

# Project Plan: Multi-Agent AI Presentation Generator & Beautifier

## Goal Alignment with Hackathon Criteria and Prizes

We are building a multi-agent presentation generator & beautifier using OpenAI's new open-weight models (gpt-oss-20b/120b). This aligns our project with the OpenAI Open Model Hackathon goals and prizes. In particular, we will target the "Most Useful Fine-Tune" category (sponsored by OpenAI, \$5,000) by fine-tuning gpt-oss on presentation-building tasks. We'll also design for a shot at "Best Overall" (\$10,000) by delivering a polished, impactful application.

The hackathon's judging criteria emphasize: effective use of gpt-oss, good design/UX (with user safety), high potential impact, and novelty . We have woven these into our plan:

- Application of gpt-oss: We will showcase the model's strengths uniquely – fine-tuning it for slide generation and leveraging its built-in agentic abilities (function calling, large context) . This ensures our solution isn't something any generic model could do .
- Design and UX: We plan a user-friendly interface with thoughtful workflow and safety checks, balancing a solid backend with a clean frontend .
- Impact: The tool addresses a broad need (fast, quality presentation creation), benefitting professionals, educators, and more. We'll highlight how it saves time and works even offline – potentially impacting users without access to expensive tools or reliable internet .
- Novelty: Our multi-agent approach and fine-tuned open model concept is creative and unique – no existing product quite combines these. We're improving on known ideas (AI slide makers) with a new twist .

Sponsor Goals: OpenAI's hackathon prompt encourages "creative, unexpected" applications of these models – e.g. "fine-tune a model into something indispensable" . By fine-tuning gpt-oss for a specialized but widely useful task, we hit that mark. Hugging Face, as Best Overall sponsor, will appreciate an open-source community utility, and we plan to release our fine-tuned model on HF Hub. NVIDIA sponsors the "Local Agent" and hardware categories – our project running fully locally on GPU demonstrates the value of their hardware. (We'll note that gpt-oss-20b can run on a single 16 GB GPU, making on-device use feasible .) In short, our plan aligns with what each sponsor is looking for, maximizing our chances.

# Market Landscape: AI Presentation Tools & Gaps

AI-powered presentation builders are emerging, but we've identified key gaps we can fill.

Popular tools include Tome, Beautiful.ai, Presentations.AI, Canva's Magic Design, Gamma.app, and others. They let users generate slide content from a prompt, often with AI-chosen imagery and styling. For example, Tome can create a deck from a prompt or imported Google Doc, and even offers multiple draft variations for each slide. However, existing solutions have limitations that our project will overcome:

- **Quality of AI-Generated Content:** Reviews note that AI slides often need heavy editing. "Tome's AI-generated text is fairly good but can be a bit wordy... you'll need to edit to avoid sounding like a robot." Many tools produce generic or verbose results that require manual polish. → Our fine-tuning aims to yield concise, well-structured bullet points in a natural tone, reducing the editing needed. We'll train the model on presentation-style writing to avoid robotic verbosity.
- **Limited Customizability & "Beauty":** Some products offer minimal design control in free versions. For instance, Tome lacks a robust template library and doesn't support exporting to PowerPoint. Presentations.AI (a Tome competitor) highlights that Tome has "no style template library... no support for PowerPoint export... no advanced features like animations". In contrast, Presentations.AI itself touts "AI-generated presentations and templates", "vast libraries of images and icons", and even matching decks to a brand style. → Our project will include theme templates and high-quality PPTX/PDF export. We plan to support at least a basic library of slide themes (with cohesive fonts/colors), and allow branding (e.g. insert a company logo or color scheme) to make slides presentation-ready. Advanced features like animations are lower priority for the hackathon, but we'll note them as future enhancements.
- **Input Flexibility:** Many users want to generate slides from existing content (reports, docs) not just a short prompt. Some AI tools now offer this – e.g. "Import any type of document and convert into a deck" is a feature of Presentations.AI, and Tome is noted as "best for creating presentations from existing documents". → We will support both prompt-based generation and document upload. For example, a user could supply a lengthy article or markdown notes; our system will parse and summarize it into slides. Thanks to gpt-oss's 128k token context window, we can ingest large documents in one go – a big differentiator from many existing models. This means even a full report or multiple chapters can be summarized into a slide deck without chunking.
- **Reliance on Proprietary/Cloud Services:** Most current tools are proprietary SaaS (Tome, Beautiful.ai, etc.) requiring internet access and come with subscription costs or limited free tiers. "You'll need a Tome subscription to use the AI presentation maker... a free trial is not available". This can be a barrier, and data privacy is a concern when uploading sensitive content. → Our solution runs locally with open models. By using gpt-oss (Apache 2.0 licensed) and possibly self-hosted image generators, all processing can be

done on the user's machine for privacy . There are already open-source efforts proving demand for this: Presenton is an open-source AI presentation app that runs fully locally via Ollama (for LLMs) . It offers features like mixing text & image generation providers and privacy-first design . Presenton's success (1.5k+ stars) shows an appetite for offline slide generators – we will build on this concept, adding our multi-agent twist and fine-tuned model to go even further.

In summary, existing tools demonstrate the feasibility and demand for AI in slide creation (saving users time and effort ), but none combines open-model fine-tuning, multi-agent intelligence, and user-centric design. This is our opportunity to stand out. We'll deliver a solution that produces higher-quality slides (less generic, more polished), offers greater control (themes, edits, exports), and respects privacy and cost constraints by running on open infrastructure.



## **Solution Concept: Multi-Agent Presentation Generator & Beautifier**

Our project will employ multiple AI agents collaborating to generate and refine a presentation. The end result is a well-structured slide deck with concise bullet points, relevant visuals, and a consistent design theme – all produced with minimal user effort.




How it works: The user will interact with a simple interface to specify the content they want in slides. They have two main options:

1. Prompt Mode: Enter a brief description of the presentation topic (e.g. "Climate change impacts and solutions overview"). Optionally, also select preferences like the desired number of slides and language.
2. Document Mode: Upload or paste a source text (e.g. a project report, an article, or meeting notes). The system will summarize and convert it into slides.

After the input is provided, our multi-agent pipeline is triggered:

-  Outline Planner Agent: This agent (powered by gpt-oss) parses the input and decides on a high-level outline – essentially slide titles or key sections. It determines the logical flow of the presentation. For example, for a climate change topic it might plan slides: 1) Introduction, 2) Evidence of climate change, 3) Impact on ecosystems, 4) Mitigation Strategies, 5) Conclusion. The Outline Agent ensures coverage of all important points from the input (it can use the model's large context to not miss details in a long document).
-  Content Generator Agent: For each slide in the outline, a content agent expands it into succinct bullet points. It uses the fine-tuned model to generate bullet-style sentences

rather than long paragraphs. We will instruct it (via prompts or system role) to follow best practices: e.g. ~5 bullets per slide, <15 words each, use clear language. The fine-tuning on presentation data will help it naturally produce text in this style. If the outline agent provided subtopics, the content agent ensures each is addressed on the slide.

-  **Design & Layout Agent:** This agent handles the “beautifier” aspect. It picks an appropriate theme/template (or follows the user’s choice of theme) and prepares the slide layouts. Concretely, this means selecting background style, font, color scheme, and positioning of text and images. We will have a set of pre-designed themes (e.g. “Corporate Blue”, “Minimal White”, “Dark Mode Modern”) – essentially HTML/CSS or PowerPoint template settings. The Design agent can also decide if a slide’s content would be clearer as e.g. a two-column layout or with an icon, etc., and tag it accordingly. (Some of these decisions may be rule-based due to time constraints, but the agent could make certain calls – for instance, “This slide lists statistics, use a chart icon”.)
-  **Visual Media Agent:** To make slides engaging, this agent finds or creates relevant visuals. We plan to integrate image generation or retrieval:
  - For literal topics (e.g. “rising CO2 graph”), the agent might call a chart API or search function (if data is provided).
  - For general topics, we can use an AI image generator (like DALL·E 3 or Stable Diffusion) or a stock photo API (Pexels/Pixabay) to fetch illustrative images. Notably, the open model supports tool use: “Built-in tool calling makes it agent-ready... matching GPT-4’s function calling syntax.” . We will leverage this by giving the model access to a function like `search_image(query)` – the Visual Agent (really the model itself in a tool-using role) can generate an image prompt from the slide text and retrieve an image URL.
  - We’ll constrain this to safe-for-work images only (using stock APIs’ filters or a moderation step) to address safety.
  - This agent ensures each slide isn’t just text. For example, it might add a relevant icon for each bullet or a background image if appropriate. (Presenton uses “Versatile Image Generation — choose from DALL-E 3, Gemini Flash, Pexels, or Pixabay” ; we can implement similar flexibility.)
-  **Revision/Critique Agent:** Once the draft slides are prepared (text + visuals + style), we’ll employ a final agent to review and refine. This could be the original model prompted to act as a “presentation coach” that checks the output. It might trim any overly long bullet, suggest simplifying jargon, and ensure consistency of tone. For instance, if one slide has a mismatched style or an irrelevant tangent, the critique agent would flag or fix it. We can implement this as another pass of the model with instructions to output

any recommended changes, which we then apply.

- We will also rely on user feedback in this stage: the app will show the draft outline for user approval early on (as many tools do). The user can edit or reorder slide titles before content generation. This human-in-the-loop step ensures the final output meets user expectations and also implicitly handles safety (user can remove any unwanted section the AI might have added).

These agents together function as an AI team, guided by a central controller (our program) that passes the results from one to the next. In practice, some of these roles will be implemented via sequential prompting of the gpt-oss model with different system instructions. Others (like image searching) will be actual external API calls triggered by the model's function outputs.

**Fine-Tuning the Model:** Our fine-tuned gpt-oss-20b will be the backbone of the Outline and Content agents (and likely also help with revision). Fine-tuning is key to achieve specialized performance:

- We'll fine-tune on a dataset of document-to-presentation pairs. For example, take an article or report and the slides summarizing it. If public data is scarce, we can generate synthetic training data using GPT-4 to create "ideal" slide decks from random texts. This trains the model to structure information into slides and use bullet point style.
- We will also include examples of good slide writing (concise, declarative bullets, no run-on sentences). By training on these, the model will internalize the "presentation tone".
- If possible, we'll fine-tune the model to output in a structured JSON format for slides (with fields for title, bullets, image\_description, etc.). The hackathon's Harmony format might assist here – gpt-oss supports a special response format for structured outputs and multi-channel reasoning . We could utilize that to get a well-structured result that our app can easily render.

Using gpt-oss has advantages beyond fine-tuning: it has a huge context window and is designed for reasoning tasks . We can feed large texts directly for summarization, unlike many models. Also, since it's open, we can run it on our own hardware. If the 20b model doesn't meet quality needs, we might attempt to fine-tune the larger 120b model for a final demo (if we secure an H100 GPU in the cloud). The 120B model reportedly fits in 80GB VRAM and has stronger reasoning – winning Best Overall might justify going big. But given resources, 20B + clever prompting/fine-tuning should suffice.

**User Experience & Interface:** We want the project to look as impressive as it works, to satisfy the Design criterion. The planned user flow:

1. **Input Stage:** The user enters a prompt or uploads a document. They select options like number of slides, output language, and choose a theme template (or “auto” for the AI to pick one).
2. **Outline Preview:** The Outline Agent’s output is shown as an editable outline. The user can tweak titles or add/remove slides. (This is similar to Presenton’s step: “Review and edit outline” before generating content .) This gives users control and builds trust in the AI.
3. **Slide Generation:** Upon confirmation, the Content, Design, and Visual agents do their work. The UI might show a loading animation per slide. Within a short time, a full draft presentation appears.
4. **Review & Edit:** The user can scroll through the slides in the app. If something is off, they can edit the text directly or press a “refine” button on a slide to trigger the Revision Agent for that slide (e.g. “Make this slide less wordy”). Because manual editing and iterative prompting are important (AI might not be perfect first try) , we’ll allow both.
5. **Export/Share:** The finished deck can be exported as a PowerPoint (.pptx) and PDF. This addresses compatibility needs (users can further tweak in PowerPoint or present offline). Many AI tools oddly don’t allow PPT export (Tome didn’t initially ); our solution will shine here by offering high-fidelity PPTX export . We’ll use a library (like Python-pptx or a JSON-to-PPT template approach) to programmatically build slides with the chosen theme and content. In addition, we can offer a web link to view the slides (perhaps using a simple web page viewer), but since we emphasize offline, local files suffice.




Throughout the UX, we’ll pay attention to safety and user agency:

- The content generation will be guided by a “safe completion” policy (avoiding disallowed content). Our fine-tune dataset will exclude toxic content, and we’ll use the model’s system message to enforce a professional tone. If a user inputs an inappropriate prompt, we’ll handle it by refusing or warning (as per OpenAI’s usage policy, which still applies to model usage).
- The user’s data stays local (if running the app locally). If we provide an online demo (e.g. via Hugging Face Space), we’ll clearly inform users that the data is processed by our server for demo purposes.
- By giving users the chance to review outline and edit final content, we ensure the AI is a helpful assistant, not a fully unchecked automaton. This addresses the “including safety of the user” aspect in judging .




In essence, our solution will feel like a smart assistant (or rather a team of assistants) that collaborates with you to craft a presentation. You provide the raw ideas or material; the AI team organizes it, writes it, makes it look good, and hands you a ready-to-use slide deck. It's fast, private, and tailored – something we believe will impress both judges and users.

## Key Features and Innovations

Our project's feature set is designed to maximize our odds of winning by hitting all the marks of an outstanding hackathon entry. Below are the key features, with notes on their implementation and how they address judging criteria:



-  **Fine-Tuned OpenAI Model (gpt-oss) for Slide Generation:** This is the core of our project. By fine-tuning gpt-oss on presentation data, we showcase the model's capabilities in a specialized, highly useful way – exactly what the “Most Useful Fine-Tune” prize is about. Fine-tuning will improve the quality of content (clear, on-point bullets) and ensure the model follows structured output formats. It also differentiates us from competitors who might just prompt GPT-4; we are investing in our own model specialization, which is novel and impressive. Moreover, using an open model means we can demonstrate it running locally (great for the “Local Agent” theme), and even share the fine-tuned model publicly for community impact. Technical detail: We'll likely use Hugging Face Transformers with the provided fine-tuning guide (which the hackathon resources link to) to train our model, possibly using parameter-efficient tuning (LoRA) to save time. This feature directly targets the Application of gpt-oss criterion – we're not just using the model, we're extending it in a unique way.
-  **Multi-Agent Architecture (Reasoning & Tool Use):** Rather than a single prompt-to-slides model, we implement multiple agents (as described) for a more robust and intelligent system. This design is innovative and leverages the model's agentic features. Judges will appreciate that we're using “chain-of-thought and tool calling support” of gpt-oss to build a complex application, not just a trivial Q&A bot. The multi-agent approach also improves results (e.g., one agent's output becomes another's input, which is a form of scaffolded reasoning). It adds a “wow factor” – it's essentially an AutoGPT-like slide creator. For the demo, we might even show a quick snippet of the agents' conversation or plan (transparency can impress on novelty). This addresses Novelty of Idea and also Design, since our system design is quite thoughtful.
-  **Automated Slide Design and Theming:** We'll have built-in templates for slide design (color schemes, layouts) and let the AI apply them. The user can choose a theme or have the AI pick the best fit. This feature ensures the output is not just textually correct but also visually appealing and professional, which boosts our Design/UX score. It's one thing to spit out text; it's another to produce a nicely formatted deck ready for use. We saw competitors highlight this: Presentations.AI stresses “AI-powered customizable presentation templates” as a key feature, and we will deliver something similar. Implementing themes might involve HTML/CSS if we use a web-based viewer or using

PowerPoint template files for export. Either way, it's a matter of mapping the AI's chosen style to a set of visual properties. This feature also allows potential branding (important for impact in business use-cases): e.g., user could input their company's brand colors and we adapt the theme, a capability that Presentations.AI markets heavily. Even if we just show a concept of it, mentioning it will signal we understand user needs in real-world scenarios (beyond hackathon).

-  Integration of Visuals (Images/Icons/Media): Each slide will have relevant imagery or media generated or retrieved by the AI. This makes presentations more engaging. Technically, we'll integrate with image generation APIs (depending on what's available—since DALL·E 3 might require OpenAI API access, we might use Stable Diffusion locally or stock images which are free). We already have a plan to allow multiple providers (similar to Presenton's "Multi-Provider Support" for text and images). Visually rich output will make our demo stand out to judges (it's literally more eye-catching than plain bullet lists). It also shows we considered completeness – a presentation isn't just text. This contributes to Impact (slides with visuals are more effective for audiences) and shows a high level of effort in implementation (not trivial). We will need to handle this carefully within 31 days, but we can start with static relevant images (like use keyword search to fetch images) and upgrade to generative if time permits. Even a few illustrative images in the final output will significantly elevate the perceived quality.
-  Multi-Language Support: Our model will be instructed (and fine-tuned data permitting) to handle multiple languages for slide content. The UI will allow selecting the language of output. This is feasible because gpt-oss is presumably trained on a variety of languages (and if not, we can still attempt translation via the model). By including this, we broaden the project's impact – for example, a user can generate the same presentation in English and Spanish effortlessly. It aligns with "benefiting all of humanity" goals by making knowledge more accessible. It's also a slight competitive edge: many existing tools focus on English. We can demo this feature by generating a slide deck in another language (if the judges are international, that's a plus). Technically, this means having training or prompting in other languages – we likely won't fine-tune separately per language, but we'll prompt the model (which is instruction-tuned) with "Respond in X language." And since we allow document input, a user could input a non-English document and get slides in that language (or even translated slides in another language). This global mindset can score points in Impact and possibly Novelty.
-  Export and Compatibility: As mentioned, we will support exporting to PPTX and PDF. This feature might seem logistical, but it's crucial for real-world usability – and the judges will value that we thought of it. It addresses the Design criterion in terms of user experience (the deliverable is immediately usable). We saw that being able to export to PowerPoint is a differentiator that even competitor comparisons talk about ("Tome does not support export to PowerPoint... Presentations.AI allows high-quality PowerPoint export"). We'll implement this using an open-source library or by generating an



OpenDocument/OOXML format. We will test the exported file on PowerPoint to ensure formatting is preserved (fonts, bullet indent, images placement, etc.). This feature will be highlighted in our demo – e.g., “Here’s the PPT file generated – now open it in PowerPoint to show it works.” It conveys a level of completeness that can impress judges (many hackathon projects stop at a web demo; we’ll hand over a tangible output).

-  Iterative Refinement & User Feedback Loop: Unlike a one-shot generation, our tool allows iteration. The user can adjust the outline and request changes (e.g., “make slide 3 simpler,” or “add a slide about X”). We might implement a simple chat-like interface for post-generation refinement, where the user types an instruction and the appropriate agent acts (similar to how one might prompt ChatGPT to modify its answer). This feature shows we prioritize user control and fine-tuning of results, which improves the Design score (a well-thought-out UX) and also addresses potential shortcomings of generative AI. It’s also a subtle way to demonstrate the model’s interactive abilities. Hackathon judges will appreciate that we didn’t build a black-box but rather an assistive tool that cooperates with the user. In the demo, for instance, we can show editing one slide’s text, or using a “regenerate” button. Having this fall-back for quality issues means we can still produce a great final presentation even if the initial generation had some quirks.
-  Use of Model’s Strengths (Context & Tools): We specifically exploit features of gpt-oss that are unique or enhanced compared to other models:
  - Large Context: The ability to feed a large document (e.g. 50 pages of text) is a game-changer. We will highlight this by possibly showing an example of generating slides from a lengthy source (like a sample whitepaper). The model’s “128k tokens context window” means it can “ingest large corpora” – we will push this to show something competitors with smaller contexts (like Llama-2 32k or GPT-3.5 16k) might miss. This directly shows off the Application of gpt-oss criterion (“showcase the strengths of the model uniquely” ).
  - Tool Usage / Function Calling: As noted, we’ll integrate at least one function (image search) and maybe a simple web search for facts if time allows. This shows that our project “applies the open models in an effective way” by using their agentic capabilities . For example, if the user’s document has outdated data, an agent could (optionally) do a quick web search for the latest stats and update a slide – demonstrating a live data integration. This is an ambitious stretch goal, so it might be something we mention (since the hackathon sponsors include vLLM who are interested in efficient tool use). Even if we only fully implement the image function, that itself is a proof of concept of model tool use.

All these features are chosen to maximize our hackathon scoring and create a compelling project. They also complement each other – fine-tuning improves base output, multi-agent structure improves reasoning and modularity, design features improve UX and polish, etc. The

end result will be a project that feels complete and impactful in just one month, demonstrating a high level of skill and creativity from our team.

## Timeline and Milestones (31 Days)

To execute this plan in ~31 days, we'll break the work into weekly sprints with clear milestones. Our team will likely work in parallel on model and app components, syncing often to integrate. Here's our proposed timeline:

- Week 1: Setup and Research (Days 1–7)

Milestone: Properly equipped to train and run the model; initial project skeleton.

- Acquire & Prep Models: Download gpt-oss-20b weights from Hugging Face and set up the environment (we'll use provided guides like "How to use gpt-oss with Transformers" for smooth setup). Test run a few prompts on the base model to gauge output style and performance on our hardware. If needed, install vLLM or Ollama for faster inference tests .
- Data Collection: Begin gathering training data for fine-tuning. Search for public slide decks with transcripts or speaker notes. Possible sources: SlideShare (if any text available), academic lecture slides vs papers (e.g., arXiv to slide conversions). Where direct data is lacking, use GPT-4 to generate synthetic pairs: e.g., feed it an article and ask for bullet-point slides. Aim to gather a few hundred examples to fine-tune on – focusing on quality over quantity given time. Also include different domains (tech, education, marketing) for generality.
- Design Brainstorm: Finalize the multi-agent design and how each agent will function. Decide on the interface architecture (likely a simple web app – maybe React or just HTML/JS + a Python backend via FastAPI or Flask to handle model calls). Sketch the UI screens (input form, outline view, slide viewer). Also define 3–5 theme styles we want to support (create basic CSS or PPT master slides for them).
- Task Assignment: If team has multiple members, split responsibilities: e.g. one focuses on model fine-tuning pipeline, one on front-end UI, one on building the PPT export logic, etc. Establish communication channels and version control (Git repo).

- Week 2: Prototype Development & Model Fine-Tuning (Days 8–14)

Milestone: Basic end-to-end generation with dummy data; fine-tuning job running.

- Model Fine-Tune Kickoff: Use the Hugging Face Transformers fine-tuning script (the hackathon resource “How to fine-tune gpt-oss using Transformers” will be handy). Start with gpt-oss-20b. We might do a LoRA fine-tuning to handle 20B on available GPUs (less memory intensive). Begin training on the dataset compiled in Week 1. This might take a few days to converge; we’ll monitor validation outputs (like see if bullet style is improving). If training is slow, consider smaller batches or use an online service with A100s for a few hours.
  - UI & Front-End: Build the input form page and results page. Implement file upload handling and prompt input. For now, the “Generate” button can call a placeholder API that returns a hardcoded example (to develop the front-end independently of model work). Design the outline edit interface (e.g., list of slide titles with an edit button). Also implement the slide viewer (initially static content). We’ll ensure the front-end is modular so we can plug in real data easily when ready.
  - Outline & Content Agents (Backend logic): Implement a basic version of the pipeline using the base model (since fine-tune may not be ready). For example, write a function that takes text input, uses gpt-oss (with a prompt template) to generate an outline (maybe using headings or JSON). Then another function to generate bullets for each outline point. This is a crude first cut, but by end of Week 2 we want to see a simple prompt result in some slide text. This will help us debug the logic and prompt formulations early. We’ll likely run this on a smaller model or short context to iterate quickly.
  - Image Integration (Prototype): Choose one image source to test. For now, maybe use Pexels API (free stock photos). Write a small function `get_image(keyword)` that returns a representative image URL for a keyword. We can test this separately (e.g. `get_image("climate change")` returns some relevant image). We won’t integrate into the model’s loop yet, but we lay the groundwork by obtaining API keys, etc.
  - Team Checkpoint: By the end of week 2, have a meeting to demo the prototype: e.g., we input “Test topic” and see an outline or some dummy slides on the UI. Also review a sample output from the fine-tuning (if completed epoch1) to gauge if we need more data or adjustments. Adjust plan if something is lagging (e.g., if fine-tune is delayed, plan to rely more on prompt engineering; if UI is behind, simplify features, etc.).
- Week 3: Integration and Feature Completion (Days 15–21)

Milestone: Full system integrated – model, agents, UI – producing actual presentations.

- Integrate Fine-Tuned Model: By early Week 3, we expect to have a fine-tuned model ready (or at least a first version). Integrate it into our pipeline: load the fine-tuned weights for inference. Compare outputs: we'll run the same test prompt through base vs fine-tuned model to ensure fine-tuning helped (e.g., fine-tuned should output nicer bullets). We can cite this improvement in our submission (showing effectiveness).
- Refine Multi-Agent Pipeline: Now wire up all agent steps in the backend. After outline generation, feed outline into content generation agent automatically. Incorporate the image agent: for each slide content, extract a few keywords (maybe using an algorithm or another mini prompt) and call the image API to get an image link. Insert that into the slide data structure. Develop simple heuristics: e.g., if a slide title contains " vs. " or numbers, maybe it's a comparison slide – choose a different layout or icon. These rules can elevate the design aspect.
- Theme and Layout Implementation: Apply the chosen theme to the generated content. In a web app, this could mean adding CSS classes for the background and text styles. For PowerPoint, use a template PPTX where we can fill in text placeholders with our content. We can generate an XML or use Python-pptx to create slides, set the background color, etc., according to theme. By mid-week, test exporting a sample PPTX and opening it to verify formatting (this might require some tweaking).
- UI Polish – Outline Editor and Slide Viewer: Hook up the real data to the front-end. After generation, show the outline step if we haven't already. Let the user edit slide titles (implement this by simply taking their edits and regenerating content for those slides – or simpler, incorporate their titles into the next step). Ensure the slide viewer displays the text and images nicely (maybe a carousel or a slide-by-slide view). Implement navigation controls if needed (next/prev slide).
- Multilingual & Other Options: Add the ability to choose language in the UI, and make sure our generation prompts respect it (e.g., include "in Spanish" in the system prompt if Spanish is selected). Quick test with a known text in another language to see if model outputs that language correctly. Also, implement the slide count option: perhaps adjust the outline agent prompt to generate exactly N slides if the user requested a number.
- Testing: Start testing the integrated system with various inputs:
  1. Short prompt (few words) vs long document.
  2. Technical content vs general topic vs narrative content (to see how it handles different styles).

3. Different themes (ensure theme switching doesn't break layout).
  4. Edge cases: no images found for a weird topic, extremely large input (how is performance/memory).
  5. If any failure or low-quality output is observed, refine prompts or agent logic accordingly. We iterate now while there's still time to fix issues.
- Week 4: Optimization, Polish, and Demo Prep (Days 22–31)

Milestone: A polished, competition-ready project with documentation and demo.

- Performance Optimizations: Running a 20B model with multiple calls per generation might be slow. In week 4, we'll optimize:
  1. Possibly use vLLM for serving the model in memory efficiently (the hackathon sponsor vLLM provides a library for fast concurrent inference). This could speed up multi-agent calls by not reloading the model each time. If setup is complex, an alternative is to cache results for repeated runs or use a smaller model for non-critical agents (e.g., maybe use a 7B open model for the critique agent if fine).
  2. We may quantize the model further (if not using the provided MXFP4 quantization) to make it faster on our GPU. The HF blog notes the model is already quantized and can use FlashAttention etc. – we'll ensure those optimizations (like running on a machine with a recent GPU that supports it).
  3. If generation is still too slow for a live demo, we'll prepare a cached example to show in the video (ensuring the video is smooth). But we'll also mention that with appropriate hardware (e.g., an NVIDIA 5090 GPU), it would be much faster – a nod to NVIDIA's sponsor interests.
- Full Run-through and UX Fine-tuning: Do complete test runs as if we are the end-user. Aim to fix any remaining rough edges:
  1. Ensure the UI is clear (add instructions or placeholder text like "Enter topic...").
  2. Verify that after generation, the user can easily navigate the slides and trigger any re-generation for slide content if needed.
  3. Add a "Download PPT" and "Download PDF" button and test them.

4. Implement any missing safety net (e.g., if the model somehow outputs a disallowed phrase, we filter or warn – though our domain is mostly safe).
  5. If possible, integrate a content fact-check step for extra impressiveness: e.g., use the model to double-check if any factual claim was made and perhaps append source info. (This is very advanced and likely skipped due to time, but even an attempt could be mentioned.)
- Visual Appeal: As this is the final week, we'll spend time on the look and feel. Create a nice logo or landing screen for the app (small touches can make it memorable). Perhaps name the project (something catchy like "SlideWhiz" or "DeckGenie") and include that in the UI. This helps with presentation and showing branding (plus looks good on our resume/project page).
  - Prepare Presentation & Video: This is crucial for hackathon submission. We'll create a short pitch deck (maybe ironically using our own tool to generate the first draft!) highlighting:
    1. The problem (making presentations is time-consuming or requires expensive tools).
    2. Our solution (multi-agent AI presenter) with its features.
    3. Quick demo of it in action.
    4. Why it's innovative (fine-tune, open-source, etc.).
    5. Impact potential (time saved, wider access, etc.).

We'll then record a demo video (likely 2-3 minutes as required). The video will show the app interface: we'll walk through inputting a prompt/document, show the outline edit briefly, then show the final slides and maybe scroll through them. We'll point out how images were added and mention the model behind the scenes. We'll also explicitly call out how we used the open model and that it's running locally (e.g., show a terminal with it running to emphasize offline). The video should end with a strong note: e.g., "In 5 minutes, our AI co-pilot created a presentation that normally takes hours – imagine what this means for productivity and accessibility!" – leaving judges with a clear impression of impact.

- Devpost Submission Materials: Finalize the project write-up for Devpost. We'll include the required details, making sure to reference the judging criteria: e.g., Application of Model – describe fine-tuning and agent use; Design – describe our UI and user testing; Impact – include a hypothetical case study (like a teacher

using it to prepare lessons, saving X hours weekly); Novelty – compare vs other tools (we can even reference that “no existing solution combines fine-tuned open models with multi-agent orchestration for presentations”, which we have from our research). We will also acknowledge sponsors (mention using Hugging Face for model hosting, NVIDIA GPUs for running the model, etc., which subtly addresses those audiences).

- Buffer and Contingency: We'll reserve the last day or two for any unexpected issues or last-minute tweaks. Also, we plan to practice our pitch (if there's a live judging or Q&A) – be ready to answer technical questions (fine-tune process, how multi-agent works under the hood, etc.) and discuss future scope (like integrating this into MS Teams or adding voice-over generation as next steps, to show we have vision).

By following this schedule, we ensure we have a working prototype early and a refined product by the deadline. Regular testing and integration will mitigate the risk of last-minute failures. The timeline also shows our ability to plan and execute efficiently – something that reflects well both in the competition and on our resumes.

## Competitive Analysis & Differentiation

We expect top engineers in this hackathon, so it's likely other teams will also build impressive projects with gpt-oss. Here's why our project will stand out:

- Unique Combination of Techniques: Some teams might fine-tune the model for a specific domain (education tutor, medical Q&A, etc.), and others might build agent-based apps (like a robot or a coder assistant). Our project does both – a fine-tuned model and an agent-based system – in a domain that is practical and demo-friendly. It's a new twist on the familiar “AI makes slides” idea, which hits the Wildcard (“unexpected use”) note while still being broadly useful.
- High Real-World Relevance: Every knowledge worker has made slides; the utility is immediately clear. This could give us an edge on Potential Impact, as judges can instantly grasp how this could save time for many people. It's not a niche or whimsical demo – it's something that could be a product. In fact, sponsors like OpenAI and HF might see potential to showcase this as a success story of their open models enabling real productivity tools.
- Polish and Presentation: We are putting significant effort into the UI/UX and output quality. Many hackathon projects focus on the technical and have bare-bones interfaces; we'll deliver a professional-looking app and outputs. From our research, we know content quality and design matter (users/judges will notice if slides look sloppy or generic). By fine-tuning the model and refining the content, we avoid the pitfalls of

generic AI output . And by including visuals and good formatting, our demo will simply look more impressive than a text-only console demo.

- **Open Source and Resume-worthy Engineering:** We will open source our code and (if allowed) our fine-tuned model. This demonstrates commitment to the community (which HF and OpenAI judges appreciate) and also means our project can live beyond the hackathon. From a resume perspective, this is great: we can show recruiters a GitHub repo of a working application, with documentation, and possibly a HuggingFace model link they can try. It's tangible proof of our skills in ML engineering, full-stack dev, and project management. Few hackathon projects achieve a level where they can be shared broadly; we aim to reach that level.
- **Alignment with Multiple Prize Categories:** Strategically, while our primary target is “Most Useful Fine-Tune” (OpenAI), our project ticks boxes in other categories too:
  - **Best Overall:** If we execute perfectly and have a bit of luck, we could contend for overall winner by scoring high in all criteria.
  - **Best Local Agent:** Our application essentially is a local AI agent that works without internet (we can demonstrate it fully offline). That's the spirit of this category (sponsored by NVIDIA) . Even though we didn't focus on robotics or hardware, just being a useful offline agent with heavy GPU usage could put us in the conversation.
  - **For Humanity:** If we emphasize educational use (e.g., helping under-resourced schools create materials quickly, or non-profits preparing info decks), we can claim social benefit. We might not specifically aim for this prize, but it doesn't hurt to mention how our tool could help educators or community workshops, etc.
  - We likely won't fit Weirdest Hardware or Best in Robotics (those require physical components we aren't doing), and Wildcard is broad – but our idea is less “weird” and more “useful,” which is fine.

By positioning our project at the intersection of technical innovation and practical utility, we maximize our odds that at least one set of judges will champion it.

## **Resume & Career Highlights**

Beyond winning the hackathon, we also see this project as a stellar addition to our professional portfolio. Here's how it shines on a resume or in interviews:

- **Cutting-Edge Technologies:** We worked with OpenAI's latest open-source LLM (gpt-oss released 2025) and implemented fine-tuning, large-context processing, and AI agent



orchestration. These are hot topics in AI – demonstrating experience in them will catch recruiters' eyes. We can truthfully say we built a mini-ChatGPT-like system but for a specialized task, including multi-model integration (LLM + image generation). This signals that we're not just users of AI APIs, but we deeply understand and can extend AI models.

- **End-to-End Project Delivery:** On our resume, this project shows full-stack capability: front-end web development, back-end server logic, and ML model training. It's a great example of bridging the gap between research and user-facing product. We can highlight specific accomplishments, e.g. "Fine-tuned a 20B-parameter language model to improve slide summarization by X%, and developed a React/Flask application for interactive use." Having metrics (like generation speed, or how much editing it reduced compared to normal) can also quantify the impact.
- **Teamwork and Agile Execution:** We delivered this in 31 days, which implies we can work under tight deadlines, divide tasks, and iterate quickly – all valuable in industry. If we lead the team, we can mention leadership; if we each took distinct roles (model specialist, UI developer, etc.), we highlight those contributions.
- **Open Source Contribution:** By releasing our project code or model, we add to our public profile. Employers often appreciate candidates who contribute to open-source projects. It also allows others to use or cite our project, increasing our visibility. We might even write a Medium article about our project (which, given the timely subject, could gain traction). This kind of initiative demonstrates passion and communication skills.
- **Hackathon Achievement:** If we win or place in the hackathon, that is a big resume boost in itself ("Winner of OpenAI's Open Model Hackathon 2025 – selected top out of 2000+ participants" sounds fantastic). Even if not, being a finalist or simply participating with a solid project is worth noting, especially given the prestigious sponsors involved. It shows we engage with the developer community and can hold our own in global competitions.
- **Demo-ready Project:** Post-hackathon, we'll have a polished demo that we can show in interviews or attach to portfolios. Many candidates talk about projects, but having a live demo where you can say "Type a topic, and watch what my AI builds" can leave a strong impression. It's the kind of project that can spark interesting discussions with interviewers (about how we did X or overcame Y challenge).

In summary, this project isn't just a hackathon entry – it's a demonstration of our abilities at the forefront of AI and software engineering. We will leverage it to open doors in career opportunities, whether that's in AI research, product development, or startup endeavors. The fact that it addresses a real problem and has a potentially wide user base also suggests entrepreneurial potential (it could be turned into a startup or a product feature). So on a resume, we can emphasize both the innovation and the real-world relevance.

## Conclusion: Maximizing Our Odds of Victory

To conclude, our multi-agent presentation generator & beautifier project is carefully crafted to excel in the hackathon and beyond. By thoroughly researching the market and sponsor goals, we've ensured our idea is both novel and aligned with what judges seek. We will demonstrate:

- Superior use of gpt-oss: Fine-tuning and deploying it in a way that showcases its strengths (huge context, agentic tasks) that other projects likely won't . This directly addresses the hackathon's core challenge of applying these models in groundbreaking ways .
- Excellent design and polish: A smooth user experience with an interface that balances AI power with human control . We're not just showing a tech demo; we're showing a prototype of a product with real users in mind.
- Impact and usefulness: A solution that could save countless hours and empower those without design skills or big budgets. Our narrative will emphasize how this AI assistant can level the playing field (for example, a small startup can now create pitch decks as slick as a big company's, thanks to our tool). Judges considering impact will see that our project, if continued, could be used by many professionals, educators, and students around the world .
- Creativity and innovation: Combining multi-agent coordination with a fine-tuned model is unique. Even if others build slide generators, ours will have an extra "brain" behind it (multiple brains, in fact). We improve upon existing ideas (we've shown we know what's out there and how we're better), fulfilling the novelty criterion well .

We have also built in contingency and adaptability – if something doesn't work perfectly, our iterative refinement approach and user-in-the-loop design means we can still produce a great end result for the demo. This resilience will help ensure we can deliver on the day of submission.

Finally, our plan to present and pitch the project is as important as the project itself. We will make sure to tell a compelling story in our submission: one that highlights the technical achievement (training a model, orchestrating agents) and the practical achievement (making slide creation effortless). By clearly citing how we met each judging criterion and by delivering a visually engaging demo, we'll make it easy for judges to vote for us.

In essence, we aim to redefine what's possible with open models in the productivity space, echoing the hackathon's call to "redefine what open models can do" . If we execute this plan, our project will not only have a high chance of winning, but it will also be something we're proud to showcase long after the hackathon.

With 31 days of hard work and smart planning, we are confident that our multi-agent AI presentation builder can impress the judges and rise to the top – and in the process, become a shining example of our capabilities for our careers moving forward. We're excited to bring this idea to life and potentially secure a victory in the OpenAI Open Model Hackathon 2025!

Sources:

- OpenAI Open Model Hackathon rules and categories
- Hackathon prompt and sponsor goals
- Presenton open-source AI presentation tool features
- Presentations.AI vs Tome feature comparison
- HubSpot review of AI presentation makers (noting Tome's limitations)
- OpenAI GPT-OSS model capabilities (context window, tool use)
- Hugging Face blog on GPT-OSS (model size and usage) and OpenAI's announcement .