https://forms.gle/nUC3Cuj7DWmrs2sc9

# Cursor

Cursor went from 1–100m ARR in 12 months: the fastest ...

Cursor Is Writing 1 Billion Lines Of Code A Day

# Introduction to LLM Applications and Cost Optimization

Hamza Salem - Innopolis university

# Who am I?

## Hamza Salem

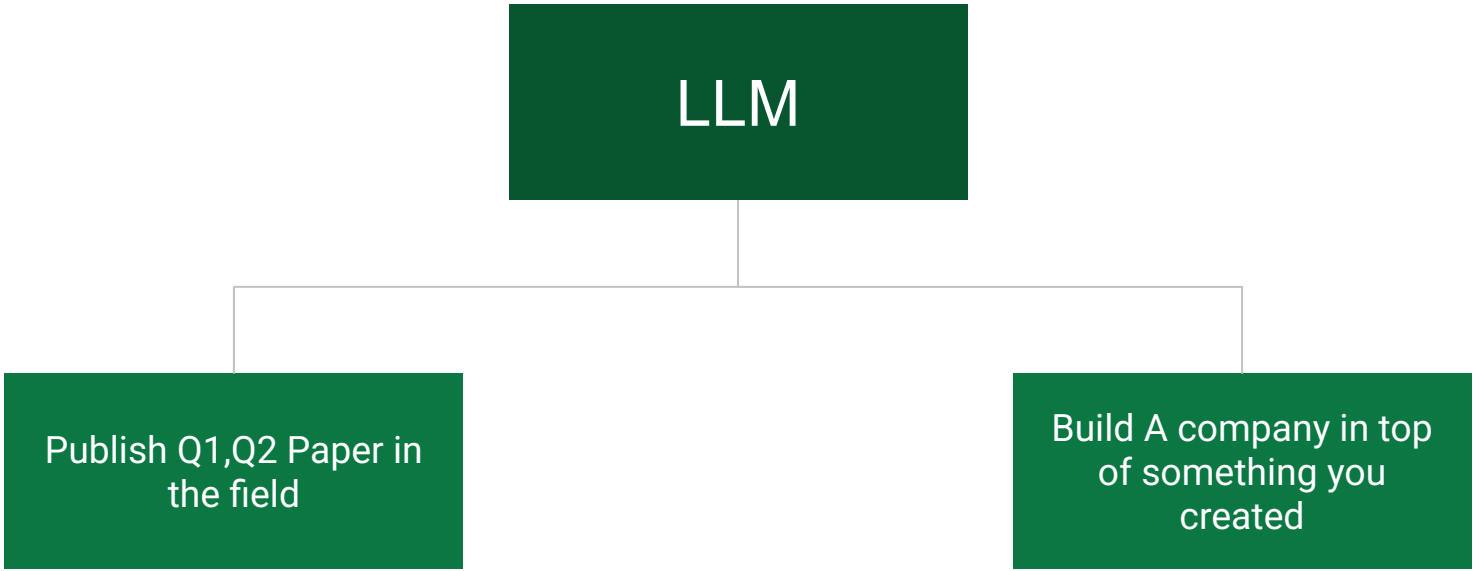
- ➔ PhD in Computer Science
- ➔ Chief Technology Officer at STYLE Protocol (Switzerland) & Flaca AI
- ➔ Tech YouTuber with 81K+ Subscribers
- ➔ Educator in Web3, Fintech, Databases, and LLM Applications
- ➔ I love building things that makes money.

# Course Goal

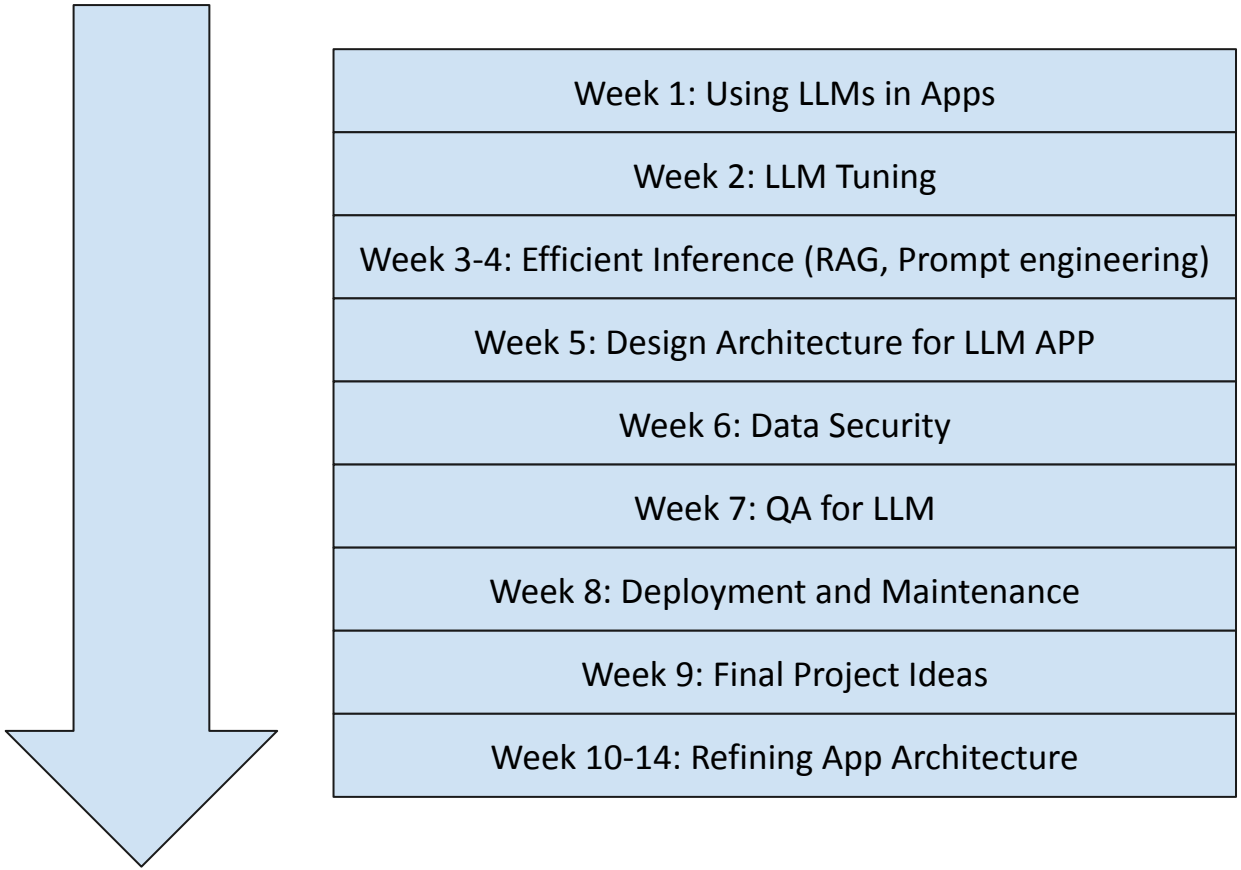
- Moodle link : https://moodle.innopolis.university/course/view.php?id=3371
- Create an optimized architecture that involve LLM as main or supportive component for web or mobile app.
- Online: https://ithelp.ktalk.ru/yjxuymstsx4q

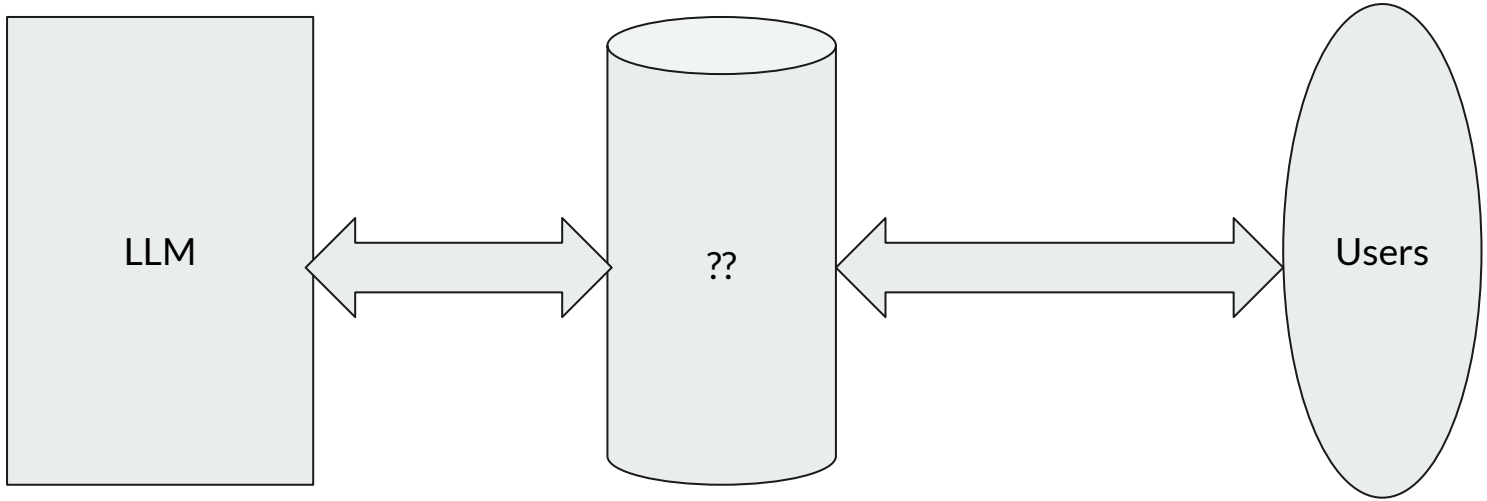
# Grades

- Labs 30%
- Midterm 20%
- Final Project 50%
- 85%+: A
- 65%-84%: B
- 50%--64%: C




- https://www.scimagojr.com/journalrank.php?category=1702
- https://www.ycombinator.com/companies/industry/ai


Week 1: Using LLMs in Apps
Week 2: LLM Tuning
Week 3-4: Efficient Inference (RAG, Prompt engineering)
Week 5: Design Architecture for LLM APP
Week 6: Data Security
Week 7: QA for LLM
Week 8: Deployment and Maintenance
Week 9: Final Project Ideas
Week 10-14: Refining App Architecture


LLM
??
Users

# Agenda

Introduction to Large Language Models (LLMs)

Real-World Applications of LLMs

Steps to Implement LLM Solutions

Cost Optimization in LLM Deployments

Summary and Q&A

# Understanding LLMs

Large Language Models (LLMs) are advanced AI systems trained on vast amounts of text data to understand, generate, and manipulate human language.

Key Features:

- Scale: Billions to trillions of parameters
- Data: Trained on diverse and extensive datasets
- Output: Text generation, translation, summarization, and more

Example: GPT-4, BERT, T5

# Scope and Applications of LLMs

Real-World Applications of LLMs:

- Content Creation: Automated article, blog post, and social media content generation.
- Customer Support: Chatbots for handling customer queries.
- Code Generation: Writing code snippets, bug fixes (e.g., GitHub Copilot).
- Healthcare: Summarizing medical documents, assisting in diagnosis.
- Finance: Automated report generation, risk assessment.

# How LLMs Work

Mechanisms Behind LLMs:

Training Process:

- Data Collection: Massive datasets (text from books, websites, etc.)
- Preprocessing: Cleaning and preparing data for training.
- Training: Using neural networks (e.g., Transformers) to learn patterns.

Fine-Tuning: Adjusting the model for specific tasks.

Inference: The model generates outputs based on input prompts.

# How LLMs Work

How LLMs work:

- Data input — You ask ChatGPT a question or tell it something.
- Text embedding — ChatGPT converts these tokens into vectors, a process known as “embedding”. Each token corresponds to a specific point in a high-dimensional space.
- Neural Network processing — ChatGPT uses its ‘brain’ (a thing called a neural network) to think about what you said. It’s like it’s solving a big puzzle!
- Context Understanding — Then it figures out how all the words you used to link together, using a trick called the ‘transformer’ model.
- Response Generation — It thinks up the best tokens to answer your question.
- Spitting out the answer — Finally, it changes the number code back into words, and that’s the answer you get!

https://colab.research.google.com/drive/1MomvqvP3mz_aLep7V5uQ-5UE0umhdb3R?usp=sharing

# Summary → How LLMs Work

1- Input: "What is AI?"

2- Tokenization: ["What", "is", "AI", "?"]

3- Embedding: Vectors like [0.25, -0.1, 0.7, ...] for each token.

4- Neural Network: Processes vectors to understand relationships.

5- Transformer: Uses self-attention to focus on "AI" as the key word.

6- Response Generation: Predicts tokens like ["AI", "stands", "for", ...].

7- Output: Converts tokens back to text: "AI stands for artificial intelligence.

# Paths to Productionizing LLM Applications

From Concept to Production

- Prototype Development: Initial testing and validation.
- Model Selection: Choosing the right model based on task and performance.
- Integration: Embedding the model into existing systems (APIs, cloud services).
- Deployment: Scaling the solution for real-world use.
- Monitoring and Maintenance: Ongoing updates, error tracking, and model performance checks.

# Importance of Cost Optimization

High Costs of LLMs:

- Training Costs: GPU/TPU resources, energy consumption.
- Inference Costs: Operational costs for generating responses.

Strategies for Cost Management:

- Model Efficiency: Use smaller, task-specific models when possible.
- Resource Allocation: Optimize cloud computing resources.
- Batch Processing: Aggregate tasks to reduce overhead.
- Serverless Architectures: Pay-as-you-go pricing models.

# Example of Cost Optimization in LLM Deployment

Scenario: Deploying a customer support chatbot for an e-commerce platform.

Challenges: High volume of queries, need for real-time responses.

Solutions Implemented:

- Model Selection: Fine-tuning a smaller LLM instead of using the largest available model.
- Resource Management: Utilizing spot instances and auto-scaling groups to manage server costs.
- Efficiency: Implementing caching for common queries to reduce API calls.

# Transformer Model..

Transformers utilize a mechanism called **attention**, allowing the model **to focus on relevant parts of the input sequence**. This architecture is foundational to many modern LLMs, enabling them to handle longer text sequences efficiently..

# Key Takeaways

LLMs are powerful tools with wide-ranging applications but come with significant costs.

Understanding the mechanisms behind LLMs helps in making informed decisions on deployment.

Cost optimization is critical to making LLMs viable in production environments.

Next Steps: Exploring specific LLM tools and hands-on lab work in the next module.

# Context Engineering..?

# Context Engineering

Definition:
Context Engineering is the strategic practice of structuring and providing relevant information to an LLM to guide its reasoning and output, without changing its core weights (fine-tuning).

Core Idea:
LLMs don't "know" anything; they predict the next token based on the context they are given. The quality of the context directly determines the quality of the output.

It's like giving a brilliant research assistant:

- Bad Context: "Write a function to connect to a database."
- Good Context: "You are a senior Python backend developer. Write a secure function using `psycopg2` to connect to a PostgreSQL database. The function should handle connection errors gracefully and use environment variables for the host, user, and password. Include a docstring."

# Context Engineering ? Why it Matters?

Accuracy: Reduces hallucinations by grounding the model in facts.

Relevance: Ensures the output matches the desired style, format, and scope.

Efficiency: Often cheaper and faster than fine-tuning for specific tasks.

# The "Context Window" is Your Canvas

The context window is the total amount of text (input + output) an LLM can process in a single interaction. It's the model's "working memory."

Key Consideration:

Limited Resource: Context windows are finite (e.g., 128K tokens). You must use this space wisely.

Cost: Larger context windows can be more expensive per API call (e.g., GPT-4 Turbo 128K is more costly than GPT-4 8K).

The Goal of Context Engineering:

To pack the most relevant and useful information into this limited space to get the best possible result.

# Case Study: Cursor as an IDE Powered by Context Engineering

What is Cursor?

Cursor is an IDE built for programming that deeply integrates an LLM (like GPT-4) into the coding workflow. It's a perfect example of advanced context engineering in action.

# How Cursor Masters Context?

1. The Global Context (The Project):
    - Cursor automatically provides the LLM with a broad understanding of your entire codebase.
    - What it includes: Key files like `package.json`, `requirements.txt`, major component files, and architecture definitions.
    - Purpose: This gives the LLM the "big picture" so it can generate code that is consistent with your project's structure, dependencies, and conventions.
2. The Local Context (The Immediate Task):
    - This is the most directly visible form of context.
    - What it includes: The specific file you are editing, the code you've just selected, the error message in your terminal, and the comments you write.
    - Purpose: Allows the LLM to perform precise, localized tasks like fixing a bug, writing a function, or refactoring a block of code.

# Case Study: Cursor as an IDE Powered by Context Engineering

3. The Interaction Context (The Conversation):

    - Cursor maintains a chat history within the editor.
    - What it includes: Your previous questions and the AI's previous answers.
    - Purpose: This creates a coherent "conversation," allowing you to iteratively refine the code. You can say, "now make it faster" or "add error handling to the function you just wrote," and the AI understands the reference.

# How Cursor's Context Engineering Works in Practice?

Example: Adding a New Feature

User Action: You type a comment: // Create a function to validate user email addresses.

Context Provided by Cursor:

Local: The current file (e.g., utils/validation.js).

Global: The project's package.json (knows you're using Jest for testing), and other utility files (so it can follow existing code patterns).

Instructional: Your natural language command.

# How Cursor's Context Engineering Works in Practice?

LLM's "Thought Process":

"I am in a JavaScript project. The user wants a validation function. I should use a robust regex pattern for email validation. I see they use JSDoc comments in other files, so I'll add one. They have a __tests__ directory, so I should also generate a corresponding test file."

Output: Cursor generates a well-documented, syntactically correct function that fits the project's style and even suggests creating a test file.

This is Context Engineering: Seamlessly weaving together project rules, local code, and user intent to produce a highly specific and useful output.

# Key Takeaways & Applying These Principles

Summary:

Context is King: The output of an LLM is a direct reflection of the input context you provide.

Structure Your Context: Be explicit. Provide role, goal, constraints, and examples.

Tools Like Cursor are successful because they automate and optimize the delivery of rich, multi-layered context to the LLM.

# Lab 01

Create simple telegram bot that connect to (openAI, google gemini, or any other LLM) and make it answer questions about **diets and calisthenics**..feel free to use any telegram Bot library (Python,NodeJS).

- Bot should reply only with (diets and calisthenics) information .
- Bot should ask weight, height and age before replying..

Submission in Gitlab https://gitlab.pg.innopolis.university/

Submit your Code in week01/main.py  folder and add readme file with bot handle.

# Resources

- https://www.perplexity.ai/search/introduction-to-large-language-N_AxP5ORRsu8wjvE1U1PEg
- https://link.springer.com/article/10.1007/s11023-020-09548-1
- https://developers.google.com/machine-learning/resources/intro-llms
- https://arxiv.org/pdf/1706.03762
- https://www.youtube.com/watch?v=P8umbT9eDiM
- AIzaSyCs-dfJ3NesMO-Ghhrhmdx-5-830h92Yro