Lab Session 1: Generative AI Foundations, AI Chats, and API Integration

Context and lab intent

This lab is designed for final-year engineering students in an AI-focused program that emphasizes hands-on training, project-based learning, open-source exposure, and industry-oriented skill development.[web:11][page:1] The department highlights Generative AI, project-based pedagogy, modern tool usage, communication, and applied AI problem solving, which makes a practical lab on AI chats and APIs a strong fit for the program outcomes.[page:1]

This first lab covers Sessions 1-3 and introduces students to Generative AI through direct use of AI chat systems and developer-facing API integration. The sequence moves from understanding prompt-response behavior, to structured prompt design, to building simple applications that call an LLM through an API.

Lab overview

- Lab block: Sessions 1-3
- Suggested duration: 3 hours per session, 9 hours total
- Theme: Generative AI using ChatGPT for conversational exploration and Groq for API-based integration
- Pedagogy: guided exploration, short exercises, mini-builds, reflection, and output evaluation
- Delivery mode: individual hands-on work with short pair discussions

Learning objectives

By the end of this lab, students should be able to:

- Explain what Generative AI is, how large language models are used in chat systems, and where such systems are useful in real applications.
- Distinguish between good and poor prompts by comparing instruction clarity, context quality, constraints, and output format.
- Use an AI chat system to perform ideation, summarization, transformation, extraction, and structured reasoning tasks.
- Call a hosted LLM through an API using a simple script or REST client.
- Interpret request payloads, API keys, model selection, system prompts, user prompts, tokens, temperature, and output structure.
- Build a small working prototype that combines user input, an LLM call, and formatted output.

- Evaluate responses for relevance, hallucination risk, formatting quality, and task completion.
- Document findings clearly in a technical lab submission, aligning with communication and modern tool usage outcomes emphasized by the program.[page:1]

Prerequisites

Students should have:

- Basic familiarity with Python or JavaScript.
- A laptop with internet access.
- A ChatGPT account for chat-based exercises.
- A Groq API key for API-based exercises.
- A code editor such as VS Code and either Python installed locally or Node.js installed locally.
- Postman, Bruno, or curl for testing HTTP APIs.

Software and setup

Install or verify the following before the lab:

1. VS Code or another code editor.
2. Python 3.10+ with `requests` installed, or Node.js 18+ with `fetch` support.
3. Postman or Bruno for API testing.
4. A Groq API key stored securely in an environment variable.

Recommended environment variable setup

Windows PowerShell

```
$env:GROQ_API_KEY="your_api_key_here"
```

macOS/Linux

```
export GROQ_API_KEY="your_api_key_here"
```

Session structure

Session	Focus	Core outcome
1	Understanding Generative AI and AI chats	Students learn prompt behavior and use ChatGPT effectively for different task patterns

Session	Focus	Core outcome
2	Prompt engineering and response evaluation	Students create high-quality prompts and evaluate outputs systematically
3	API integration with Groq	Students build a working script that sends prompts and processes responses

Session 1

Understanding Generative AI and AI chats

Session objective

Students should understand what Generative AI does, how chat interfaces expose LLM capability, and how different prompt styles change output quality.

Concept coverage

- What Generative AI means in practice
- Difference between predictive AI and generative AI
- What an LLM does during text generation
- Typical use cases: summarization, Q&A, ideation, rewriting, extraction, tutoring, content drafting
- Limitations: hallucinations, outdated knowledge, prompt sensitivity, overconfidence, privacy concerns

Instructor demonstration

The instructor should show the same task in ChatGPT using four prompt styles:

- Vague prompt
- Clear prompt
- Prompt with role and context
- Prompt with output format constraints

Example task:

```
Summarize the impact of AI in healthcare for a college presentation.
```

Then improve it:

```
You are a teaching assistant for final-year engineering students. Summarize the impac
```

Exercise 1.1: Explore basic prompting

Task

Students must try five prompt categories in ChatGPT:

- Explain a concept
- Summarize a paragraph
- Rewrite text for a different audience
- Generate ideas
- Extract structured information

Step-by-step instructions

1. Open ChatGPT.
2. Create a fresh chat named `GenAI Lab Session 1`.
3. Enter one prompt from each category.
4. Copy each response into a lab notes document.
5. For each response, write two observations: what worked well, what was missing.

Example prompts

```
Explain retrieval-augmented generation to a final-year engineering student in simple
```

```
Summarize the following text in 4 bullet points: [paste a paragraph from a technical
```

```
Rewrite this email in a professional tone: [paste short raw email text].
```

```
Give 10 startup ideas that use generative AI in education.
```

```
Extract the following into JSON with keys: name, skills, project_title, tools_used.
```

Expected outcome

Students will observe that the same AI system can perform multiple task types, but output quality depends strongly on prompt specificity.

Exercise 1.2: Compare weak and strong prompts

Task

Students compare outputs from weak prompts and improved prompts for the same problem.

Step-by-step instructions

1. Choose one task such as summarization or idea generation.
2. Write a weak prompt with minimal detail.
3. Write a strong prompt with role, context, constraints, and desired format.
4. Run both prompts in ChatGPT.
5. Compare outputs using the following criteria: clarity, completeness, format, factual caution, usefulness.
6. Record the comparison in a 5-row table.

Comparison template

Criteria	Weak prompt output	Strong prompt output
Clarity		
Relevance		
Format		
Specificity		
Usefulness		

Expected outcome

Students will understand that prompt design functions like interface design: better instructions usually produce better task completion.

Exercise 1.3: ChatGPT as a study assistant

Task

Students use ChatGPT to learn a topic from the course and check whether iterative questioning improves understanding.

Step-by-step instructions

1. Pick one technical topic, such as transformers, embeddings, vector databases, or AI agents.
2. Ask for a beginner explanation.
3. Ask for an analogy.

4. Ask for a more technical explanation.
5. Ask for a quiz with five questions.
6. Answer the quiz yourself.
7. Ask ChatGPT to evaluate your answers.
8. Write a short reflection: did iterative dialogue improve learning quality?

Expected outcome

Students will see that chat interfaces are useful not only for answers but also for guided learning, revision, and self-assessment.

Deliverable for Session 1

- One lab note file containing prompts, outputs, comparisons, and a 200-word reflection.

Session 2

Prompt engineering and response evaluation

Session objective

Students should learn how to systematically engineer prompts and evaluate output quality for practical tasks.

Concept coverage

- Prompt components: role, task, context, examples, constraints, format
- Zero-shot, one-shot, and few-shot prompting
- Structured output generation
- Error analysis and response evaluation
- Hallucination detection and verification habits

Instructor demonstration

The instructor should demonstrate how the same task improves using:

- Zero-shot prompt
- One-shot prompt
- Few-shot prompt
- Prompt with explicit JSON output

Exercise 2.1: Prompt pattern design

Task

Students redesign weak prompts into production-style prompts.

Step-by-step instructions

1. Take any three weak prompts from Session 1.
2. Rewrite each prompt using the structure below.

Prompt design template

```
Role:
Task:
Context:
Constraints:
Output format:
```

3. Run the new prompts in ChatGPT.
4. Save both old and new responses.
5. Write one paragraph for each prompt explaining the improvement.

Expected outcome

Students will learn reusable prompt templates rather than relying on trial and error.

Exercise 2.2: Generate structured output

Task

Students force the model to produce clean machine-readable output.

Step-by-step instructions

1. Ask ChatGPT to produce output as JSON for a fixed use case.
2. Paste the output into a JSON validator.
3. If the output fails, refine the prompt and try again.
4. Repeat until valid JSON is produced.

Example use case

```
You are an assistant that extracts internship candidate data.
From the text below, return valid JSON only with fields: full_name, college, branch,
```

Expected outcome

Students will understand why structured prompting matters when LLMs are used inside applications.

Exercise 2.3: Response quality rubric

Task

Students evaluate AI outputs using a rubric.

Step-by-step instructions

1. The instructor provides a task, such as writing a short technical explainer on AI agents.
2. Students create one prompt each and obtain outputs.
3. Students exchange outputs in pairs.
4. Each student scores the partner's output from 1 to 5 on these criteria:
    - Accuracy confidence
    - Task completion
    - Clarity
    - Structure
    - Hallucination risk
5. Students revise the prompt and generate a second version.
6. Students compare the first and second versions.

Expected outcome

Students will understand that prompt engineering is iterative and that evaluation is central to reliable AI use.

Exercise 2.4: Safety and responsible usage mini-task

Task

Students identify which tasks are safe to delegate to AI and which need human verification.

Step-by-step instructions

1. Review five sample scenarios provided by the instructor: medical advice, code generation, exam explanation, legal interpretation, email drafting.
2. Mark each one as low, medium, or high verification requirement.
3. Write one sentence explaining the reason.
4. Discuss in small groups for five minutes.

Expected outcome

Students will connect technical capability with responsible use, which aligns with the program's emphasis on ethics and sustainable professional practice.[page:1]

Deliverable for Session 2

- A prompt design worksheet containing three improved prompts, one JSON extraction task, one rubric-based evaluation table, and a short responsible-AI reflection.

Session 3

API integration with Groq

Session objective

Students should move from using AI as end users to using AI as developers by calling an LLM API from code and handling the response in an application flow.

Concept coverage

- What an AI API is
- API key handling
- Request and response structure
- System prompt and user prompt in chat-completion APIs
- Model selection and inference parameters
- Basic error handling
- Parsing returned text for use in applications

The Groq ecosystem exposes fast inference for open models and is commonly integrated through chat-completion style APIs, while n8n workflows can connect model, memory, and external tools in later agent-based labs.[web:3][web:8][web:10]

Instructor demonstration

The instructor should show:

- A request in Postman or curl
- The same request in Python or JavaScript
- How changing temperature and prompt changes the output
- How to read the assistant message from the JSON response

Exercise 3.1: First API call using curl or Postman

Task

Students send a simple prompt to a Groq chat model and inspect the raw JSON response.

Step-by-step instructions

1. Open Postman, Bruno, or terminal.
2. Create a POST request to the Groq chat completions endpoint.
3. Add the `Authorization` header using the API key.
4. Add the `Content-Type: application/json` header.
5. Use a body similar to the example below.
6. Send the request.
7. Inspect the returned JSON and identify the generated message.

Sample request body

```
{
  "model": "llama3-8b-8192",
  "messages": [
    {"role": "system", "content": "You are a helpful teaching assistant for engineer
    {"role": "user", "content": "Explain prompt engineering in 120 words with one exa
  ],
  "temperature": 0.3
}
```

Expected outcome

Students will understand the anatomy of a real API request and locate useful fields in the response JSON.

Exercise 3.2: Python integration

Task

Students build a Python script that takes user input, sends it to Groq, and prints the response.

Step-by-step instructions

1. Create a file named `groq_chat.py`.
2. Paste the code below.
3. Replace the environment variable setup if needed.
4. Run the script.
5. Ask at least three different questions.
6. Save one screenshot of successful execution.

Sample Python code

```
import os
import requests

api_key = os.getenv("GROQ_API_KEY")
url = "https://api.groq.com/openai/v1/chat/completions"

payload = {
    "model": "llama3-8b-8192",
    "messages": [
        {"role": "system", "content": "You are a helpful AI tutor for final-year engi
        {"role": "user", "content": input("Enter your prompt: ")}
    ],
    "temperature": 0.4
}

headers = {
    "Authorization": f"Bearer {api_key}",
    "Content-Type": "application/json"
}

response = requests.post(url, json=payload, headers=headers, timeout=30)
response.raise_for_status()
result = response.json()
print(result["choices"][0]["message"]["content"])
```

Expected outcome

Students will create a working command-line GenAI client.

Exercise 3.3: Build a mini application

Task

Students build one of the following mini tools:

- Text summarizer
- Resume bullet improver
- FAQ generator from notes
- Email drafting assistant
- Study quiz generator

Step-by-step instructions

1. Choose one mini-tool idea.
2. Define the user input and expected output.
3. Write a clear system prompt for the chosen task.
4. Create a script that accepts user input.
5. Send the input to Groq.
6. Display the formatted output cleanly.
7. Test the tool with three inputs.
8. Write a short note on where the tool succeeds and where human review is needed.

Suggested extension

Students who finish early can add one more feature:

- Save output to a text file.
- Add a menu of prompt types.
- Add input validation.
- Add markdown output formatting.

Expected outcome

Students will see how an LLM becomes part of an application, not just a chat interface.

Exercise 3.4: Parameter experiment

Task

Students compare model behavior by changing one parameter at a time.

Step-by-step instructions

1. Keep the prompt fixed.
2. Run the same request with temperature 0.1, 0.5, and 0.9.
3. Compare the outputs for determinism, creativity, and consistency.
4. Record findings in a table.

Expected outcome

Students will understand why model configuration matters in production applications.

Deliverable for Session 3

- Source code for one working mini-tool.
- Screenshot or terminal log of successful API calls.
- One-page note on parameter experiments and practical observations.

Assessment plan

Component	Weightage
Session 1 prompt exploration and reflection	20%
Session 2 prompt engineering worksheet	25%
Session 3 API exercises	25%
Mini application quality	20%
Clarity of documentation and explanation	10%

Evaluation rubric

Criterion	Excellent	Good	Needs improvement
Conceptual understanding	Explains GenAI, prompts, and API flow clearly with examples	Understands most concepts with minor gaps	Uses tools mechanically without clear understanding
Prompt quality	Uses role, context, constraints, and format effectively	Prompts are clear but inconsistent	Prompts are vague or incomplete
API integration	Makes correct request, handles response, and tests variations	Basic integration works with limited analysis	Request structure or response handling is weak
Application thinking	Builds a useful mini-tool with sensible design choices	Working tool but limited usability	Tool is incomplete or poorly aligned to task
Critical evaluation	Identifies errors, limitations, and verification needs	Some reflection present	Minimal or no evaluation

Expected cumulative outcomes

At the end of Sessions 1-3, students should be able to:

- Use AI chat tools productively for learning, drafting, and structured task execution.
- Write better prompts by intentionally controlling role, context, and output structure.
- Integrate a hosted LLM into a small application using an API.
- Explain the difference between using AI interactively and integrating AI programmatically.
- Recognize the need for human verification in high-stakes or ambiguous tasks.

Suggested instructor notes

- Begin each session with a 15-minute demonstration and end with a 15-minute review.
- Use one common theme across all exercises, such as student productivity, interview preparation, or academic support, so students can connect the tasks.
- Ask students to maintain a prompt journal; this helps them observe iteration quality over time.
- Encourage pair review for prompt evaluation, because students learn quickly by comparing outputs and prompting styles.
- Keep the mini-project intentionally small so the focus remains on understanding concepts completely rather than rushing into complex software engineering.

Materials to provide students

- One starter handout with API endpoint details, sample payloads, and setup steps.
- One sample text file for summarization and extraction tasks.
- One prompt worksheet template.
- One evaluation rubric template.
- One starter code template in Python and optionally JavaScript.

Submission checklist

Students must submit:

- Prompt exploration notes from Session 1
- Prompt engineering worksheet from Session 2
- Groq API test evidence from Session 3
- Source code of the mini application
- Short reflection on strengths, limitations, and verification needs

Expected final lab outcome

By the end of this first lab block, each student should have a practical understanding of Generative AI, know how AI chat systems differ from API integrations, and possess at least one working prototype that uses an LLM for a real task. This directly supports a curriculum that values project-based learning, modern AI tools, industry relevance, communication, and ethical application of AI systems.[page:1]