.4 Speech Model Integration. Open WebUI has some built-in integration for speech-to-text and text-to-speech capabilities through

the "conversation" feature where users can converse with a model, but these capabilities were not available as a standalone feature. Thus, based on requests from users to who wanted to work with audio files as input, we added a new tab in the workspace to allow users to interact with these models.

3.2.5 Image Generation Integration. Support for image generation features exists within Open WebUI but provides a sub-optimal user experience as it is primarily designed for text-based interactions. Image generation must be supported by an additional Kubernetes deployment, in this case, AUTOMATIC1111's (A1111) Stable Diffusion implementation [1]. To improve user experience, we added a separate page for image generation, based on the A1111 interface to provide users with much more control and levers for working with image generation models.

3.2.6 Metrics and Usage. Finally, while the database tracks some usage information, it is important to be able to communicate this to show value. Thus, while not a change to the app itself, we created custom workflows to track usage across categories, including chat/message metrics, model usage, performance, and users. Crucially, through this effort, we identified key changes that were required to the database to support this as, by default, deleted chats and messages sent via API are not retained by Open WebUI for statistics. Additionally, to quickly catch any downtime on a model or overall service level, we implemented several workflows to alert (via Slack) on the health status of the applications.

4 Discussion and Future Work

4.1 Future Roadmap

Running an on-premise GenAI service presents a new opportunity to engage and contribute to the open-source software ecosystem. The rapidly evolving nature of open-source LLM frameworks brings frequent releases with improved features and updates. In our experience with Open WebUI, new releases would deliver valuable enhancements that sometimes conflicted with or duplicated our customizations. Moving forward, we will take a more active involvement in Open WebUI development by submitting feature requests, bug reports, and code contributions as we look for new opportunities to contribute back to the project based on our experiences.

We intend for future development to continue to be driven by user needs and requests, and we have identified and are working on several key areas. Many users have asked us to integrate our GenAI services with other applications like learning management systems (e.g. Brightspace). Second, users currently need to work outside our platforms to perform fine-tuning and we aim to incorporate built-in capabilities and hosting options for these customized models. Finally, we are building more sophisticated pre- and post-processing tools, including enhanced web search and improved document parsing through Docling for RAG implementations [7].

4.2 Scalability

We are actively developing a scalable model-serving infrastructure that leverages both Ollama (for smaller, dynamic model loading) and vLLM (for high-demand, larger models). To streamline deployment and management we're integrating LLMariner, an open-source platform designed for generative AI workloads [6]. Built on Kubernetes, LLMariner enables efficient training and inference management while offering OpenAI-compatible APIs. Our work is focusing on optimizing model serving across different runtimes, efficiently routing requests from OpenWebUI, and implementing dynamic scaling based on inference thresholds to balance efficiency and scalability.

4.3 Discussion

While we have added hardware and custom features over time based on requests from our user base, a key distinction is that these are not required to get started. Instead, we propose a generalized GenAI service deployment strategy focused on a low barrier to entry and iterative improvement. Purchasing new, dedicated hardware immediately to deploy a GenAI pilot is unnecessary; rather it is sufficient to build on existing hardware. By attracting a growing user base, making the case for investing more in hardware and development efforts becomes easier. This "grow as you go" approach provides a scalable foundation for supporting the majority of GenAI use cases in a campus environment.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 2005632. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] Automatic1111. 2025. Automatic1111. https://github.com/AUTOMATIC1111/stable-diffusion-webui/wiki
[2] CILogon. 2025. CILogon. https://www.cilogon.org/
[3] Patrick Lewis et al. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 9459–9774. https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf
[4] Preston Smith et al. 2020. The "Geddes" Composable Platform - An Evolution of Community Clusters for a Composable World. In *2020 IEEE/ACM International Workshop on Interoperability of Supercomputing and Cloud Technologies (SuperCompCloud)*. 33–38.
[5] OpenID Foundation. 2023. https://openid.net/developers/how-connect-works/
[6] LLMariner. 2025. LLMariner. https://llmariner.ai/
[7] Kim Martineau. 2024. A new tool to unlock data from enterprise documents for generative AI. https://research.ibm.com/blog/docling-generative-AI
[8] Gerry McCartney, Thomas Hacker, and Baijin Yang. 2014. Empowering Faculty: A Campus Cyberinfrastructure Strategy for Research Communities. *Educause Review* (2014).
[9] Nanohub. 2025. Nanohub. https://nanohub.org
[10] Nvidia. 2025. Nvidia TensorRT-LLM. https://docs.nvidia.com/tensorrt-llm
[11] Ollama. 2025. Ollama. https://ollama.com
[12] PGVector. 2025. PGVector. https://github.com/pgvector/pgvector
[13] PyMuPDF. 2025. PyMuPDF, LLM and RAG. https://pymupdf.readthedocs.io/en/latest/rag.html
[14] Preston Smith, Stephen Lien Harrell, Alex Younts, and Xiao Zhu. 2019. Community Clusters or the Cloud: Continuing cost assessment of on-premises and cloud HPC in Higher Education. In *Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (learning)* (Chicago, IL, USA) *(PEARC '19, Article 97)*. Association for Computing Machinery, New York, NY, USA, 1–4.
[15] California State University. 2025. CSU Announces Landmark Initiative to Become Nation's First and Largest AI-Empowered University System. https://www.calstate.edu/csu-system/news/Pages/CSU-AI-Powered-Initiative.aspx
[16] vLLM. 2025. vLLM. https://docs.vllm.ai/en/latest/
[17] Open WebUI. 2025. Open WebUI. https://openwebui.com