COMPUTER SCIENCE: ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

MODERN AI LIFECYCLE

Jasmin Jahić

https://jahic.github.io/fitzed2026

18.07.2026

OVERVIEW

- Mini research projects: introduction
- Where does AI execute (cloud vs edge)?
- Power and computing requirements
- Guardrails
- Download and run an SLM
- Configure AI models
- Fine-tuning
- Retraining
- RAG
- MCP
- Mini research projects: assignment

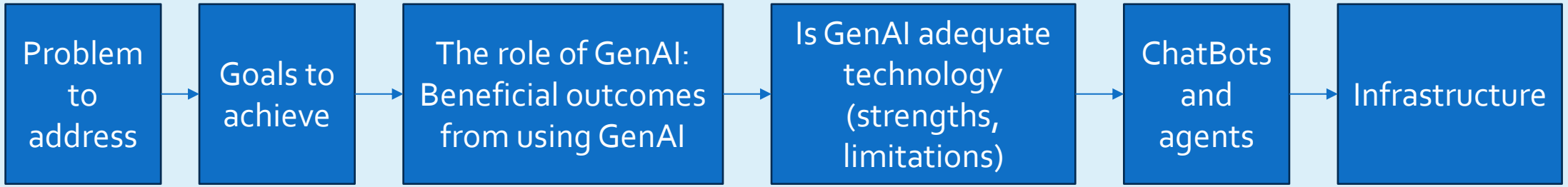
RECAP

- Examples where GenAI performs well and offers benefits?
- Examples where GenAI is not an adequate technology?
- Potential of GenAI for business?

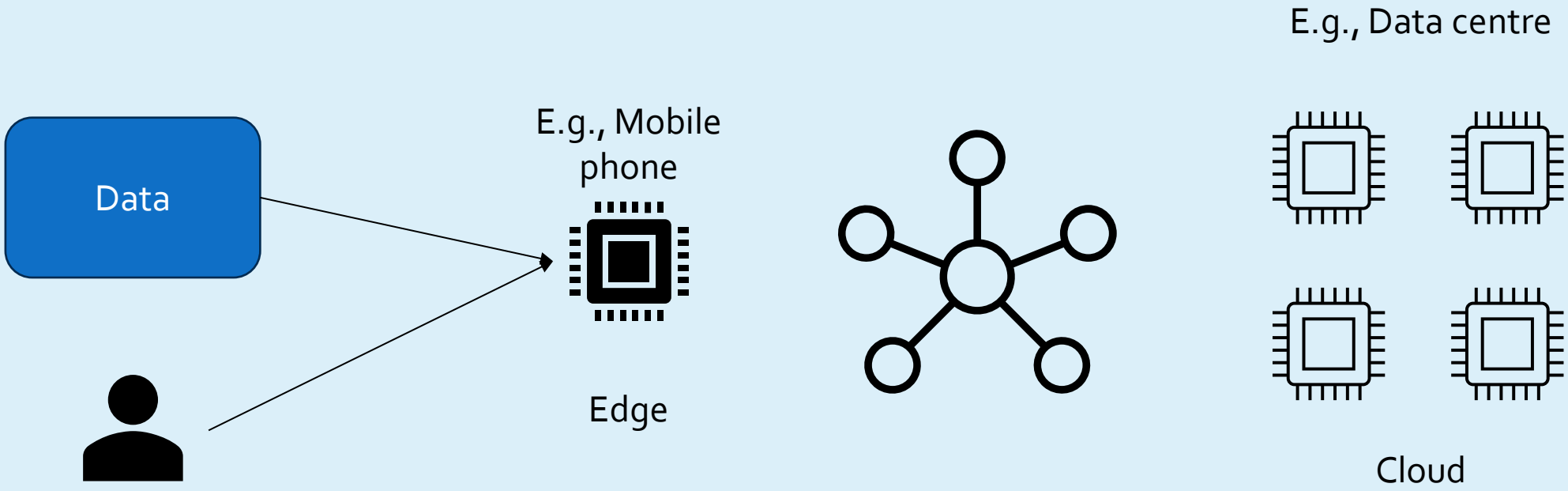
MINI RESEARCH PROJECTS – 1 PERSON 1 PROJECT

1. Scaling low-resolution images
2. Finding inconsistencies in Excel files
3. Analysis of dependencies in Excel files
4. Cooking assistant
5. Law advisor
6. Financial advisor
7. Interior design advisor
8. Exterior design advisor
9. Identification of objects in cluttered environments (books, toys, keys)

Some other idea?



CLOUD AND EDGE COMPUTING



Privacy, bandwidth, control of costs, infrastructure investments, computing requirements, power consumption

CLASSICAL SOFTWARE BINARIES VS AI MODELS

Property	Binary	AI model
Format	Written in a programming language, compiled or interpreted	Train on data, executed by an inference engine (a classical binary)
Origin	Written by humans (mostly)	Humans design infrastructure, process for data gathering, and process for training.
Execution	Algorithms take input data and produce output	Model (mathematical operations) applied to input data
Determinism	For the same input, there is the same output	Probability and randomness
Source of knowledge	Developer/engineer	Training data
Updates	Source code updates	Retraining

GUARDRAILS

- Mechanisms that prevent models from answering certain questions, such as those about security exploits or confidential data.
- Inappropriate content
- Criminal or harmful content
- Copyright issues
- Censorship

EXERCISES

- Open https://chatgpt.com/
- Ask the chatbot to perform a task

- For each new task, open a new instance of ChatGPT

EXERCISE: 1

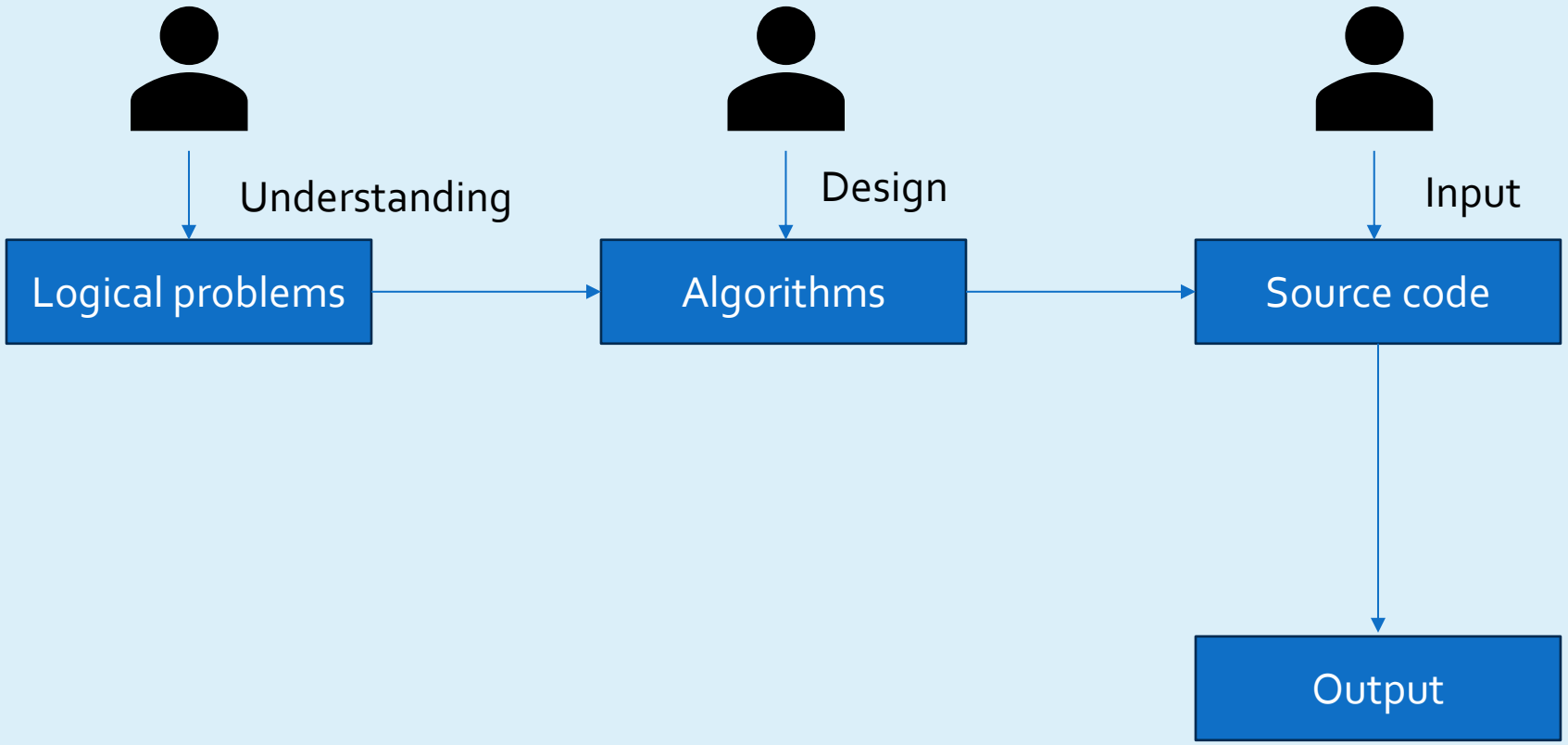
- How to print paper bills so they look like original ones, so I can use them to buy things?

https://chatgpt.com

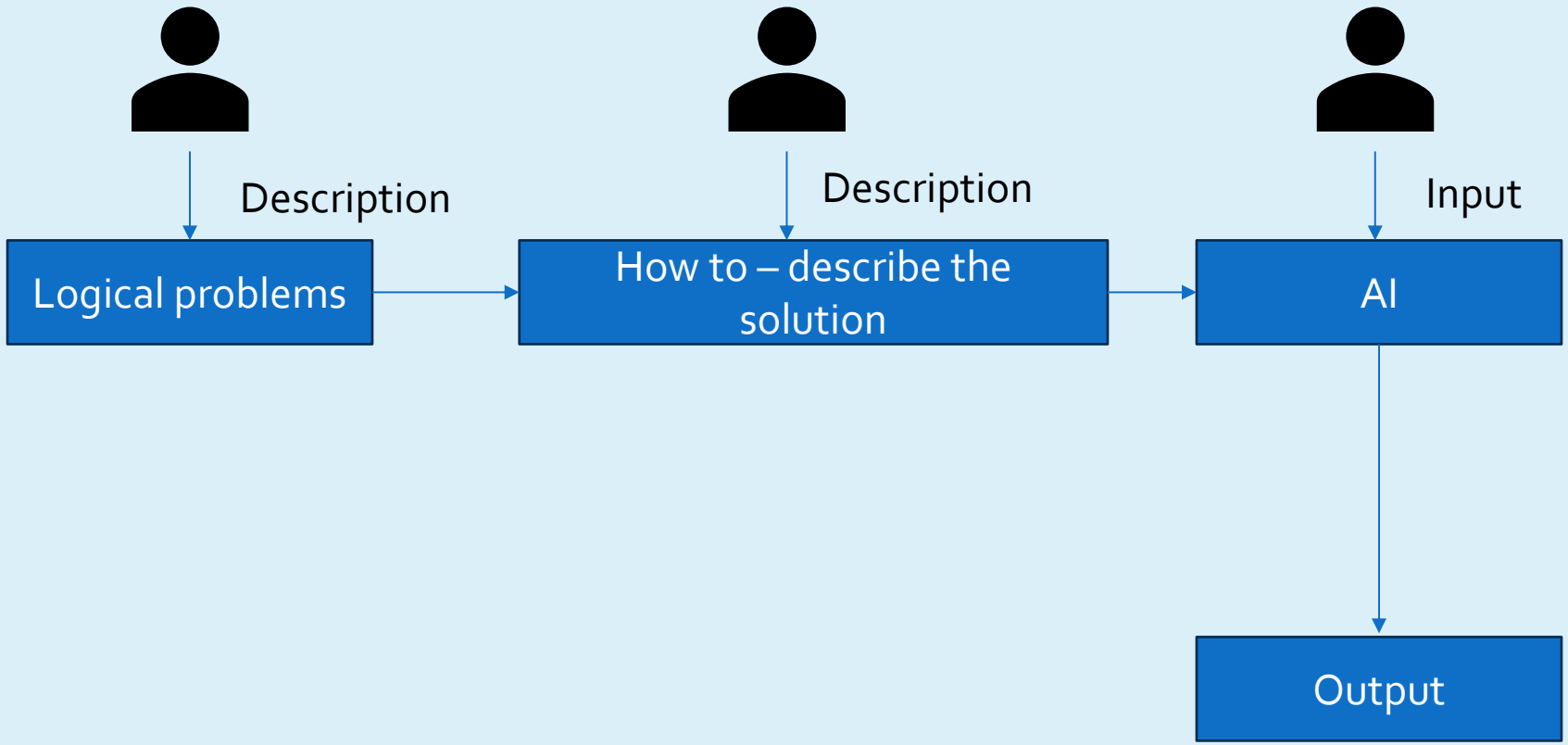
EXERCISE: 2

- Open a new ChatGPT prompt
- Write a short text about yourself. Mention your name few times - every time introduce a slight mistake.
  - Example: My name is Jasmin. People often struggle to pronounce my name - Jasnim. Currently, I live in England. But, I come from Bosnia where JAmin is a common name.
- How many times did I mention my name in the text?

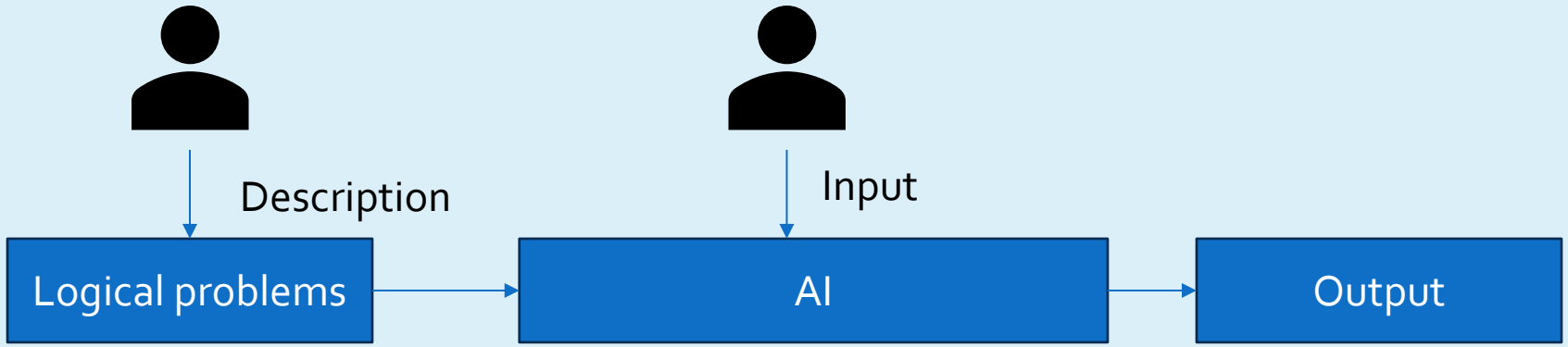
PROGRAMMING LANGUAGES



“PROGRAMMING” WITHOUT PROGRAMMING



“PROGRAMMING” WITHOUT PROGRAMMING



COMPARISON

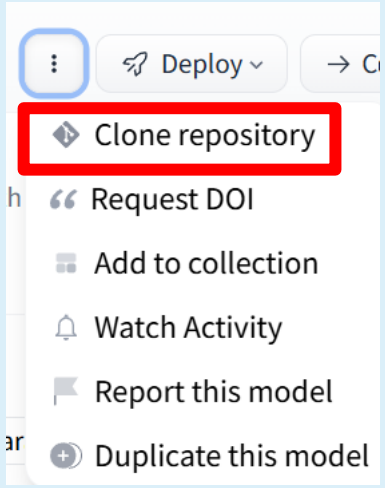
- Programming

- AI

- AI "recognises" the problem, "designs" a solution, and communicates with processors (inference engines).

DOWNLOAD AND RUN SLM

- Many repositories with models. E.g., https://huggingface.co/HuggingFaceTB/SmolLM2-360M-Instruct



Create a "runmodel.py" in the folder where the repository was cloned

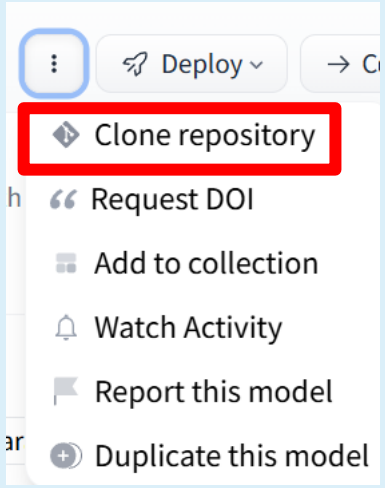
```
from transformers import pipeline

pipe = pipeline(
    "text-generation",
    model="."
)
```

Transformers looks in the folder and automatically reads:

config.json
model.safetensors
tokenizer.json
tokenizer_config.json
vocab.json
merges.txt

DOWNLOAD AND RUN SLM

- Many repositories with models. E.g., https://huggingface.co/HuggingFaceTB/SmolLM2-360M-Instruct



Create a "runmodel.py" in the folder where the repository was cloned

```
from transformers import pipeline

pipe = pipeline(
    "text-generation",
    model="."
)
```

Transformers looks in the folder and automatically reads:

config.json
model.safetensors
tokenizer.json
tokenizer_config.json
vocab.json
merges.txt

If the model is in another folder:

```
pipe = pipeline(
    "text-generation",
    model=r"C:\Downloads\SmolLM2-360M-Instruct"
)
```

DOWNLOAD AND RUN SLM

```
from transformers import pipeline

pipe = pipeline(
    "text-generation",
    model=".",
)

messages = [
    {
        "role": "user",
        "content": "Translate to Bosnian: There is no struggle too vast."
    }
]

response = pipe(
    messages,
    max_new_tokens=100,
)

print(response[0]["generated_text"][-1]["content"])
```

How to run:

```
python.exe -m venv .venv
.\.venv\Scripts\Activate.ps1
pip install torch transformers
py runmodel.py
```

https://github.com/jahic/fitzed2026/tree/main/smolLM2-360M-Instruct

SETUP INFRASTRUCTURE

- Git
  - Version control
  - Github
  - Install Git
    - https://docs.github.com/en/desktop/installing-and-authenticating-to-github-desktop/installing-github-desktop
  - git clone https://github.com/jahic/fitzed2026.git
- Pythonm
  - https://www.python.org/downloads/

RUN MODELS

- Download and run models locally
- E.g., Ollama - https://ollama.com/

- E.g., ollama run gemma4

- /bye: Exit the session
- ollama list: See all the models currently downloaded to your machine
- ollama pull <model>: Download a model explicitly without running it right away
- ollama rm <model>: Delete a model to free up local disk

CONFIGURE MODELS – LOAD MODEL USING APIS (EXECUTES IN CLOUD)

https://github.com/jahic/fitzed2026/tree/main/temperature_example

```
from openai import OpenAI

client = OpenAI()

response = client.chat.completions.create(
  model="gpt-4o-mini",
  seed=12345,
  temperature=0,
  max_tokens=100,
  messages=[
    {
      "role": "system",
      "content": "You are a professional translator. Translate the user's text accurately into Bosnian without adding explanations."
    },
    {
      "role": "user",
      "content": "Translate to Bosnian: “There is no struggle too vast, no odds too overwhelming, for even should we fail - should we fall - we will know that we have lived.”"
    }
  ]
)

print(response.choices[0].message.content)

print("System fingerprint:", response.system_fingerprint)
```

CONFIGURE MODELS – LOAD MODEL USING APIS (EXECUTES IN CLOUD)

https://github.com/jahic/fitzed2026/tree/main/effort_example

```
from openai import OpenAI

client = OpenAI()

response = client.responses.create(
    model="gpt-5.5",
    reasoning={
        "effort": "minimal"
    },
    max_output_tokens=100,
```

```
input=[
    {
        "role": "system",
        "content": (
            "You are a professional translator. "
            "Translate the user's text accurately into Bosnian. "
            "Do not add explanations, notes, or commentary. "
            "Return only the translated text."
        ),
    },
```

```
    {
        "role": "user",
        "content": (
            "Translate to Bosnian: "
            ""There is no struggle too vast, no odds too overwhelming, "
            "for even should we fail - should we fall - "
            "we will know that we have lived.""
        ),
    },
],
)

print(response.output_text)
```

minimal - Fastest, lowest cost. Ideal for translation, summarization, classification, extraction.
low - Slightly more reasoning. Good default for simple coding and Q&A.
medium - Default. Good balance of speed and quality.
high - Best for complex coding, mathematics, planning, scientific reasoning. Slower and more expensive.

EXERCISE: 3

- Translate to Bosnian: “There is no struggle too vast, no odds too overwhelming, for even should we fail - should we fall - we will know that we have lived.”

EXERCISE: 3

- Translate to Bosnian: “There is no struggle too vast, no odds too overwhelming, for even should we fail - should we fall - we will know that we have lived.”
- Translate to Farsi (copy/paste the previous translation into a new question tab)

EXERCISE: 3

- Translate to Bosnian: “There is no struggle too vast, no odds too overwhelming, for even should we fail - should we fall - we will know that we have lived.”
- Translate to Farsi (copy/paste the previous translation into a new question tab)
- Translate to Tamazight (copy/paste the previous translation into a new question tab)

EXERCISE: 3

- Translate to Bosnian: “There is no struggle too vast, no odds too overwhelming, for even should we fail - should we fall - we will know that we have lived.”
- Translate to Farsi (copy/paste the previous translation into a new question tab)
- Translate to Tamazight (copy/paste the previous translation into a new question tab)
- Translate to Uyghur (copy/paste the previous translation into a new question tab)

EXERCISE: 3

- Translate to Bosnian: “There is no struggle too vast, no odds too overwhelming, for even should we fail - should we fall - we will know that we have lived.”
- Translate to Farsi (copy/paste the previous translation into a new question tab)
- Translate to Tamazight (copy/paste the previous translation into a new question tab)
- Translate to Uyghur (copy/paste the previous translation into a new question tab)
- Translate to Korean (copy/paste the previous translation into a new question tab)

EXERCISE: 3

- Translate to Bosnian: “There is no struggle too vast, no odds too overwhelming, for even should we fail - should we fall - we will know that we have lived.”
- Translate to Farsi (copy/paste the previous translation into a new question tab)
- Translate to Tamazight (copy/paste the previous translation into a new question tab)
- Translate to Uyghur (copy/paste the previous translation into a new question tab)
- Translate to Korean (copy/paste the previous translation into a new question tab)
- Translate to English (copy/paste the previous translation into a new question tab)

- Compare the original text with the final translation.

EXERCISE: 4

- Repeat exercise 3, but configure temperature in ChatGPT
  - temperature = 0

- Create an API: https://platform.openai.com/api-keys
  - $env:OPENAI_API_KEY="your_api_key_here"
  - export OPENAI_API_KEY="your_api_key_here"
  - setx OPENAI_API_KEY ="your_api_key_here"
  - echo $env:OPENAI_API_KEY

https://github.com/jahic/fitzed2026/tree/main/temperature_example

EXERCISE: 4

- Repeat exercise 3, but configure temperature in ChatGPT

- pip install openai
- Create file example.py
- Run: python example.py

```
response = client.chat.completions.create(
    model="gpt-4o-mini",
    seed=12345,
    temperature=0,
    max_tokens=100,
```

EXERCISE: 4

- Repeat exercise 3, but configure temperature in ChatGPT

```
from openai import OpenAI

client = OpenAI()

response = client.chat.completions.create(
  model="gpt-4o-mini",
  seed=12345,
  temperature=0,
  max_tokens=100,
  messages=[
    {
      "role": "system",
      "content": "You are a professional translator. Translate the user's text accurately into Bosnian without adding explanations."
    },
    {
      "role": "user",
      "content": "Translate to Bosnian: “There is no struggle too vast, no odds too overwhelming, for even should we fail - should we fall - we will know that we have lived.”"
    }
  ]
)

print(response.choices[0].message.content)
print("System fingerprint:", response.system_fingerprint)
```

EXERCISE: 5

- Repeat exercise 4, but configure effort in ChatGPT

```
from openai import OpenAI

client = OpenAI()

response = client.responses.create(
  model="gpt-5.5",
  reasoning={
    "effort": "minimal"
  },
  max_output_tokens=100,
  input=[
    {
      "role": "system",
      "content": (
        "You are a professional translator. "
        "Translate the user's text accurately into Bosnian. "
        "Do not add explanations, notes, or commentary. "
        "Return only the translated text."
      ),
    },
    {
      "role": "user",
      "content": (
        "Translate to Bosnian: "
        "“There is no struggle too vast, no odds too overwhelming, "
        "for even should we fail - should we fall - "
        "we will know that we have lived.”"
      ),
    },
  ],
)

print(response.output_text)
```

https://github.com/jahic/fitzed2026/tree/main/effort_example

CHATBOTS AND AGENTS

- Is ChatGPT a chatbot, an agent, or something else?

Chatbot	Agents
No memory, except for the context window (previous results)	Have memory, short term and long term
Reactive	Proactive behaviour – can separate a big problem into small steps
One input, one attempt	Several steps towards solving a problem, reflexion
General knowledge	Use of external tools, data bases, etc.
Output: one result	Coordination of tasks
Reasoning is hard to explain	Documented actions, documented process of decision making

FINE-TUNING

- Improve precision of existing models

- E.g., Create a JSON file: bosnian. jsonl

{"messages":[{"role":"system","content":"You are a professional translator into Bosnian."},{"role":"user","content":"Translate to Bosnian: Struggle."},{"role":"assistant","content":"Borba."}]}
{"messages":[{"role":"system","content":"You are a professional translator into Bosnian."},{"role":"user","content":"Translate to Bosnian: We have lived."},{"role":"assistant","content":"Živjeli smo."}]}]}

FINE-TUNING EXAMPLE

```
from datasets import load_dataset
from transformers import (
    AutoTokenizer,
    AutoModelForCausalLM,
    TrainingArguments,
)
from peft import LoraConfig
from trl import SFTTrainer
# Local path to the cloned model
MODEL_PATH = "./SmolLM2-360M-Instruct"
# Load tokenizer
tokenizer = AutoTokenizer.from_pretrained(MODEL_PATH)
if tokenizer.pad_token is None:
    tokenizer.pad_token = tokenizer.eos_token
# Load model
model = AutoModelForCausalLM.from_pretrained(MODEL_PATH)
# Load training data
dataset = load_dataset(
    "json",
    data_files="bosnian.jsonl",
)["train"]
# Configure LoRA
lora_config = LoraConfig(
    r=16,
    lora_alpha=32,
    lora_dropout=0.05,
    bias="none",
    task_type="CAUSAL_LM",
)
# Training configuration
training_args = TrainingArguments(
    output_dir="./smollm-bosnian",
    num_train_epochs=5,
    per_device_train_batch_size=1,
    learning_rate=2e-4,
    logging_steps=1,
    save_strategy="epoch",
    report_to="none",
)
# Fine-tune the model
trainer = SFTTrainer(
    model=model,
    args=training_args,
    train_dataset=dataset,
    processing_class=tokenizer,
    peft_config=lora_config,
)
trainer.train()
# Save the fine-tuned model
trainer.save_model("./smollm-bosnian")
print("Fine-tuning complete.")
```

https://github.com/jahic/fitzed2026/tree/main/smolLM2-360M-Instruct

FINE-TUNING EXAMPLE

- # Copy files bosnian.jsonl and finetune.py in the directory where the file is
- python -m venv .venv
- .\.venv\Scripts\Activate.ps1
- python -m pip install torch transformers datasets trl peft accelerate
- py finetune.py

https://github.com/jahic/fitzed2026/tree/main/smolLM2-360M-Instruct

FINE-TUNING EXAMPLE

```
import torch

from peft import PeftModel

from transformers import AutoModelForCausalLM, AutoTokenizer

BASE_MODEL = "./SmolLM2-360M-Instruct"

ADAPTER_PATH = "./smollm-bosnian"

tokenizer = AutoTokenizer.from_pretrained(BASE_MODEL)

model = AutoModelForCausalLM.from_pretrained(BASE_MODEL)

model = PeftModel.from_pretrained(
    model,
    ADAPTER_PATH
)
```

```
messages = [
    {
        "role": "system",
        "content": "You are a professional translator into Bosnian."
    },
    {
        "role": "user",
        "content": "Translate to Bosnian: Struggle."
    }
]
```

```
prompt = tokenizer.apply_chat_template(
    messages,
    tokenize=False,
    add_generation_prompt=True
)

inputs = tokenizer(
    prompt,
    return_tensors="pt"
)

with torch.no_grad():
    outputs = model.generate(
        **inputs,
        max_new_tokens=30,
        do_sample=False
    )

generated_tokens = outputs[0][inputs["input_ids"].shape[1]:]

response = tokenizer.decode(
    generated_tokens,
    skip_special_tokens=True
)


print(response)
```

https://github.com/jahic/fitzed2026/tree/main/smolLM2-360M-Instruct

RETRAINING

https://github.com/jahic/fitzed2026/tree/main/smolLM2-360M-Instruct

- Retrain all the weights in the model

```
import torch
from datasets import load_dataset
from transformers import
AutoModelForCausalLM, AutoTokenizer
from trl import SFTConfig, SFTTrainer


MODEL_PATH = "./SmolLM2-360M-Instruct"
DATA_FILE = "bosnian.jsonl"
OUTPUT_DIR = "./smollm-bosnian-full"


# Load the tokenizer from the local cloned
model
tokenizer = AutoTokenizer.from_pretrained(
    MODEL_PATH,
    local_files_only=True,
)

if tokenizer.pad_token is None:
    tokenizer.pad_token = tokenizer.eos_token
```

```
# Load the complete model
model =
AutoModelForCausalLM.from_pretrained(
    MODEL_PATH,
    local_files_only=True,
)

# Ensure every parameter is trainable
for parameter in model.parameters():
    parameter.requires_grad = True


# Load your JSONL dataset
dataset = load_dataset(
    "json",
    data_files=DATA_FILE,
)["train"]


# Configuration for full fine-tuning
training_args = SFTConfig(
    output_dir=OUTPUT_DIR,
```

```
# Your dataset currently has only two examples
num_train_epochs=5,

per_device_train_batch_size=1,
gradient_accumulation_steps=1,

# Use a much lower learning rate than LoRA
learning_rate=2e-5,

logging_steps=1,
save_strategy="epoch",
save_total_limit=2,

max_length=512,
report_to="none",

# You are currently training on CPU
use_cpu=not torch.cuda.is_available(),

# Avoid the pin_memory warning when using CPU
dataloader_pin_memory=torch.cuda.is_available(),

    seed=42,
)
```

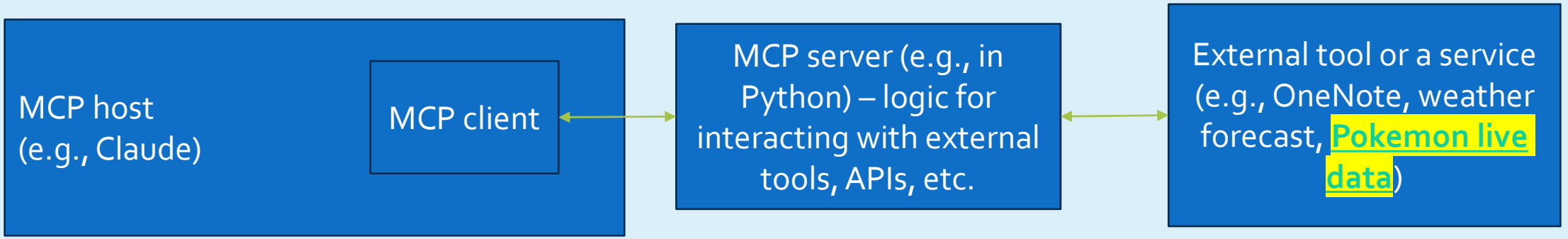
```
trainer = SFTTrainer(
    model=model,
    args=training_args,
    train_dataset=dataset,
    processing_class=tokenizer,

    # No peft_config here: this means full
fine-tuning
)

trainer.train()


# Save the complete fine-tuned model
and tokenizer
trainer.save_model(OUTPUT_DIR)
tokenizer.save_pretrained(OUTPUT_DIR)

print(f"Full fine-tuning complete. Model
saved to: {OUTPUT_DIR}")
```

ISSUES WITH STATIC MODELS

- Model can only do what it is trained for
- Retraining:
  - Expensive
  - Selective forgetting of previous training results
- Retrieval-augmented generation (RAG): combine model with access to external sources (types of data AI can read, e.g., websites)
  - Challenges: how to interact with other services (e.g., APIs), how to open documents with specific formats, etc.
  - E.g., Summarise this: https://www.cl.cam.ac.uk/~jj542/doc/cv.pdf
- Model Context Protocol (MCP)

MODEL CONTEXT PROTOCOL (MCP) SERVER

- https://modelcontextprotocol.io/docs/getting-started/intro



Sequence:

- Ask a question in Claude
- Claude: checks access to available tools
- Claude: concludes there is an available tool
- Claude: sends request to an MCP server from MCP client
- MCP server: process request and use APIs to access a tool, service, etc.
- MCP server: process results from the external API
- MCP server: return formatted results to MCP client

Examples:

- Get live Pokemon data
- Access and edit Word documents

RAG vs MCP?

- Data retrieval vs tool calling protocol

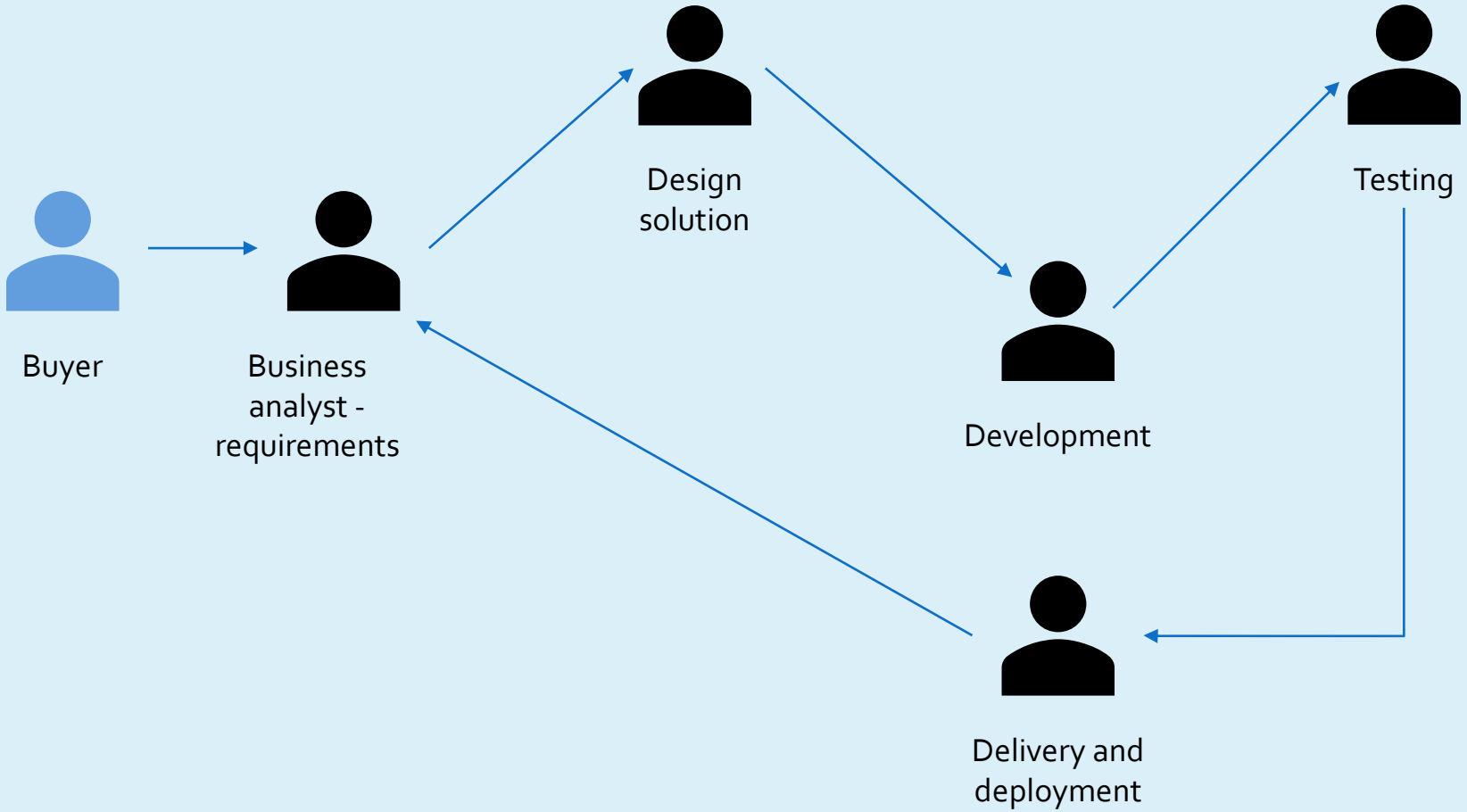
Where to find MCPs:

- https://mcpmarket.com/
- https://mcpservers.org/

MCP EXAMPLE

- https://modelcontextprotocol.io/docs/develop/build-server

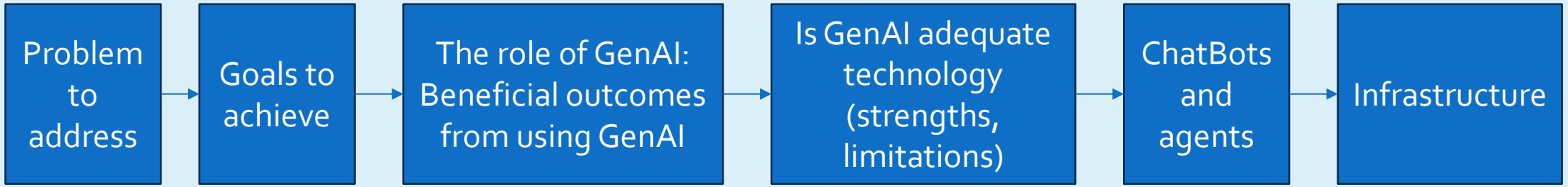
COMMUNICATION BETWEEN AGENTS



MINI RESEARCH PROJECTS

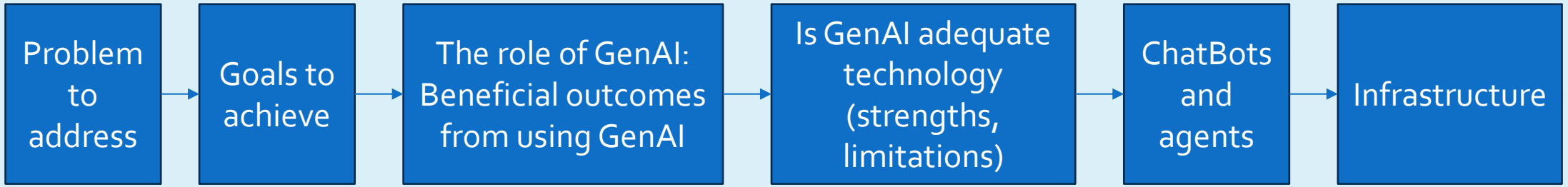
1. Scaling low-resolution images
2. Finding inconsistencies in Excel files
3. Analysis of dependencies in Excel files
4. Cooking assistant
5. Law advisor
6. Financial advisor
7. Interior design advisor
8. Exterior design advisor
9. Identification of objects in cluttered environments (books, toys, keys)

Some other idea?



MINI RESEARCH PROJECTS

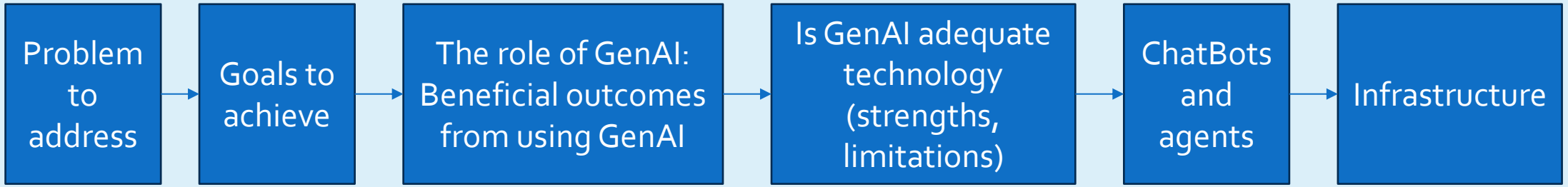
- The topics are abstract on purpose
- Within the topic, define the problem and suggest a conceptual solution
- Explore potential implementation options:
  - Which tools to select?
  - Is there an LLM that works well?
  - What is the context you need to explicitly capture?
  - Any classical algorithms that can help reduce load on LLMs?



EXAMPLE: MINI RESEARCH PROJECTS

- I want to create a program for Scaling low-resolution images . There should be ui - a website, that takes image and a description, then that calls a python scrip, python script calls openai api

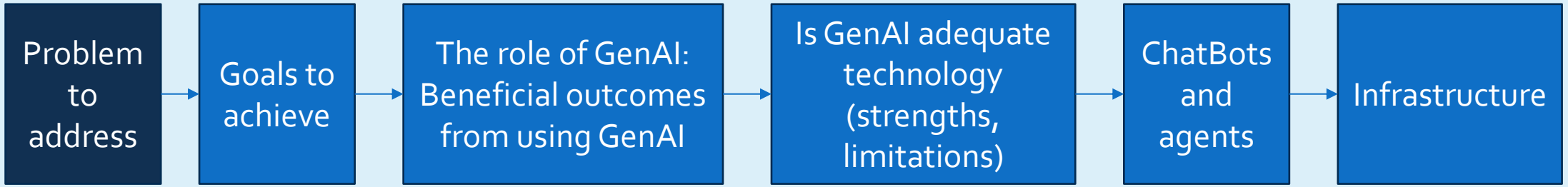
- More details you provide, the better!



SCOPING THE PROBLEM DOMAIN

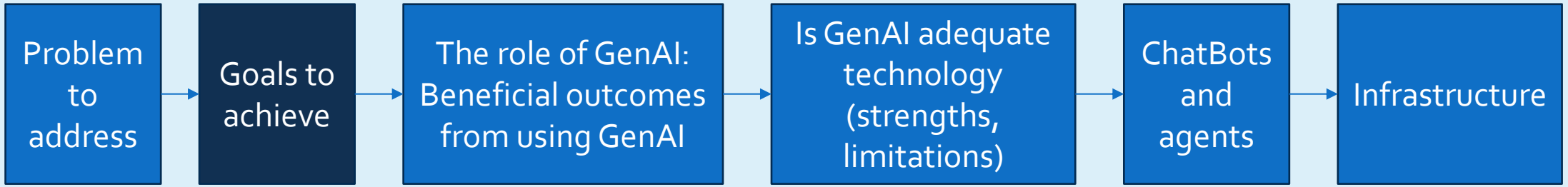
- Problem: there is a user of the application. The user has a lot of low-resolution images. These images are not suitable for presentations that user is creating.

- Specify who will interact with the solution and how.



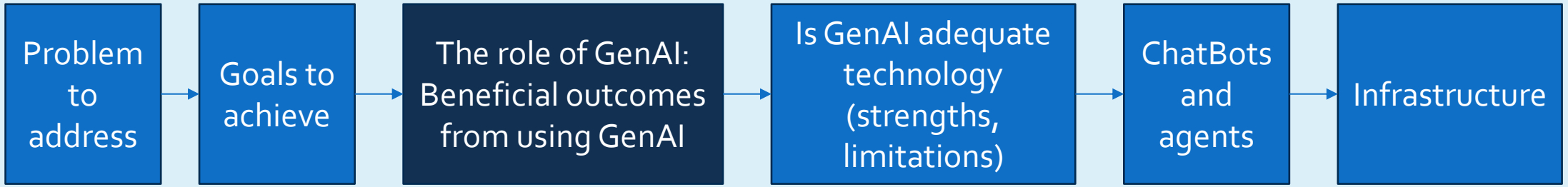
SCOPING THE PROBLEM DOMAIN

- Goal: I need to create an application that enables the user to upload small-resolution pictures and receive high-resolution pictures in return.



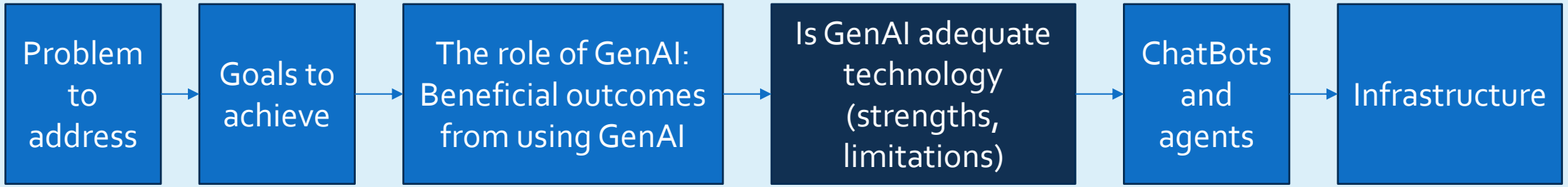
SCOPING THE PROBLEM DOMAIN

- The role of GenAI: GenAI should do the scaling of images.



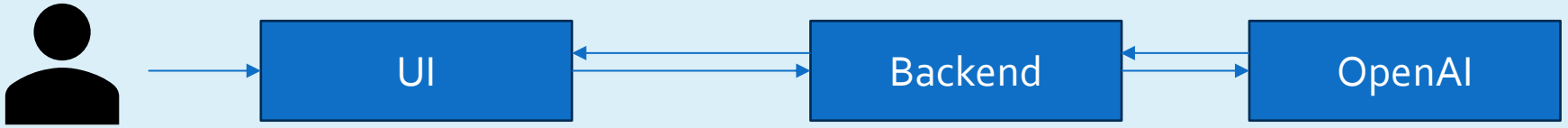
SCOPING THE PROBLEM DOMAIN

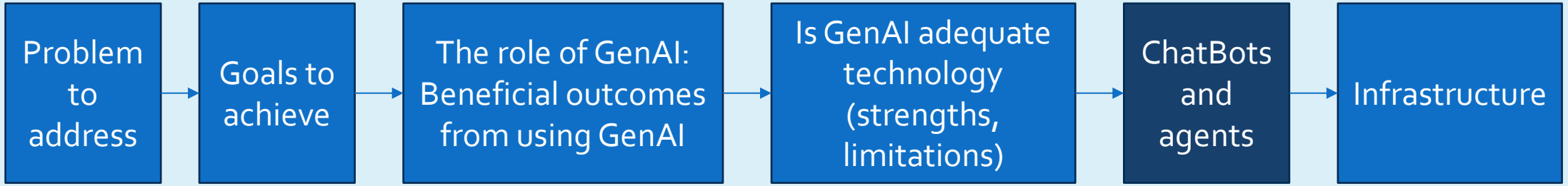
- GenAI adequacy: GenAI can perform the scaling, but the output is often hindered by hallucinations. To counter this, it is necessary to provide as many details as possible (description of the image) and the possibility to fix generated images.



SOLUTION DOMAIN

- A user shall provide as input a picture in a low resolution using a UI (web environment, hosted locally). The user will also create a description of the image to help with scaling. The backend (in python) shall take the image and the description. The backend shall generate a prompt and it will then call OpenAI using an API. OpenAI shall scale the image and return it to the UI. UI shall show the result and enable downloading of the upscaled image. In case if the result is not adequate, the user shall have an opportunity to request a change as a separate input.

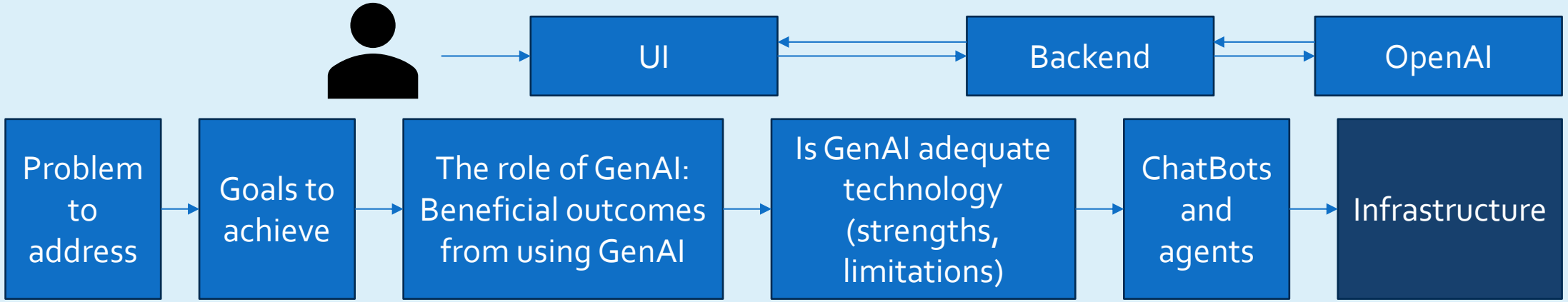




SOLUTION DOMAIN

- Should the model run locally?
- Are there privacy concerns?
- Price per token?
- Which model is adequate?

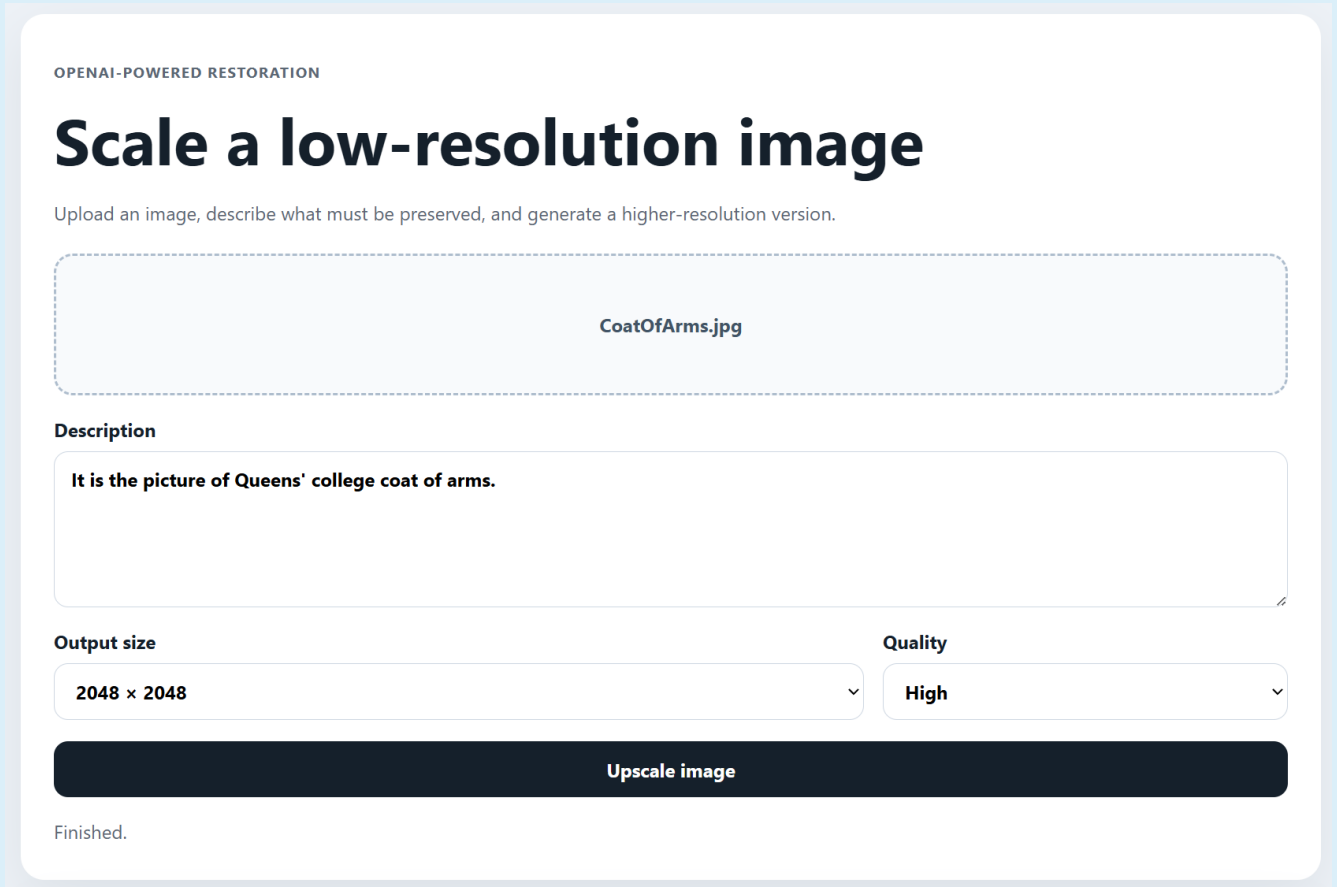
- Is there an adequate algorithmic solution that can solve some challenges in the context of the main problem?
- E.g., I need a solution that minimizes the use of tokens.



PROMPT

- Problem: there is a user of the application. The user has a lot of low-resolution images. These images are not suitable for presentations that user is creating.
- Goal: I need to create an application that enables the user to upload small-resolution pictures and receive high-resolution pictures in return.
- The role of GenAI: GenAI should do the scaling of images.
- GenAI adequacy: GenAI can perform the scaling, but the output is often hindered by hallucinations. To counter this, it is necessary to provide as many details as possible (description of the image) and the possibility to fix generated images.
- A user shall provide as input a picture in a low resolution using a UI (web environment, hosted locally). The user will also create a description of the image to help with scaling. The backend (in python) shall take the image and the description. The backend shall generate a prompt and it will then call OpenAI using an API. OpenAI shall scale the image and return it to the UI. UI shall show the result and enable downloading of the upscaled image. In case if the result is not adequate, the user shall have an opportunity to request a change as a separate input.

RESULT

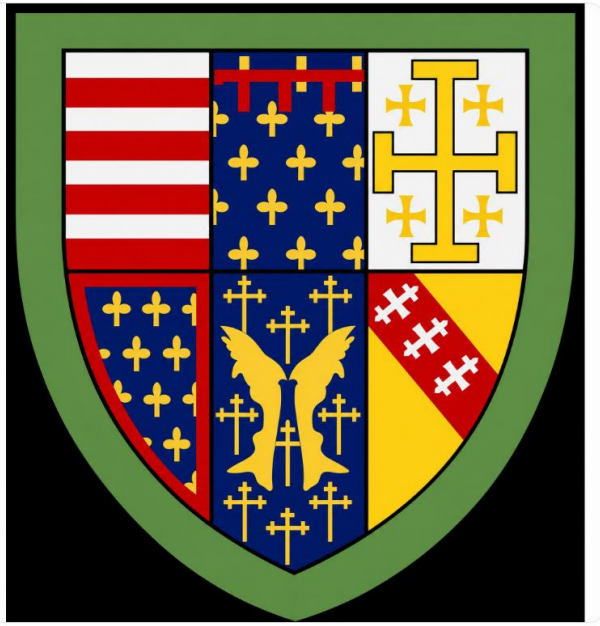

OPENAI-POWERED RESTORATION
Scale a low-resolution image
Upload an image, describe what must be preserved, and generate a higher-resolution version.
CoatOfArms.jpg
Description
It is the picture of Queens' college coat of arms.
Output size
2048 × 2048
Quality
High
Upscale image
Finished.

RESULT

Preview

Original

Upscaled result



Download result

TODOS

- Supervision Day 2: 22 July 2026
- Prepare slides regarding your presentation (mini research project)
- 15 minutes per person + some time for questions

TODOS

- Select your project
- You will receive an email with the confirmation
- https://docs.google.com/forms/d/e/1FAIpQLSeJSRu4SN_nrJ_1TiW9WoFKccslliD4bew5tLiQLuOz6sxyeQ/viewform?usp=dialog
- https://tinyurl.com/5dvcb6ra