EFSP Auxiliary ESL Curriculum

AI Development English

Instructor guide for high-level ESL learners working in AI engineering, research, product, safety, and deployment

Audience: instructors, coaches, technical English trainers, and learning partners

Focus: high-level professional English for AI development teams, including technical vocabulary, engineering discussion patterns, research/product tradeoffs, evaluation language, risk communication, and realistic workplace dialogue.

Designed for advanced ESL learners who already work with software, data, or AI systems and need field-specific fluency rather than basic grammar instruction.

Teaching stance: AI language changes quickly. Teach learners to ask precise clarification questions, define terms in context, and distinguish research claims, implementation details, benchmark results, and product promises.

Purpose and Teaching Position

This EFSP curriculum is for high-level ESL learners who work in AI development or closely adjacent roles: engineers, data scientists, researchers, product managers, technical program managers, QA specialists, solutions engineers, and AI safety or governance staff.

The course is not an introduction to programming or machine learning. It is a professional English course for people who need to participate in AI design reviews, standups, incident discussions, eval readouts, customer escalations, research/product debates, and launch decisions.

Core language challenge

AI teams often compress complex concepts into short jargon: RAG miss, eval slice, context budget, LoRA, P95, grounding failure, non-idempotent tool call, prompt injection, or model regression. Learners need both the vocabulary and the conversational moves around it: clarify, challenge, qualify, summarize, and recommend.

Course objectives

- Use common AI-development terms accurately in meetings, tickets, design reviews, and technical updates.
- Explain model behavior, data issues, evaluation results, and deployment tradeoffs in concise professional English.
- Participate in discussions about LLMs, RAG, agents, embeddings, fine-tuning, inference, safety, observability, and product risk.
- Ask high-quality clarification questions when requirements, metrics, or model outputs are ambiguous.
- Describe failures without exaggeration: hallucination, regression, drift, leakage, prompt injection, latency spikes, and retrieval misses.
- Deliver realistic AI-development updates to engineers, product managers, researchers, customers, and executives.

AI Team Communication Principles

Separate layers before blaming the model

In AI product work, 'the model failed' is often too vague. The failure may be in ingestion, chunking, embeddings, retrieval, reranking, prompt construction, tool calling, validation, UI, logging, or release configuration. Strong AI English names the layer, the evidence, the user impact, and the next diagnostic step.

Qualify claims

- Use 'the current eval suggests...' rather than 'the model is better' when the evidence is limited.
- Use 'this slice regressed' rather than 'the release is bad' when one subset has a problem.
- Use 'unsupported answer' or 'grounding failure' when the issue is a false claim without source support.
- Use 'we need a product eval' when a public benchmark may not match real user traffic.

Ask engineering-grade clarification questions

Weak question	Stronger AI-development question
Why is it wrong?	Which expected behavior did it violate, and do we have the prompt, output, trace, and retrieved context?
Can we fine-tune it?	Is the problem missing knowledge, wrong style, policy behavior, tool choice, or reasoning under context?
Is the model good?	Which eval set, which metric, which slice, and what are the severe failure examples?

Weak question	Stronger AI-development question
Can it be faster?	Where is the latency: retrieval, reranking, model inference, tool call, validation, or network?

Nomenclature and Jargon

Teach these terms as working vocabulary. Learners should be able to define the term, use it in a sentence, ask a clarification question about it, and explain its business consequence.

Core model terms

Term	Working meaning
LLM	Large language model; a model trained to process and generate language-like sequences.
Transformer	A neural architecture based on attention mechanisms, common in modern language models.
Parameter	A learned numerical value in a model; not the same as an API parameter.
Checkpoint	A saved version of model weights at a point in training or fine-tuning.
Foundation model	A broadly trained model adapted to many downstream tasks.
Multimodal	Able to handle more than one data type, such as text, image, audio, or video.

Prompt and context terms

Term	Working meaning
Prompt	The instructions, examples, user request, and context given to a model.
System prompt	High-priority instructions that guide model behavior inside an application.
Few-shot	Including examples in the prompt to show the desired pattern.
Context window	The amount of input and generated text the model can consider in one request.
Token	A unit of text processed by the model; token count affects cost, context, and latency.
Temperature	A generation setting that affects output variability.

Retrieval and RAG terms

Term	Working meaning
Embedding	A vector representation used for similarity search, clustering, classification, and related tasks.
Vector store	A database or index for storing and searching embeddings.
Chunking	Splitting documents into retrievable pieces.
Reranker	A model or step that reorders retrieved results for relevance.
RAG	Retrieval-augmented generation: retrieve relevant context, then generate an answer using it.
Grounding	Tying model output to retrieved, cited, or verified source information.

Training and adaptation terms

Term	Working meaning
Fine-tuning	Updating model weights on task- or domain-specific data.
SFT	Supervised fine-tuning with input-output examples.
RLHF	Reinforcement learning from human feedback; training with human preference signals.
DPO	Direct preference optimization; preference tuning without a separate reward model in common workflows.
LoRA	Low-rank adaptation; a parameter-efficient fine-tuning method.
Adapter	A small trainable module inserted into or attached to a pretrained model.

Evaluation terms

Term	Working meaning
Eval	A test or evaluation suite for model or system behavior.
Benchmark	A standardized test used to compare systems, often imperfect for a product use case.
Golden set	Curated examples used repeatedly to test important behavior.
Regression	A behavior that gets worse after a change.
Pass rate	The percentage of eval cases meeting the success criterion.
LLM-as-judge	Using a model to evaluate outputs, usually with calibration and human review.

Production terms

Term	Working meaning
Inference	Running a trained model to produce an output.
Latency	How long a request takes to return a result.
Throughput	How many requests a system can handle in a period of time.
Batching	Processing multiple requests together for efficiency.
Streaming	Sending partial output to the user as it is generated.
Fallback	A backup behavior when the preferred path fails.

Safety and security terms

Term	Working meaning
Hallucination	A generated claim that is unsupported, false, or not grounded in the provided context.
Prompt injection	Untrusted input tries to manipulate model instructions or tool use.
Jailbreak	A prompt or interaction that tries to bypass safety constraints.
Guardrail	A control that detects, blocks, changes, or routes risky behavior.
PII	Personally identifiable information.
Red team	A structured effort to find failures, vulnerabilities, or unsafe behavior.

Instructor Module Plans

Module 1. Speaking the AI Development Stack (90 minutes)

AI teams use a layered vocabulary: model, data, prompt, retrieval, tools, serving, evaluation, monitoring, and product experience. Learners need to locate a problem in the stack before they can discuss it clearly.

Learning objectives

- Name the layer where an AI issue occurs.
- Distinguish model capability from application behavior.
- Use concise stack language in standups and tickets.

Core concepts

- Model vs application: the model generates outputs; the application wraps prompts, retrieval, tools, policy, UI, logging, and fallback behavior around it.
- Pipeline: a sequence of steps such as ingest, chunk, embed, retrieve, rerank, prompt, generate, validate, and log.
- System boundary: what belongs to the model provider, the app team, the data team, or the platform team.

Activities

1. Stack map: learners place 24 terms in the correct layer.
2. Ticket rewrite: learners turn vague issue reports into stack-specific bug reports.
3. Standup drill: learners give a 45-second AI feature update with blocker, evidence, and next step.

Learner outputs

- AI stack vocabulary map.
- Three polished standup updates.

> **Facilitator note**
>
> Push learners toward evidence. When they use a broad term such as better, broken, hallucinated, unsafe, or slow, ask: compared with what, measured how, in which slice, and with what user impact?

Module 2. LLMs, Transformers, Tokens, and Context (90 minutes)

High-level AI communication often depends on explaining what the model sees: tokens, messages, context window, instructions, examples, retrieved text, and tool results.

Learning objectives

- Explain tokens, context window, prompts, and attention in workplace English.
- Discuss prompt changes without implying that prompting is magic.
- Describe context limits, truncation, and cost/latency tradeoffs.

Core concepts

- Tokenization: text is split into model-readable units; tokens affect context length, cost, and latency.
- Context window: the information available to the model at generation time.
- Instruction hierarchy: system/developer instructions, user request, retrieved context, examples, and tool outputs may compete.

Activities

1. Plain-English explainer: learners explain tokenization to a product manager.
2. Prompt review: learners critique a prompt for ambiguity, hidden assumptions, and missing output constraints.
3. Context budget negotiation: learners decide what to include or remove when a prompt is too large.

Learner outputs

- Prompt review checklist.
- Context budget explanation script.

Facilitator note

Push learners toward evidence. When they use a broad term such as better, broken, hallucinated, unsafe, or slow, ask: compared with what, measured how, in which slice, and with what user impact?

Module 3. Data, Datasets, Labels, and Leakage (90 minutes)

AI systems are shaped by data quality. Teams need precise language for dataset splits, annotation guidelines, leakage, imbalance, representativeness, and privacy constraints.

Learning objectives

- Describe a dataset and its limitations without overclaiming.
- Explain labels, ground truth, noisy labels, and inter-annotator agreement.
- Raise leakage and privacy concerns clearly.

Core concepts

- Train/validation/test split: different data subsets serve different purposes in model development and evaluation.
- Ground truth: the expected answer or label, often created or verified by humans.
- Data leakage: information from the target, test set, or future state accidentally enters training or evaluation.

Activities

1. Dataset card discussion: learners summarize source, coverage, bias, limitations, and risk.
2. Annotation meeting: learners negotiate ambiguous labeling guidelines.
3. Leakage hunt: learners identify possible leakage in five project descriptions.

Learner outputs

- Dataset limitation statement.
- Annotation guideline clarification questions.

Facilitator note

Push learners toward evidence. When they use a broad term such as better, broken, hallucinated, unsafe, or slow, ask: compared with what, measured how, in which slice, and with what user impact?

Module 4. Retrieval, Embeddings, Vector Search, and RAG (90 minutes)

Many production AI apps combine retrieval with generation. Learners need to discuss chunking, embeddings, vector stores, recall, reranking, grounding, citations, and retrieval misses.

Learning objectives

- Explain embeddings, semantic search, vector stores, and RAG.
- Diagnose retrieval failures separately from generation failures.
- Discuss chunking, metadata, reranking, and grounding in design reviews.

Core concepts

- Embedding: a vector representation of text or other data designed to preserve useful meaning for tasks such as search or clustering.
- RAG: retrieval brings external context into the generation step so the system can answer from documents rather than only from model parameters.
- Grounding: connecting output claims to retrieved or otherwise verified source material.

Activities

1. RAG failure triage: learners decide whether the bug is ingestion, chunking, retrieval, reranking, prompt, or generation.
2. Retrieval metrics discussion: learners compare recall, precision, MRR, and answer faithfulness.
3. Architecture explanation: learners explain a RAG pipeline to a non-technical stakeholder.

Learner outputs

- RAG troubleshooting flow.
- Stakeholder explanation of retrieval vs generation.

Facilitator note

Push learners toward evidence. When they use a broad term such as better, broken, hallucinated, unsafe, or slow, ask: compared with what, measured how, in which slice, and with what user impact?

Module 5. Fine-Tuning, Alignment, and Adaptation (90 minutes)

Teams often confuse prompt changes, RAG, fine-tuning, adapters, supervised fine-tuning, preference tuning, and RLHF. The language goal is to recommend the right adaptation method for the problem.

Learning objectives

- Differentiate prompt engineering, RAG, fine-tuning, LoRA/adapters, SFT, DPO, and RLHF at a practical level.
- Discuss when fine-tuning is not the right solution.
- Explain cost, data, evaluation, and maintenance tradeoffs.

Core concepts

- Fine-tuning changes model weights; prompting and RAG change the information or instructions around the model.
- PEFT/LoRA/adapters: parameter-efficient approaches that adjust a smaller number of parameters than full fine-tuning.
- Alignment and preference tuning: training or optimizing models to follow desired behavior patterns, policies, or preferences.

Activities

1. Method selection: learners choose between prompt fix, retrieval fix, fine-tune, or product constraint.
2. Tradeoff pitch: learners defend an adaptation method to engineering and product.
3. Risk statement: learners describe what could get worse after fine-tuning.

Learner outputs

- Adaptation decision matrix.
- Fine-tuning recommendation memo.

Facilitator note

Push learners toward evidence. When they use a broad term such as better, broken, hallucinated, unsafe, or slow, ask: compared with what, measured how, in which slice, and with what user impact?

Module 6. Evaluation, Benchmarks, and Regression (90 minutes)

AI teams need language for uncertainty. 'It looks better' is not enough. Learners need to discuss offline evals, online evals, golden sets, human review, model-graded evals, regression, pass rate, and confidence.

Learning objectives

- Explain evaluation design and limitations.
- Report metrics with context and uncertainty.

- Push back on weak benchmarks or misleading averages.

Core concepts

- Golden set: a curated set of representative examples used for repeated evaluation.
- Regression: a previously working behavior gets worse after a change.
- LLM-as-judge: a model evaluates outputs, useful but requiring calibration and human spot checks.

Activities

1. Eval readout: learners present a metric change, sample failures, and release recommendation.
2. Metric critique: learners find blind spots in an eval plan.
3. Regression triage: learners decide whether to block a release.

Learner outputs

- Evaluation readout template.
- Release recommendation script.

Facilitator note

Push learners toward evidence. When they use a broad term such as better, broken, hallucinated, unsafe, or slow, ask: compared with what, measured how, in which slice, and with what user impact?

Module 7. Inference, Latency, Cost, and Deployment (90 minutes)

AI development is also systems engineering. Learners need vocabulary for inference paths, throughput, batching, caching, rate limits, GPUs, quantization, streaming, fallbacks, and SLOs.

Learning objectives

- Explain latency and cost drivers in AI applications.
- Discuss deployment constraints without losing product trust.
- Use incident and observability language for model-serving issues.

Core concepts

- Inference: running a trained model to produce outputs.
- Latency vs throughput: response time for one request vs volume handled over time.
- Fallback: a planned behavior when the preferred model, tool, or retrieval path fails.

Activities

1. Latency budget: learners explain where time is spent in a request.
2. Cost tradeoff: learners compare a larger model, smaller model, caching, batching, and retrieval changes.
3. Incident update: learners write a status update for a model-serving degradation.

Learner outputs

- Latency/cost explanation.
- AI incident update template.

Facilitator note

Push learners toward evidence. When they use a broad term such as better, broken, hallucinated, unsafe, or slow, ask: compared with what, measured how, in which slice, and with what user impact?

Module 8. Safety, Security, Privacy, and Governance (90 minutes)

AI teams must discuss risk precisely: hallucination, prompt injection, jailbreaks, PII, data retention, bias, harmful output, policy enforcement, audit logs, and human-in-the-loop review.

Learning objectives

- Name safety and security risks without dramatic overstatement.
- Explain guardrails, validation, red-teaming, and monitoring.
- Escalate privacy or policy concerns with evidence.

Core concepts

- Prompt injection: untrusted content attempts to override or manipulate system behavior.
- Guardrail: a policy, model, validation layer, or product control designed to reduce unsafe or unwanted behavior.
- Human-in-the-loop: a workflow where humans review, approve, correct, or audit model behavior.

Activities

1. Risk register: learners write risk, trigger, impact, mitigation, owner, and review cadence.
2. Red-team debrief: learners report adversarial testing results clearly.
3. Privacy escalation: learners decide when to pause launch and involve legal/security.

Learner outputs

- AI risk register entry.
- Safety escalation script.

Facilitator note

Push learners toward evidence. When they use a broad term such as better, broken, hallucinated, unsafe, or slow, ask: compared with what, measured how, in which slice, and with what user impact?

Assessment and Coaching

Pre-course diagnostic

- Learner explains their current AI project in 90 seconds, including model, data, user, risk, and release stage.
- Learner defines ten common AI-development terms and uses five in realistic workplace sentences.
- Learner handles a short role-play: a product manager asks whether a model upgrade is ready to ship.

Performance rubric

Skill	Developing	Proficient	Strong
Terminology	Recognizes terms but uses them loosely.	Uses common terms accurately in context.	Defines terms, asks precise questions, and notices misuse.
Technical clarity	Explains everything as a model issue.	Names likely stack layer and evidence.	Separates hypotheses and proposes diagnostic steps.
Eval communication	Reports metrics without limitations.	Reports metric, slice, sample, and failure examples.	Connects eval evidence to release recommendation.
Dialogue control	Waits passively in technical debate.	Clarifies, summarizes, and pushes back politely.	Guides mixed technical/product discussion toward decision.
Risk language	Overstates or minimizes AI risks.	Names risk and mitigation clearly.	Escalates with evidence, owner, and review cadence.

Capstone simulation

Learners lead a release-readiness meeting for an AI document assistant. The system has improved average answer quality, but one legal-policy slice regressed; latency also increased after reranking. The learner must explain the evidence, ask for missing information, recommend ship/hold/limited beta, and write the follow-up summary.

Source orientation for instructors

- OpenAI API documentation: embeddings, tools/function calling, structured outputs, evals, and agent eval guidance.
- OpenAI Cookbook examples on evaluation and RAG workflows.
- Hugging Face documentation: Transformers, Tokenizers, PEFT, Evaluate, and Hub concepts.
- Google Machine Learning Glossary and Responsible AI glossary.
- Lewis et al., Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks, NeurIPS 2020.