ChatGPT

Personal Friday Night AI Hackathon Plan (Next 3 Quarters)

You will dedicate 4–6 hours every Friday night to a **Personal Friday Night AI Hackathon** – half of the time for learning and half for hands-on projects. This plan is organized into three quarterly segments (approximately 12 weeks each) with clear learning goals, project milestones, and a timeline. The focus is on **Generative AI (GenAI)** – leveraging free, open-source courses and materials (like the Hugging Face Transformers course and Stanford's CS25 seminar) and building projects that either provide functional real-world value or demonstrate theoretical foundations (e.g. fine-tuning a foundation model). Throughout the journey, you will **share weekly progress on LinkedIn** to document your learning and showcase your growth to peers and future employers. Below is the comprehensive quarter-by-quarter plan:

- **Time Commitment:** ~4–6 hours every Friday (or equivalent time over the week if needed), split roughly 50/50 between **learning** (coursework, videos, reading) and **building** (coding projects).
- **Resources Used:** Free open-source materials, including the Hugging Face course on NLP/LLMs, Stanford CS25: *Transformers United* seminar talks, and various open-source libraries (Hugging Face Transformers, Datasets, etc.).
- **Project Portfolio:** You will create at least one project per quarter (hosted on GitHub and possibly Hugging Face Hub/Spaces) that demonstrates the skills learned. Each project will be well-documented and shared publicly.
- **Public Progress Sharing:** After each Friday session (or weekly), write a brief LinkedIn update highlighting what you learned or built that week. Include any interesting insights, screenshots or demos, and links to your GitHub or Hugging Face project. This consistent sharing will build an audience over time and signal your continuous learning to your network.

With these principles in mind, let's break down the plan for each quarter:

Quarter 1: Foundation and Hugging Face Mastery

Objective: Refresh and strengthen your foundation in transformers, NLP, and LLMs, while becoming proficient with the Hugging Face ecosystem. By the end of Q1, you will have completed the core Hugging Face *LLM/Transformers Course* (covering how transformers work, using pre-trained models, fine-tuning, etc.) and built a **mini NLP project** (e.g. a text classification or summarization demo) to solidify your knowledge. This sets the stage for more advanced projects later.

Learning Focus: Dive into the Hugging Face course on transformers/LLMs (which is free and open-source). This course will teach you both the **concepts** and the **practical skills** needed to work with transformers using Hugging Face libraries [1]. It starts from the basics of transformer models and progresses through using pre-trained models, fine-tuning them on datasets, and even how to share your models and build

demos [1] [2]. Given your strong Python/ML background, this will serve as a refresher and an update on the latest LLM workflows:

- *Hugging Face Course Chapters 1–4:* Fundamentals of transformers and how to use Hugging Face Hub models. By the end of chapter 4, you'll understand how Transformer models work and even know how to fine-tune a model on a dataset and share results on the Hub [3]. (Expect to perform a **simple fine-tuning** exercise here, such as fine-tuning a small model for sentiment analysis or text classification.)
- *Chapters 5–8:* Learn the Hugging Face **Datasets** and **Tokenizers** libraries and tackle classic NLP tasks. By completing these, you'll be able to handle common language processing tasks independently [4] (e.g. tokenization, dataset preparation, and training models for tasks like named entity recognition or translation).
- *Chapter 9:* Learn to **build and share demos** of your models on Hugging Face Hub (e.g. using Gradio for a web UI) [2]. This will allow you to showcase your work to the world by the end of the quarter. *(The Hugging Face course emphasizes sharing your work – by Chapter 9 you'll deploy an interactive demo on Hugging Face Spaces [5].)*

Project Focus: As you progress through the course, apply the lessons to a hands-on project. A suitable project for this quarter is a **simple NLP application** that uses a pre-trained transformer model and possibly a fine-tuned model that you train during the course. For example, you could build a **text summarizer or FAQ chatbot** for a small set of documents, or a **sentiment analyzer** for a specific type of text. The scope should be manageable (something you can develop and polish within a few weeks). The key outcomes of the project are:

- A trained or fine-tuned model (even if it's a small one like DistilBERT or GPT-2) saved to your GitHub/Hugging Face Hub.
- An interactive demo (e.g. Gradio app on Hugging Face Spaces) that allows anyone to input text and get the model's output (classification, summary, etc.).
- A clear README or documentation explaining the project's purpose, the data used, and how the model was fine-tuned or implemented.

This project will both reinforce your learning and populate your GitHub with a concrete example of your GenAI skills. Importantly, it's an opportunity to demonstrate end-to-end ability: picking a model, preparing data, fine-tuning, and deploying a small web app.

Tentative Timeline (Quarter 1 Weeks 1–12):

1. **Weeks 1–2: Setup and Course Kickoff** – Set up your environment (Python, PyTorch/TensorFlow as needed, install Hugging Face `transformers` and other libs). Start the Hugging Face course with **Chapter 1 & 2**, which introduce transformer models and what they can do. Refresh fundamental concepts like the transformer architecture and attention mechanism as needed. Use this time to also plan your quarter: decide on a mini project idea (e.g. "summarize news articles") that aligns with course content. *Share on LinkedIn:* announce your Personal Friday AI Hackathon initiative, outline your goals for Q1, and mention you're starting the HF course (this post sets context for your network).

2. **Weeks 3–4: Transformer Basics and First Fine-Tune** – Complete Chapters 3–4 of the HF course, diving into how transformers work and how to fine-tune a pretrained model on a dataset [3]. As a

hands-on exercise, fine-tune a small language model on a simple task. For example, you might use a dataset from Hugging Face Hub (like IMDb reviews for sentiment, or a Q&A dataset) and fine-tune a model (e.g. DistilBERT or a small GPT-2) for that task. The course will guide you through using the *Trainer* API or equivalent to do this. By the end of Week 4, aim to have one **fine-tuned model artifact** and upload it to the Hugging Face Hub (or your GitHub) with a model card. *LinkedIn update:* share what task you fine-tuned a model for and any initial results or challenges (this shows you're gaining practical skill).

3. **Weeks 5–6: Datasets, Tokenizers, and NLP Tasks** – Cover Chapters 5–6, learning how to use the `datasets` library to manage and preprocess data, and `tokenizers` to handle text tokenization efficiently. Practice by cleaning/processing the data for your chosen project. If your project is a summarizer, for example, prepare a small corpus of texts and their summaries; if it's a classifier, get labeled data. Proceed to Chapter 7 (Classical NLP tasks) and Chapter 8 if time allows, to apply transformers to tasks like NER, translation, etc., which solidifies your understanding of applying models to different NLP tasks [4]. *Project tie-in:* incorporate what you learn into improving your project's data pipeline (e.g., use the Hugging Face Datasets library to load and split your dataset). *LinkedIn update:* discuss one of the tools you learned (datasets or tokenizers) and how you're using it in your project.

4. **Weeks 7–8: Build the Mini Project** – By now, you have the knowledge to build your Quarter 1 project. Spend these weeks coding the project: load your fine-tuned model (or a pre-trained model if fine-tuning wasn't feasible for your task), integrate it into a simple application. If it's a web demo, create a Gradio interface (the Hugging Face course's Chapter 9 is all about this). For example, build a Gradio app that takes user input (a paragraph of text) and outputs a summary or classification using your model. Test the app locally. Also finish Chapter 8 (and 9 preview) in this period, to ensure you know how to deploy the demo. *LinkedIn update:* share a mid-project progress, maybe a screenshot of your app in development or a snippet of model output that looks interesting.

5. **Weeks 9–10: Deploy and Share the Demo** – Complete Chapter 9 of the course, which covers sharing demos on the Hugging Face Hub [2] (using Spaces). Deploy your project's app to **Hugging Face Spaces** (free hosting for ML demos) or an alternative like Streamlit community cloud, so that others (recruiters, colleagues) can interact with it. Ensure your GitHub repository for the project is up to date with all code, and add a thorough README describing the project, how to use it, and what you learned. This is also a good time to do code cleanup and add comments for clarity. *LinkedIn update:* proudly announce the **completion of your first project** – include the link to the live demo and the GitHub repo. Highlight what the tool does and invite feedback. This post can significantly showcase your hands-on achievement.

6. **Weeks 11–12: Buffer and Retrospective** – Use the final two weeks of the quarter to catch up on any course chapters not finished (e.g., if Chapter 8 or some advanced content remains, or if you want to skim Hugging Face's additional units like *Transformers for audio/vision or production optimization*). The Hugging Face course also has sections on more advanced topics (like optimizing models for production with quantization/pruning [6]); you can explore these as optional learnings if time permits. Reflect on the quarter's learning: write a short **retrospective** of what you accomplished, challenges overcome, and key learnings about GenAI. Plan for Quarter 2 by listing what you want to focus on next (e.g., "I want to dive deeper into fine-tuning larger models and understanding recent

transformer research"). *LinkedIn update:* Share a quarter-end reflection or a compilation of takeaways (this shows growth mindset and sets the stage for your next quarter's goals).

By the end of Quarter 1, you will have a solid grounding in transformer models and the Hugging Face toolkit, as well as a first GenAI project in your portfolio (with a live demo). This foundation will prepare you to tackle more advanced transformer architectures and larger-scale projects in the coming quarter. Your consistent LinkedIn updates will also start getting noticed; even if your GitHub had little audience before, driving traffic through these posts (and perhaps engaging with the Hugging Face community on Twitter/LinkedIn) will gradually grow your following.

Quarter 2: Advanced Transformers and Fine-Tuning a Foundation Model

Objective: Build on your foundation by exploring advanced transformer topics and applying them in a more complex project: **fine-tuning a foundation model** for a specialized task. In Q2, you will split your learning time between finishing advanced Hugging Face course chapters and **auditing Stanford's CS25: Transformers United** seminar for cutting-edge insights. The hands-on project will be to fine-tune a larger language model (LLM) or generative model and **demonstrate a specific GenAI technique** (e.g., fine-tuning with LoRA adapters, or training a model with a custom dataset) – this showcases your understanding of the theoretical foundations in practice. By the end of Q2, you'll have a fine-tuned model and an in-depth case study on your GitHub, plus deeper knowledge of state-of-the-art Transformer research.

Learning Focus: This quarter's learning is two-pronged: (1) **Advanced Hugging Face topics**, and (2) **Stanford CS25 seminar lectures**.

- **Complete Hugging Face LLM Course (Chapters 10–12):** These chapters cover *advanced* topics like curating high-quality datasets for LLMs, fine-tuning large language models, and building reasoning or chain-of-thought capabilities [7]. For example, Chapter 11 focuses on fine-tuning large models (likely including techniques for efficient fine-tuning) and Chapter 12 on building reasoning models. This knowledge directly feeds into your project. Pay special attention to any sections on techniques like low-resource fine-tuning (parameter-efficient fine-tuning, prompt tuning, etc.) since you'll apply one such technique. Hugging Face's content may discuss methods like LoRA or Prompt Tuning for large models – note those down as you'll likely use them.

- **Stanford CS25: Transformers United (V5) Lectures:** Each week, watch **one guest lecture from CS25** (the course is freely available via YouTube [8] and had top researchers presenting). These talks will expose you to the **latest research and applications of Transformers** across NLP and beyond. CS25 covers how transformers are pushing boundaries in areas like computer vision, reinforcement learning, generative art, biology, etc., and features prominent researchers like Geoffrey Hinton, Andrej Karpathy, and others [9]. By tuning in weekly, you'll learn about new ideas such as reinforcement learning with human feedback, transformer-based AI agents, multi-modal models, etc. This will broaden your perspective and potentially spark ideas for future projects. *Practically*, treat each lecture as a mini-class: take notes on the core idea presented and consider how it might relate to your work. For instance, if one lecture is about **Transformers in Vision** or *Transformers in diffusion models*, you might gain insight into multi-modal generative AI, which could be useful if you ever extend into image generation [10]. If a lecture discusses **AI agents or GPT-based systems**, note

how they evaluate or align models – relevant if you later build AI assistants. *(CS25 is a seminar – no homework – so it's a perfect lightweight way to stay current* [11] *.)*

Combining these: your structured learning (Hugging Face chapters) gives you *how-to* skills for fine-tuning, while CS25 lectures give you *big-picture* understanding of why and where to apply such techniques.

Project Focus: Fine-Tune a Foundation Model – This quarter you will undertake a more ambitious project: selecting a modern pre-trained generative model and fine-tuning it for a particular purpose. This demonstrates your ability to adapt large models (which is a valuable skill and showcases understanding of model internals). Some ideas for this fine-tuning project:

- Fine-tune a moderate-sized **LLM (e.g., 7B parameter range)** on a domain-specific dataset. For example, fine-tune Meta's **LLaMA-2 7B** (or another open-source model like GPT-J/GPT-NeoX) on a dataset of your choice (financial documents, healthcare texts, coding Q&A, etc. — something you're interested in). The goal could be to make the model an expert in that domain (answer questions or generate content specific to that field). You will likely use techniques like **Low-Rank Adaptation (LoRA)** and 4-bit quantization to make this feasible on a single GPU or cloud instance [12] [13]. *For instance, one Hugging Face tutorial demonstrates fine-tuning a 3.8 billion parameter model to speak in "Yoda" style using LoRA and bitsandbytes 4-bit compression* [12] *– such techniques allow fine-tuning large models with much less GPU memory by reducing the model size and training only small adapter weights.*

- Alternatively, train or fine-tune a **Transformer on a new modality or task** to broaden your GenAI portfolio. For example, you could fine-tune a Transformer-based image generation model (like Stable Diffusion's text-to-image model) on a themed image dataset, or fine-tune a speech Transformer (like Wav2Vec2) on some audio classification. This is optional and depends on your interest – the key is to demonstrate mastery of fine-tuning *generative models*, whether text or another modality. (Resources: Hugging Face's course segments "Transformers can hear/see" could guide you in applying transformers to audio/vision [10], and the *Diffusers Course* is available if you choose the diffusion route [14].)

- Another angle: implement a known **research paper or technique** on fine-tuning. E.g., implement a simplified version of Google's *UL2* fine-tuning strategy or OpenAI's RLHF (Reinforcement Learning from Human Feedback) on a small scale to demonstrate theoretical foundations. This could be more complex, so it's an stretch goal if you have extra time or interest in research. For most, sticking to standard fine-tuning with existing libraries (Transformers, PEFT, etc.) is sufficient.

The deliverable for the project will be: - The **fine-tuned model** itself (uploaded to Hugging Face Hub or at least the adapter weights if using LoRA, along with instructions to load it). - A detailed **report or documentation** (in your GitHub README or a separate Markdown) describing the process: how you prepared the data, which base model you started with, your fine-tuning method (e.g. "Using Hugging Face's Trainer with LoRA for efficient fine-tuning [15]"), how long/what resources it took, and the results (include metrics like before vs after performance, examples of output). - Possibly a **small demo** of the fine-tuned model's capability. This might simply be a few inference examples in a notebook or script. If the model is user-facing (like a chatbot), you could create a Gradio demo like in Q1 to let others try it.

This project highlights your ability to *adapt and improve a generative model*, which is a strong demonstration of practical understanding of GenAI.

Tentative Timeline (Quarter 2 Weeks 1–12):

1. **Weeks 1–2: Plan & Advanced Learning Completion** – Begin by wrapping up the Hugging Face course Chapters 10–12 on advanced LLM topics [7]. This gives you theoretical footing for fine-tuning strategies and data best practices. Simultaneously, finalize your plan for the fine-tuning project: decide on the **task and dataset** and choose an appropriate base model. For example, you might decide *"I will fine-tune a 7B language model on a dataset of financial news to create a finance-focused Q&A bot."* Ensure the dataset is obtainable (maybe use a public dataset or scrape texts if allowed) and not too large for 4-6 hour weekly work (you might use a subset if needed). Set up the environment for training: verify you have access to a GPU (could be a cloud VM, Colab with Pro if necessary, or your company's resources if permitted for personal use). If needed, spend time learning any new tool for fine-tuning (for instance, reading the Hugging Face documentation on the `peft` library for LoRA, or the `transformers.trainer` usage for causal language models [16]). *CS25:* Watch the **first 1–2 CS25 lectures** in these weeks, possibly "Overview of Transformers" (to reinforce fundamentals from a research POV) or any talk aligning with your project domain. *LinkedIn update:* announce your Q2 project choice (e.g. "I'm going to fine-tune Model X on Y – tackling domain-specific language modeling!") and share excitement for diving deeper.

2. **Weeks 3–4: Data Preparation and Baseline** – Start the fine-tuning work by preparing your dataset. Use Hugging Face Datasets to tokenize and create train/val splits. If needed, perform preprocessing (clean text, format into prompt-response pairs, etc.). These weeks, aim to run a small **baseline test**: e.g., run the base model on a few prompts from your dataset to see how it behaves pre-fine-tuning (and save these outputs for comparison later). Implement the fine-tuning pipeline: set up the Trainer or whichever method (could be using Hugging Face's `SFTTrainer` if following a community tutorial [17]). If using LoRA, configure the LoRA adapter (e.g., define `LoraConfig` as shown in tutorials [16]). Essentially, get everything ready so that the heavy training can start. *CS25:* Watch the next lectures – for example, if there is a lecture on *Reinforcement Learning and co-design* or *AI alignment* (which CS25 v5 had around April 8 by Karina Nguyen [18]), absorb those insights; they may be relevant if you consider how to evaluate your model's real-world usability or alignment. *LinkedIn update:* share how you prepared your data and any interesting challenges (like cleaning domain-specific text, or setting up the training framework). This shows your followers the meticulous work behind fine-tuning.

3. **Weeks 5–6: Fine-Tune the Model (Iteration 1)** – Kick off the fine-tuning process. Given your time constraints, you might not finish in one session – that's okay. Aim to train in manageable increments. For example, run the training for a certain number of epochs or hours each Friday, monitor the loss/metrics, and save checkpoints. Leverage efficient training tricks: use 4-bit quantization if needed to fit the model in memory (as described in recent Hugging Face blogs, converting model weights to 4-bit can reduce memory by ~8x [13]), and train only the LoRA adapter weights (freezing most of the model) to reduce computation. Track progress: is the model's performance improving on validation data? Adjust hyperparameters if needed (learning rate, batch size). By week 6, you ideally have a first version of a fine-tuned model. It may not be fully optimal, but you can already test it on some examples to see how it improved over the base model. *CS25:* Continue weekly lectures; perhaps around now, a lecture on *"Advent of AGI" by guest speaker* or *AI agents* appears – these visionary topics [19] [20] can be inspiring and remind you of the bigger picture of why fine-tuning domain models is useful on the path to more general AI. *LinkedIn update:* share an update about the training process – for instance, "Fine-tuning is underway! Using parameter-efficient fine-tuning (LoRA) to adapt a 7B

model – learned how quantizing to 4-bit lets me fit the model in memory [13]. Early results show X." Be sure to explain in simple terms for a broad audience (this educates others and demonstrates your mastery).

4. **Weeks 7–8: Refine and Evaluate** – Now focus on improving the model and evaluating its performance. If the first training run was insufficient, continue training for more epochs or implement improvements (maybe incorporate more data if underfitting, or regularization if overfitting). This is also a good time to evaluate qualitatively and quantitatively:

5. **Quantitative**: If you have labeled validation data, compute metrics (e.g. accuracy for classification, ROUGE for summarization, etc. depending on your task). If no standard metrics, consider proxy metrics (perplexity for language modeling, etc.).
6. **Qualitative**: Prepare a set of test prompts/questions for the model relevant to the domain and compare outputs between the fine-tuned model and the original base model. Document these examples.

7. If results are not as good as expected, debug: Maybe the model is hallucinating or not much better than base. This could lead you to adjust training (more data, different prompt format, etc.). Use your ML intuition here. By week 8, you should have the **final fine-tuned model** weights and a decent sense of its capabilities and limitations. *CS25:* Keep watching new lectures (e.g., transformers in multi-modal applications, or any remaining ones). There might be a talk on evaluation or on using transformers in production which could give you ideas for testing your model's reliability [21]. *LinkedIn update:* share some *results*. For example, "Fine-tuning complete! My custom model can now do XYZ. Here's an example: [prompt -> model output]." If there were improvements (or even surprising challenges), mention them. This shows the outcome of your technical work.

8. **Weeks 9–10: Project Documentation and Mini-Demo** – Use these weeks to consolidate everything you've done into a presentable form. Write a **comprehensive README or blog** detailing the project from start to finish. Explain the problem, data, model choice, fine-tuning approach, and results. Incorporate the evaluation examples and any charts (maybe a training loss curve or a table of metrics). This documentation is crucial for others (and recruiters) to understand the depth of your work. If possible, also create a **simplified demo** for the fine-tuned model. For instance, if it's a chatbot or QA model, you could create a Gradio web demo where a user asks a question and the model responds using your fine-tuned weights. (If the model is too large to host freely, you might use a smaller distilled version for the demo, or just host it locally and share a video/gif of it responding.) At minimum, provide a script/notebook in the repo to easily load and try the model. *CS25:* At this point you've likely watched ~8–10 lectures (if the Stanford seminar had that many). Finish any remaining ones. Notably, Stanford's CS25 often ends with forward-looking ideas (e.g., human-centric AI, ethical considerations). These might not directly impact your project but are great for rounding out your knowledge and might be worth mentioning in interviews or posts ("I also learned about X from Stanford researchers, which is important for the future of AI"). *LinkedIn update:* announce the **completion of your fine-tuning project**. This is a big milestone – explain what you built, link to your GitHub repo and any demo. Emphasize the skills demonstrated (e.g., "This project showcases my ability to fine-tune large-scale models and adapt cutting-edge research for real-world tasks"). Such a post not only marks your achievement but also serves as a mini tech article that others can learn from.

9. **Weeks 11–12: Learnings Integration and Networking** – In the final weeks, ensure you haven't missed key learning objectives. You might skim back through your CS25 notes and pick one or two topics that fascinated you for deeper follow-up (maybe read a paper or try a small experiment related to that topic). For example, if an *AI agent* lecture intrigued you, you could spend a bit of time exploring libraries like LangChain or a basic agent framework, just to plant seeds for future projects. Additionally, take time to **engage with the community**: consider writing a longer-form LinkedIn article or a Medium blog about your "Fine-tuning journey" – this can be a compilation of your weekly posts and lessons learned. Sharing a well-written article can increase your visibility. Also, if you used any open-source tools or got help from specific blogs (like a Hugging Face fine-tuning guide), consider thanking or tagging them on social media – this can spark new connections. *LinkedIn update:* do a Q2 wrap-up reflecting on how this quarter deepened your expertise (e.g., "Three months ago I had never fine-tuned a 7B model; now I have, and here are 3 things I learned about large-scale adaptation…"). Mention your gratitude for free resources (HuggingFace, Stanford CS25) – this shows humility and community appreciation.

By the end of Quarter 2, you will have an impressive **case study of fine-tuning** in your portfolio. You'll also have significantly expanded your theoretical knowledge by learning from top researchers (CS25) – being able to reference current AI developments in conversation is a big plus for a senior engineer. Your GitHub will now feature at least two solid GenAI projects (the Q1 demo and the Q2 fine-tuned model), and your consistent LinkedIn sharing will have further grown your professional audience. This sets you up for the final quarter, where you'll aim to **tie everything together into a capstone real-world project**.

Quarter 3: Capstone Project – Real-World GenAI Solution & Integration

Objective: In the final planned quarter, leverage everything you've learned to build a **capstone GenAI project** that is as close to a real-world application as possible. This means identifying a problem or use-case where generative AI can provide value, and building a solution around it end-to-end. The project should ideally integrate multiple components (e.g., an LLM with external data or tools) and demonstrate both your technical skills and understanding of how GenAI can be deployed usefully. By the end of Q3, you will have a showcase-worthy application in your GitHub (and possibly live for others to try), and you will have documented your journey throughout – making you confident to present this to hiring managers or at meetups.

Learning Focus: Unlike previous quarters, there may not be a single course to follow here – instead, **self-driven research and specific skill acquisition** will guide your learning: - **Identify Specific Technologies or Skills Needed for Your Project:** Once you choose a capstone idea (see Project Focus below), determine what new things you need to learn. For example, if you decide to build a chatbot that answers questions about a set of documents, you might need to learn about **Retrieval-Augmented Generation (RAG)** with vector databases. RAG is a technique that combines LLMs with an external knowledge base to reduce hallucinations and provide up-to-date, domain-specific information [22]. You would then focus your learning on how to implement a RAG pipeline (reading blogs or tutorials on LangChain, Haystack, or LlamaIndex usage, etc. – e.g., how to index documents, perform similarity search, and feed context to the LLM [23] [24]). If your project involves multi-modal inputs/outputs (say generating an image or speech), you might need to learn a bit about those domains (maybe a quick dive into Hugging Face Diffusers or text-to-speech tools). Essentially, **make a list of learning targets** relevant to your project and tackle them at the start of the

quarter. - **Stay Updated with Current GenAI Trends:** Allocate some learning time to stay abreast of new developments. Quarter 3 (which will be later in 2025 or 2026) might bring new tools or libraries. Skim through AI news, popular GitHub projects, or arXiv papers weekly to catch anything that could enhance your project. You might discover, for example, a new open-source model that could be better for your task, or a library that simplifies deployment. Since you've completed formal courses, this "learning" is more unstructured – think of it as transitioning from student to independent researcher/engineer. - **Possibly engage in a community**: If available, join an online study group or Discord for GenAI enthusiasts (for instance, the Stanford CS25 Discord which you joined earlier [25] or Hugging Face's community channels). Discussing your project idea with others can provide feedback and new ideas. Also, consider attending webinars or conference talks if any are accessible; these can inspire and inform your approach.

Project Focus: Build a Real-World GenAI Application (Capstone) – This is the highlight of your personal hackathon series. The project should be something that **solves a real problem or provides a useful service** using generative AI, integrating the skills from prior quarters. Given your experience and what you've learned, here are a few capstone project ideas:

- **Intelligent Document Assistant (QA Chatbot with RAG):** Create a chatbot that can answer questions about a collection of documents or data (for example, company policies, financial reports, technical manuals – whatever interests you). This would involve using an LLM in combination with a **vector database** of embeddings of your documents – a classic Retrieval-Augmented Generation setup. The RAG pipeline has two main phases: indexing the data into vectors, and then at query time retrieving relevant chunks and generating an answer [23] [24]. You can use open-source tools like Chroma or FAISS for the vector store and a library like LangChain or LlamaIndex to orchestrate the process. The result would be a functional QA system that is *actually useful* for people who need information from those docs. This demonstrates your ability to **combine an LLM with external knowledge** and address the common LLM limitation of factual accuracy [22]. It's also a very in-demand skill (many companies want custom chatbots for their data).

- **Generative Data Analysis or Advisor:** Leverage an LLM to create a tool that helps with a specific professional task. For example, a "Financial Report Analysis Assistant" that takes in a company's quarterly report and generates an analysis summary, or a "Code Review Helper" that takes a code diff and produces review comments. This would involve prompt engineering and possibly fine-tuning (you could even use your Q2 fine-tuned model if applicable) to specialize the LLM's behavior. The real-world angle comes from targeting a practical use-case (finance, coding, marketing, etc.). If it's finance-related, this plays to your Capital One background and might be attractive to future fintech employers – it shows you can apply GenAI in a domain context.

- **AI Agent for Task Automation:** If you're adventurous, build a small-scale **AI agent** that can perform a sequence of actions to accomplish a goal. For instance, an agent that, when given a task (like "research topic X and write a summary"), will use an LLM to break the task into steps, perhaps use tools (search the web, etc.), and then produce a result. This concept was popularized by projects like AutoGPT and BabyAGI. You could use frameworks like LangChain to give your LLM tool-using abilities (like calling a search API, reading from a file, etc.). While a full AutoGPT may be too large to implement alone, you can design a narrower agent for a specific domain (e.g., an agent that takes a URL of a webpage and uses an LLM to read it and answer questions about it, effectively combining retrieval + reasoning). This kind of project shows you're on the cutting edge of GenAI applications. It

will involve ensuring the agent remains controlled and evaluating its outputs carefully (something you learned about alignment and evaluation possibly from CS25 discussions).

- **Multi-modal Generative Application:** To diversify your portfolio, you could incorporate another modality: e.g., *generative art* or *audio*. For example, an app that generates **custom images from text prompts** for a niche (say concept art for cooking recipes or slides for presentations). Using Stable Diffusion via Hugging Face's `diffusers` library, you can fine-tune a diffusion model (the HF Diffusers Course covers how to train/fine-tune diffusion models [26]) or at least customize prompts to make a themed generator. Another idea: a simple **text-to-speech voice assistant** – use an open text-to-speech model to make your chatbot talk. These additions aren't necessary, but if you have interest, they show you can work with more than just text.

Choose **one major project** that excites you and has a clear real-world value proposition. Given 12 weeks, scope it so that it's achievable: it's better to have a smaller feature set that works end-to-end than an over-ambitious project that remains unfinished. Importantly, this capstone should integrate the knowledge from previous quarters (e.g., you might use your fine-tuned model from Q2 as a component, or apply the demo deployment skills from Q1 to launch this project for users).

Deliverables will include: - A fully working application (running either on a web app, or at least a well-documented script). If possible, host a **live demo** (Hugging Face Spaces or a simple web app) so people can actually use your solution. - The **source code** on GitHub, organized professionally (this is where you really polish your software engineering skills: proper structure, config files, perhaps some tests, and clear instructions to run). - A **write-up** (README or separate blog) that describes the problem, how you solved it using GenAI, and any notable technical challenges. This is your chance to narrate a story: "I noticed problem X. I designed a GenAI solution Y. It works like this... Here are examples... Here's why it's useful."

Tentative Timeline (Quarter 3 Weeks 1–12):

1. **Weeks 1–2: Define Project & Ramp-Up** – Finalize your capstone project idea. Outline the project requirements and components. For instance, if doing the Document QA bot, list steps: data ingestion, vector DB setup, LLM inference with retrieval, front-end UI. Identify what you need to learn or set up: e.g., "Learn to use Chroma or FAISS for vector search," or "figure out how to call an external API from within a Python script for the agent." Start learning those specific pieces: read documentation, follow a small tutorial for each tool. If possible, do a **proof-of-concept** for the core mechanism: e.g., for RAG, maybe quickly index a single sample text and see if you can retrieve and answer a question with a known open-source model. This will validate your approach. Create a repository for the project and set up the basic structure (even if code is minimal at this point, having a skeleton helps). *LinkedIn update:* announce your capstone project choice and why you think it's exciting or valuable ("For my final quarter, I'm building an AI-powered XYZ that will [solve problem]. Can't wait to apply everything I've learned to make this real!"). Generating anticipation can keep your network engaged.

2. **Weeks 3–4: Backend Development – Core AI Pipeline** – Begin implementing the main AI pipeline of your project. For example, if it's the QA chatbot:

3. Implement the **indexing process**: load documents, chunk them, generate embeddings (you might use a model like `sentence-transformers` for embeddings if not using the LLM itself for that), and store in a vector database [27].

4. Implement the **retrieval + generation** flow: given a query, retrieve top relevant chunks [24], then prompt the LLM with those and the question to get an answer [28]. If you have a fine-tuned model from Q2 that's suitable, integrate it here; otherwise you might use a pre-trained model via API or local (depending on resources). For an agent project: implement the loop where the LLM chooses an action (like "search") and your code executes it, then feeds result back – ensure you have a way to break out to avoid infinite loops. Focus on getting a **working prototype** of the logic, even if it's rough (e.g., no UI yet, and using small data). This is the engine of your application. Meanwhile, continue any learning needed (you'll likely learn a lot by doing here – consult Stack Overflow or forums when you hit issues). *LinkedIn update:* share a tech snippet – maybe "I got vector search working" or "Connected an LLM with a tool successfully." This shows the technical progress and gives you content to discuss (for instance, "I learned how to use LangChain to do X and it's pretty cool.").

5. **Weeks 5–7: Full Development – Application & UI** – Expand your prototype into a full application. This includes:

6. **Frontend or Interface:** If it's a web app, develop the interface (could be a simple Gradio or Streamlit app for expedience). Make it user-friendly: for a chatbot, a chat-style interface; for an analysis tool, maybe a form to upload a file or enter text. Leverage templates or examples (Gradio has many demo examples).
7. **Integration:** Make sure the front-end calls your back-end pipeline correctly. Handle edge cases (no results found, long queries, etc.) gracefully.
8. **Performance considerations:** Since this is a "real-life" solution, think about response time and resources. Perhaps limit the size of input or number of retrieved chunks to keep latency reasonable. If the model is large and slow, consider ways to optimize (maybe use a smaller model if needed, or do some caching of embeddings etc.). This doesn't have to be enterprise-grade, but showing awareness of these issues is good.

9. **Testing:** Manually test your application with various inputs. Does the QA bot correctly answer different types of questions from the docs? Does the agent perform the intended task reliably? Identify failure cases and either handle them or note them as future work. By end of week 7, aim to have a **functionally complete** version of the application. *LinkedIn update (around week 7):* announce that a working version is nearly ready. Maybe share a short screen recording or image of it in action (visuals grab attention). For example, a screenshot of the chatbot answering a question, or a generated image from your app. Tease the upcoming release and what it can do.

10. **Weeks 8–9: Polish, Optimize, and Deploy** – Now improve and polish the project:

11. **Polish UX/UI:** Refine the interface look (even small CSS tweaks or adding instructions for users helps). Ensure the input/output formatting is clean (e.g., the bot's answer is well-formatted, maybe with sources if your RAG can show which document it came from).
12. **Optimize:** If you have time and it's needed, optimize the pipeline. For example, if using a local model, see if using half-precision or quantization can speed it up. Or reduce the number of documents considered if quality remains good to gain speed. Ensure memory usage is within limits for your hosting environment.

13. **Deployment:** Deploy the app for public access. For Gradio on HuggingFace Spaces, this means pushing your code to the Space – you might have done this in Q1, so reuse that knowledge. If the app is heavier (e.g., needs a GPU), request a GPU Space or use an alternate like AWS or a personal server. You could also deploy on a free service like Heroku or Streamlit Community Cloud if appropriate.

14. **Monitoring:** Once deployed, test it live as a user would. Nothing like a real environment test to catch final bugs. Additionally, prepare your **project documentation**. Write a detailed README and/or a separate document describing how everything works. Include instructions in case someone wants to run it locally. *LinkedIn update (week 9):* Officially **launch your project**! Share the link to the live demo or repository. Write this post as if announcing a product: clearly state the problem it solves, show an example of it working, and invite people to try it or give feedback. This could be your most important post yet, as it culminates the entire series. Tag any relevant organizations or use hashtags (e.g., #GenerativeAI, #LLMs, etc.) to broaden reach. Given your consistent posting, some of your connections might reshare or comment, increasing visibility.

15. **Weeks 10–11: User Feedback and Iteration** – After launch, gather any feedback from users or peers. Perhaps colleagues or LinkedIn contacts will comment – address their questions, and consider their suggestions for minor improvements. This is a short cycle of iteration: e.g., if someone found a bug or the agent did something funny, fix it; if users ask for a feature that is small, maybe add it. This shows responsiveness and that you can improve a product post-release. If the project is quite complex, you might not have much user feedback, so alternatively use this time to do your own analysis: What could be future improvements? (Maybe the model could be bigger or fine-tuned more, maybe the UI could handle more concurrent users, etc.) Document these as "Future Work" in your README. Also, consolidate everything you've learned in this capstone process: you likely picked up new tech skills (like vector DBs or web frameworks). Make note of these to mention in resumes or interviews. *LinkedIn update:* share a behind-the-scenes or a lessons learned post. For example, "Building this app taught me X about combining knowledge retrieval with LLMs… One key challenge was Y, which we overcame by Z [22]. If anyone is looking to do something similar, happy to share insights." This positions you as someone knowledgeable and willing to help others.

16. **Week 12: Quarter (and Journey) Retrospective & Next Steps** – Congratulations, you've reached the end of the third quarter and accomplished a tremendous amount! Use this final week to reflect on the **entire 3-quarter journey**:

17. Summarize the progression: from learning fundamentals to fine-tuning a model to building a real application. Recognize how far you've come in terms of skills and portfolio. This narrative can be very powerful when talking to recruiters or writing cover letters – you have a story of continuous growth.
18. Update your resume/LinkedIn profile to include the projects and new skills. Perhaps write a line under your job experience about this self-driven "AI Hackathon" initiative, emphasizing initiative and self-learning.
19. Think about how to continue the momentum. The plan was for 3 quarters, but learning never really stops in AI. You might already have ideas for new projects or maybe contributing to open source. Jot these down as potential future goals. For instance, you could aim to present your capstone at a local tech meetup or internal demo at work (if appropriate).
20. *LinkedIn final update:* write a **journey conclusion post**. This is a bit like a graduation speech of your Personal Hackathon. Thank those who followed along or gave feedback. Highlight key accomplishments (perhaps a bullet list: X courses completed, Y models fine-tuned, Z apps built).

Share how this experience has prepared you for your next challenges. Importantly, mention that you are open to new opportunities or collaborations – since one goal was to showcase to future employers, explicitly signaling that can open doors. This final reflection not only cements what you learned, but also demonstrates to everyone your dedication and ability to drive your own development.

By the end of Quarter 3, you have a capstone project that exemplifies applying generative AI to solve real problems. Along with the projects from Q1 and Q2, your GitHub now shines with examples of your work – ranging from notebooks and demos to full applications. You will have substantially grown your network's awareness of your expertise through consistent LinkedIn sharing. In essence, you have **proven your ability to learn and apply cutting-edge AI** independently, which is a highly attractive quality in the job market.

Conclusion and Tips for Success

Over these three quarters, you will have transformed your Friday nights into a productive, structured learning and building routine. To maximize success:

- **Stay Consistent:** Treat these hackathon sessions as important appointments with yourself. Consistency is key – even if some weeks progress feels slow, over months it compounds into major achievements.
- **Leverage the Community:** Use forums, Discords, and social media to ask questions and share progress. You might be surprised – people are often willing to help or cheer you on. Since you're sharing on LinkedIn, you might connect with others on a similar journey.
- **Documentation & Code Quality:** Given that you want to impress future employers, pay attention to clean code and good documentation in your projects. This shows professionalism. A well-written README with clear instructions and insights is often as important as the code itself.
- **Balance Learning and Doing:** Each quarter is designed with half learning, half doing. If you ever find yourself stuck in "tutorial hell" or conversely coding without direction, remember to rebalance. The courses and talks guide you, but the projects are where you cement knowledge. Conversely, if a project hurdle is tough, step back and do a bit more research or learning. This dynamic balance is what makes self-learning effective.
- **Adapt the Plan if Needed:** The plan is comprehensive, and it's okay if you need to adjust based on actual progress. If Q1 takes 14 weeks, or you swap a project idea in Q3, that's fine. The key outcomes (foundation, fine-tuning experience, capstone app) matter more than rigid timelines.
- **Maintain Work-Life Balance:** Since this is your Friday night, ensure this remains enjoyable and not a source of burnout. The topics are exciting – keep it fun by choosing projects that genuinely interest you. If needed, team up with a friend for some sessions or reward yourself after a big milestone.

By following this plan, after three quarters you will have a robust portfolio and demonstrable expertise in generative AI. You'll have shown that you can **learn continuously**, **apply knowledge creatively**, and **share your journey** – all excellent qualities for a machine learning engineer in the eyes of employers. Good luck with your Personal Friday Night AI Hackathon, and enjoy the journey of learning and building!

Sources:

- Hugging Face Transformers Course – outlines fundamental through advanced LLM training and demo sharing [1] [2]. This course (free and open-source) ensures proficiency in using the Hugging Face ecosystem for NLP/LLMs.
- Stanford CS25: *Transformers United* – a seminar featuring leading researchers discussing cutting-edge Transformer research and applications across domains [9] [29]. Auditing these lectures provides current insights beyond basic tutorials.
- Hugging Face Fine-Tuning Guide (2025) – example of fine-tuning a 3.8B parameter language model with LoRA and 4-bit quantization [12] [13], demonstrating techniques you can use in your Q2 project to adapt large models on limited hardware.
- Medium – *Retrieval-Augmented Generation (RAG) Pipeline* explanation, describing how combining LLMs with a vector database of knowledge can ground answers in external data [22] [23]. Relevant for designing a real-world QA chatbot in Q3.

---

[1] [2] [3] [4] [5] [7] Introduction - Hugging Face LLM Course
https://huggingface.co/learn/llm-course/chapter1/1

[6] [10] The Hugging Face course goes open-source! : r/learnmachinelearning
https://www.reddit.com/r/learnmachinelearning/comments/tr0s6a/the_hugging_face_course_goes_opensource/

[8] Stanford CS 25 Transformers Course (OPEN TO EVERYBODY)
https://www.reddit.com/r/learnmachinelearning/comments/1k54bfd/stanford_cs_25_transformers_course_open_to/

[9] [11] [18] [19] [20] [21] [25] CS25: Transformers United V5 | CS25
https://web.stanford.edu/class/cs25/

[12] [13] [15] [16] [17] Fine-Tuning Your First Large Language Model (LLM) with PyTorch and Hugging Face
https://huggingface.co/blog/dvgodoy/fine-tuning-llm-hugging-face

[14] [26] Hugging Face Diffusion Models Course - Hugging Face Diffusion Course
https://huggingface.co/learn/diffusion-course/en/unit0/1

[22] [23] [24] [27] [28] How I built a Simple Retrieval-Augmented Generation (RAG) Pipeline | by Dr Julija | Medium
https://medium.com/@drjulija/what-is-retrieval-augmented-generation-rag-938e4f6e03d1

[29] Stanford University Explore Courses
https://explorecourses.stanford.edu/search?view=catalog&filter-coursestatus-Active=on&q=CS%2025:%20Transformers%20United&academicYear=20212022