PROMPT ENGINEERING

Course Code	23SA8756	Year	IV	Semester	I
Course Category	Skill Enhancement Course	Branch	IT	Course Type	Practical
Credits	2	L-T-P	0-1-2	Prerequisites	Artificial Intelligence, Python Programming,
Continuous Internal Evaluation :	30	Semester End Evaluation:	70	Total Marks:	100

Course Outcomes		
Upon successful completion of the course, the student will be able to:		
CO1	Apply prompt engineering basics and fix common prompt problems	L3
CO2	Make use of advanced techniques like few-shot, role-based, and negative prompting	L3
CO3	Organize structured output and make use of step-by-step reasoning	L3
CO4	Build a simple RAG pipeline using LangChain	L3
CO5	Apply prompts for safety, ethics, and use LLM-as-Judge	L3

Contribution of Course Outcomes towards achievement of Program Outcomes & Strength of correlations (3: High, 2: Medium, 1: Low)

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	3	3	3	3				3		3	3	3
CO2	3	3	3	3	3				3		3	3	3
CO3	3	3	3	3	3				3		3	3	3
CO4	3	3	3	3	3				3	3	3	3	3
CO5	3	3	3	3	3	3	3	3	3	3	3	3	3

Syllabus

Unit No	Contents	Mapped CO
I	**Unit I: Foundations of Prompt Engineering:** Definition of prompt engineering, Distinction between prompt engineering and model fine-tuning, Motivation and benefits of prompt engineering, Core principles of effective prompt design, Anatomy of a prompt, Setting up the Python environment for LLM interaction, Iterative prompting lifecycle, and Common prompt pitfalls and remediation **Lab Experiments:** 1. Environment & Connectivity: Install required packages (e.g., transformers, openai); securely configure the API key; run a simple "Hello, world" prompt to verify model access. 2. Baseline vs. Enhanced Prompts: Execute a naïve prompt ("Write a one-paragraph bio of Ada Lovelace.") and an enhanced prompt that adds role framing, specificity, and explicit format instructions; compare both outputs for relevance, completeness, and style. 3. Iterative Refinement on a Simple Task: Summarize the plot of the Shakespearean play Romeo and Juliet in two sentences through three rounds of prompt tweaking: a. Minimal instruction.	CO1

	b. Addition of length and style constraints c. Specification of key content elements (setting and theme) Document how each iteration changes and improves the result. 4. Diagnosing Prompt Failures & Edge Cases: Craft a vague or contradictory prompt; analyse the failure mode (ambiguity, missing context, or format errors); refine the prompt by adding examples or clarifying instructions.	
II	**Unit II: Advanced Prompt Patterns & Techniques:** Enhanced prompt anatomy: contextual detail and explicit output specifications, Few-shot in-context prompting, Prompt structuring and template design, Role-based prompting to establish personas/ or system behaviour, Negative prompting to filter or suppress undesired content, Constraint specification and instruction enforcement (e.g., length, format), Iterative prompt refinement and optimization **Lab Experiments:** 1. Few-Shot vs. Zero-Shot Comparison: Design and execute a zero-shot prompt and a few-shot prompt (with 2–3 exemplar input-output pairs) for a chosen text task (e.g., sentiment classification or translation); compare outputs for accuracy, consistency, and adherence to examples. 2. Role-Based & Negative Prompting: Craft a role-based prompt to establish a specific persona (e.g., "You are a financial advisor..."); then create a negative prompt to suppress undesired content (e.g., "Do not mention any brand names"); evaluate how each influences the model's response. 3. Constraint Specification & Iterative Refinement: Select an open-ended task (e.g., summarizing a technical article); issue a basic prompt; identify failures in length or format; refine the prompt by adding explicit constraints (word count, bullet format, etc.); document improvements over two refinement cycles.	**CO2**
III	**Unit III: Structured Output & Reasoning Techniques:** Importance of structured outputs for real-world applications, Prompting for specific formats (lists, tables, Markdown), Generating valid JSON and YAML via explicit instructions, Eliciting chain-of-thought reasoning in zero-shot prompts, Decomposing complex tasks into manageable sub-tasks **Lab Experiments:** 1. Structured Format Prompting: Instruct the model to output information as bullet lists and Markdown tables (e.g., "List three benefits of daily exercise in a Markdown table with columns 'Benefit' and 'Description.'"); verify the output matches the requested structure. 2. JSON/YAML Generation: Provide a brief dataset description (e.g., three books with title, author, publication year) and prompt the model to produce valid JSON or YAML; use a parser to validate syntax and refine the prompt if errors occur. 3. Chain-of-Thought & Task Decomposition: Present a multi-step problem (e.g., a logic puzzle) and apply zero-shot CoT prompting (e.g., "Let's think step by step. Explain your reasoning before the final answer."); separately, decompose the problem into sequential sub-questions, collect partial answers, combine them, and compare accuracy against a direct-answer baseline.	**CO3**
	Unit IV: Retrieval-Augmented Generation &LangChain Workflows: Limitations of LLM internal knowledge, Need for external data sources, Introduction to Retrieval-Augmented Generation (RAG), Overview of RAG	**CO4**

IV	architecture (indexing vs. retrieval + generation), Getting started with LangChain for LLM applications, Basics of LangChain Expression Language (LCEL), Simplified indexing pipeline: document loading & text splitting, Fundamentals of embeddings and vector stores, Building a basic retrieval-generation pipeline with an LCEL chain **Lab Experiments:** 1. Building a Simple LCEL Chain: Create a minimal LCEL script that accepts a fixed instruction (e.g., "Summarize this text: ..."), passes it to an LLM, and prints the result; verify end-to-end execution. 2. Basic Data Indexing for RAG: Load a small collection of documents; split into uniform chunks (e.g., 200 tokens); generate embeddings for each chunk; store them in an in-memory vector store; inspect for consistency. 3. Constructing & Running a Basic RAG Chain: Build a pipeline that: a. Receives a user query b. Retrieves the top-k relevant chunks c. Constructs a combined prompt with context + query d. Send it to the LLM e. Returns the answer Test with sample queries and compare factual accuracy against a prompt without retrieval.	
V	**Unit V: Agents, Multimodal AI & Ethical Evaluation:** Introduction to LLM agents and their basic architecture, Overview of multimodal AI models (VLMs), Prompting for text-to- image generation and image understanding, Importance of prompt evaluation beyond subjective judgment, Manual evaluation techniques (heuristic checks for accuracy, relevance, format), Introduction to "LLM-as-Judge" for automated evaluation, Security considerations (prompt injection, sensitive-information risks), Prompt-based mitigation strategies for safety and robustness, Ethical concerns (bias, misinformation, data privacy), Brief exploration of UI frameworks (Streamlit/Gradio) for deploying prompt-driven apps, Adapting to the evolving nature of prompt engineering through continuous learning **Lab Experiments:** 1. Building a Simple LLM Agent: Register a tool (e.g., a calculator function) and craft prompts that instruct the agent to invoke it when required; implement using LangChain or a function-calling API; test on queries requiring tool execution. 2. Multimodal Prompting Exploration: Generate images from detailed text prompts; feed one generated image into an image-understanding model or API with an appropriate prompt; compare the returned caption to the original prompt to evaluate alignment. 3. Prompt Evaluation & Ethics Workshop: a. Select two existing prompts and generate multiple outputs; apply manual heuristic checks for accuracy, relevance, and format compliance. b. Use an "LLM-as-Judge" prompt (e.g., "Rate these outputs on a scale of 1–5 for clarity and correctness.") to automate evaluation. c. Design a prompt- injection test (e.g., "Ignore previous instructions..."), observe the response, then refine system prompts to mitigate the vulnerability.	**CO5**

Learning Resources
TextBooks
1. John Berryman, Albert Ziegler, “Prompt Engineering for LLMs”, O'Reilly Publications, 2024. 2. Shivendra Srivastava, Naresh Vurukonda, “Prompt Engineering for AI Systems”, Manning Publications, 2025.
References
1. Anand Nayyar, Ajantha Devi Vairamani, Kuldeep Kaswan, “Mastering Prompt Engineering: Deep Insights for Optimizing Large Language Models”, Morgan Kaufmann (Elsevier), 2025.
E-Resources and other Digital Material
1. https://www.promptingguide.ai/ 2. https://cloud.google.com/discover/what-is-prompt-engineering 3. https://www.ibm.com/think/topics/prompt-engineering 4. https://swayam-plus.swayam2.ac.in/courses/course-details?id=P_IITMPR_34