LEUPHANA
UNIVERSITÄT LÜNEBURG

Simulation von KI Agenten

Projekt Softwareentwicklung
Major Wirtschaftsinformatik (ab Studienbeginn WiSe 20/21)
~ 4 Semester

Week 1

Prof. Dr. Ricardo Usbeck

- B.Sc./M.Sc. in CS at the University Halle-Wittenberg
- Ph.D. in CS at Leipzig University, Worked at Start-Ups and SMEs in DE and US
- PostDoc at University Paderborn and Fraunhofer IAIS Dresden
  - Team Lead Data Science and Smart Speakers
- Assistant Professor (Juniorprofessur) at Hamburg University
  - Semantic Systems
- Full-Professor at Leuphana University
  - Information Systems, in particular AI and Explainability
- Lead several European and German projects: DIESEL, QAMEL, LIMBO, SOLIDE, SPEAKER - recently COYPU, NFDI4DS, RescueMate
- Research interests: Knowledge Extraction, Question Answering Systems, Dialog Systems, Chatbots
- https://www.inf.uni-hamburg.de/sems
- https://www.leuphana.de/aix

Debayan Banerjee, M. Sc.

- *Akademischer Rat* (Academic Advisor) in the AIX group with Ricardo
- Currently Ph.D. student UHH in the area of NLP and Knowledge Graphs
- M.Sc. in Informatik from University of Bonn
- Bachelor in Information Technology from National Institute of Technology, Durgapur, India
- 7 years of industry experience with 3 years running own startup (gazemetrix.com)
- email: debayan.banerjee@leuphana.de
- Room: C4.318 (meet with prior appointment)
- https://debayan.github.io/

Research Group AIX

Goal: Machine-understandable, and, thus, explainable AI approaches

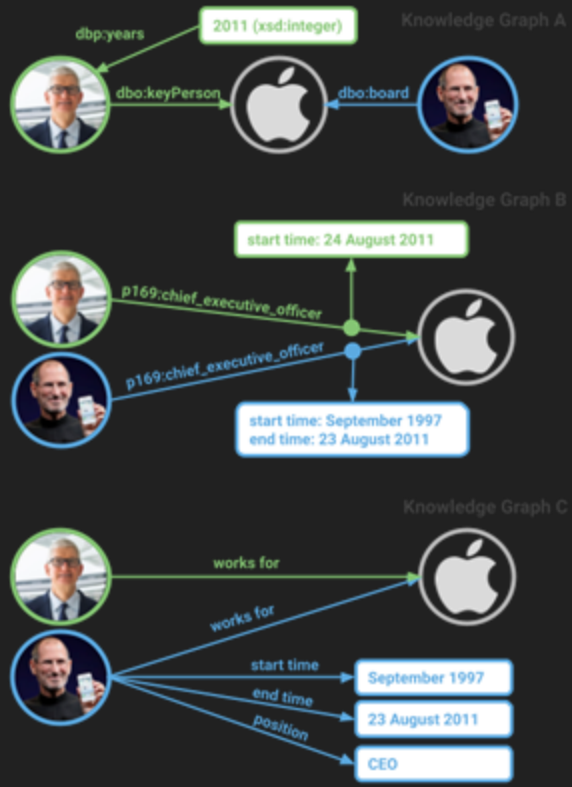
- Knowledge Graphs / Semantic Web / Linked Data
- Knowledge Extraction from unstructured and semi-structured data
- Question Answering over Knowledge Graphs, Speech Assistants
- Knowledge Validation, AI Ethics, AI for Sustainability
- FAIR Benchmarking, Research Data Management

Updates:

- https://twitter.com/Ricardo_Usbeck

Code:

- https://github.com/semantic-systems



That was us, who are you? What was your last project?

Organisatorisches - Inhalt

- **Inhalte:** In diesem Software-Projekt-Kurs werden die Studierenden die spannende Welt der agentenbasierten Simulationen erkunden und ihre Fähigkeiten im Bereich der Softwareentwicklung, Modellierung und Teamarbeit vertiefen. Der Fokus des Kurses liegt auf der **Entwicklung von agentenbasierten Simulationssystemen**, bei denen autonome, KI-basierte Agenten in einer virtuellen Umgebung agieren und miteinander interagieren um ein Ziel zu erreichen. Die Studierenden werden in Teams arbeiten, um solche Simulationen selbst zu entwerfen, zu implementieren und am Ende des Semesters zu präsentieren.
- **Ziel:** Die Studierenden lernen **Teamarbeit** und Rollenverteilung kennen, während sie ein sich weiterentwickelndes System weiter implementieren. Für einen gegebenen Anwendungsfall, können sie gezielt einen **Methodenkatalog auswählen und Implementierungs- sowie Evaluationsvorschläge** präsentieren. Sie sind in der Lage Anwender*innen in der Umsetzung ihrer fachlichen Frage zu unterstützen.

Organisatorisches - Ablauf

- Format = 2x 90 mins pro Woche (14+14 Meetings)
- Semester: 2. April 2024 - 5. Juli 2024
- Zeit:
  - Montag (Seminar Ricardo) 08:15 - 09:45
  - Donnerstag (Seminar Debayan) 14:15 - 15:45
- Raum: C 25.021 Seminarraum
  - Mo, 13.05.2024, 08:15 - 09:45 | C 14.204 Seminarraum
  - Do, 20.06.2024, 14:15 - 15:45 | C 14.204 Seminarraum
  - Do, 27.06.2024, 14:15 - 15:45 | C 14.204 Seminarraum
- Material und Kommunikation via mystudy

Organisatorisches - Prüfung

Studienleistung: Eine Zusammenarbeit in 2er- bis 6er-Teams ist gestattet. Projektvorschläge von Seiten der Studierenden werden in den ersten Seminaren besprochen und mit dem Lehrpersonal abgestimmt.

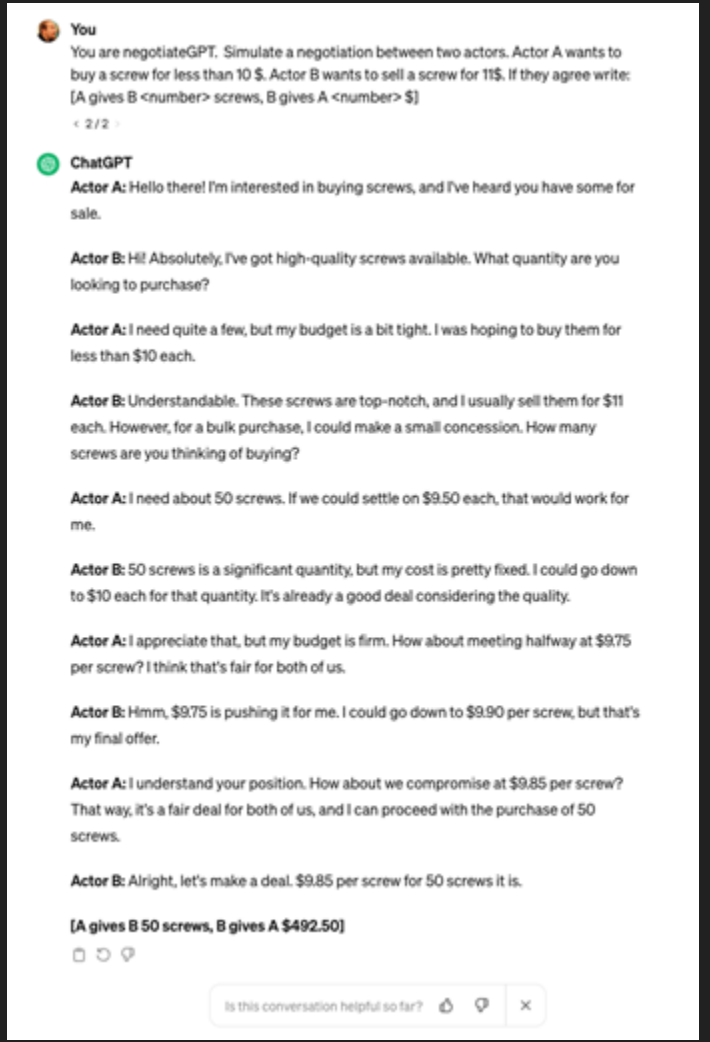
Prüfungselemente:

- Software-Projekt mit Präsentation (30%), Projektdokumentation (70%) ~ Funktion und Benchmarks von Code als ReadMe.md im Repo
- Präsentation des Projektes in der letzten VL Woche
- Abgabe Dokumentation am letzten Tag der VL Woche

Wenn Sie eine bestimmte Note haben wollen, sagen Sie es vorher, dann prüfen wir entsprechend

Existierende Projekte

ChatGPT + Prompting



You

You are negotiateGPT. Simulate a negotiation between two actors. Actor A wants to buy a screw for less than 10 $. Actor B wants to sell a screw for 11$. If they agree write: [A gives B <number> screws, B gives A <number> $]

‹ 2 / 2 ›

ChatGPT

Actor A: Hello there! I'm interested in buying screws, and I've heard you have some for sale.

Actor B: Hi! Absolutely, I've got high-quality screws available. What quantity are you looking to purchase?

Actor A: I need quite a few, but my budget is a bit tight. I was hoping to buy them for less than $10 each.

Actor B: Understandable. These screws are top-notch, and I usually sell them for $11 each. However, for a bulk purchase, I could make a small concession. How many screws are you thinking of buying?

Actor A: I need about 50 screws. If we could settle on $9.50 each, that would work for me.

Actor B: 50 screws is a significant quantity, but my cost is pretty fixed. I could go down to $10 each for that quantity. It's already a good deal considering the quality.

Actor A: I appreciate that, but my budget is firm. How about meeting halfway at $9.75 per screw? I think that's fair for both of us.

Actor B: Hmm, $9.75 is pushing it for me. I could go down to $9.90 per screw, but that's my final offer.

Actor A: I understand your position. How about we compromise at $9.85 per screw? That way, it's a fair deal for both of us, and I can proceed with the purchase of 50 screws.

Actor B: Alright, let's make a deal. $9.85 per screw for 50 screws it is.

[A gives B 50 screws, B gives A $492.50]

Is this conversation helpful so far?

Existierende Projekte

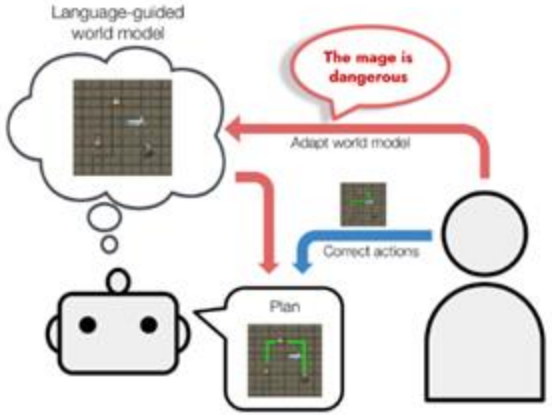
https://language-guided-world-model.github.io/

Language-Guided World Models A Model-Based Approach to AI Control

Alex Zhang*†, Khanh Nguyen*‡, Jens Tuyls†, Albert Lin†, Karthik Narasimhan†
Princeton University† & UC Berkeley‡
*Indicates Equal Contribution

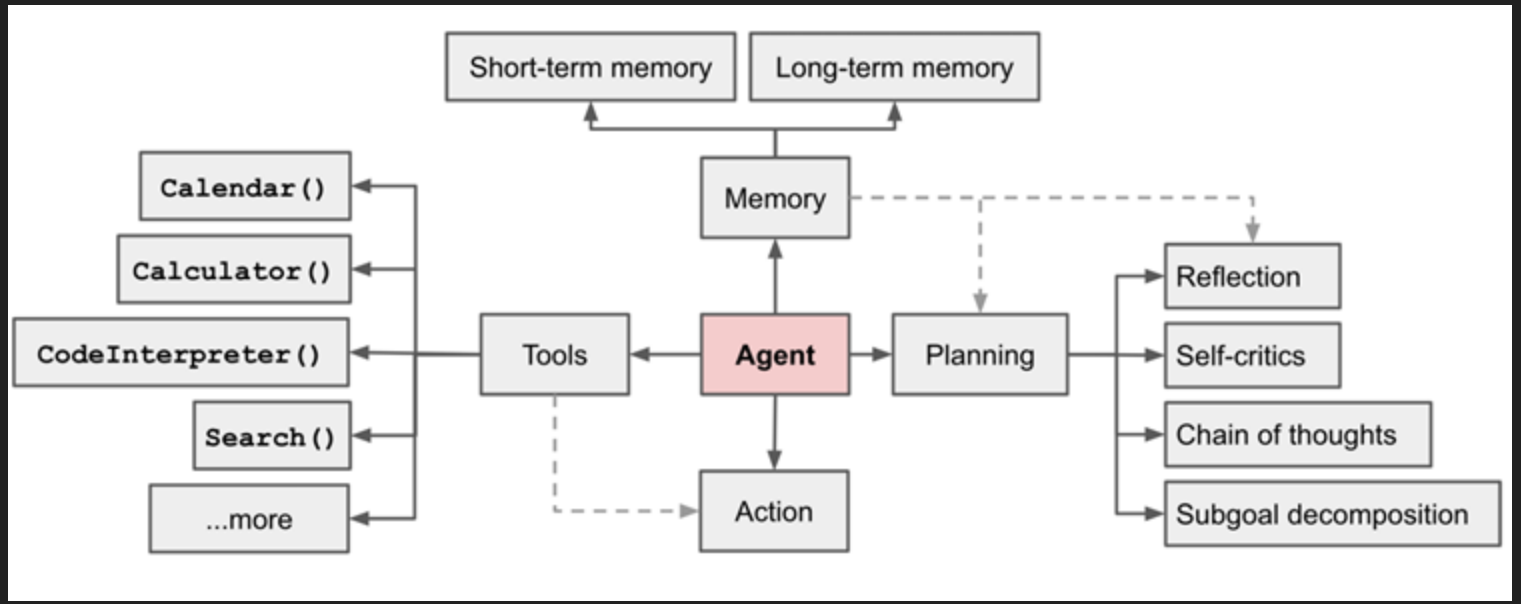
Paper | arXiv | Code

We develop world models that can be adapted with natural language. Intergrating these models into artificial agents allows humans to effectively control these agents through verbal communication.



Existierende Projekte

LLM Powered Autonomous Agents | Lil'Log - Introduction to Agents



Existierende Projekte

[a16z-infra/ai-town: A MIT-licensed, deployable starter kit for building and customizing your own version of AI town - a virtual town where AI characters live, chat and socialize.]

Existierende Projekte

https://store.steampowered.com/app/2521630/Mini_Settlers/

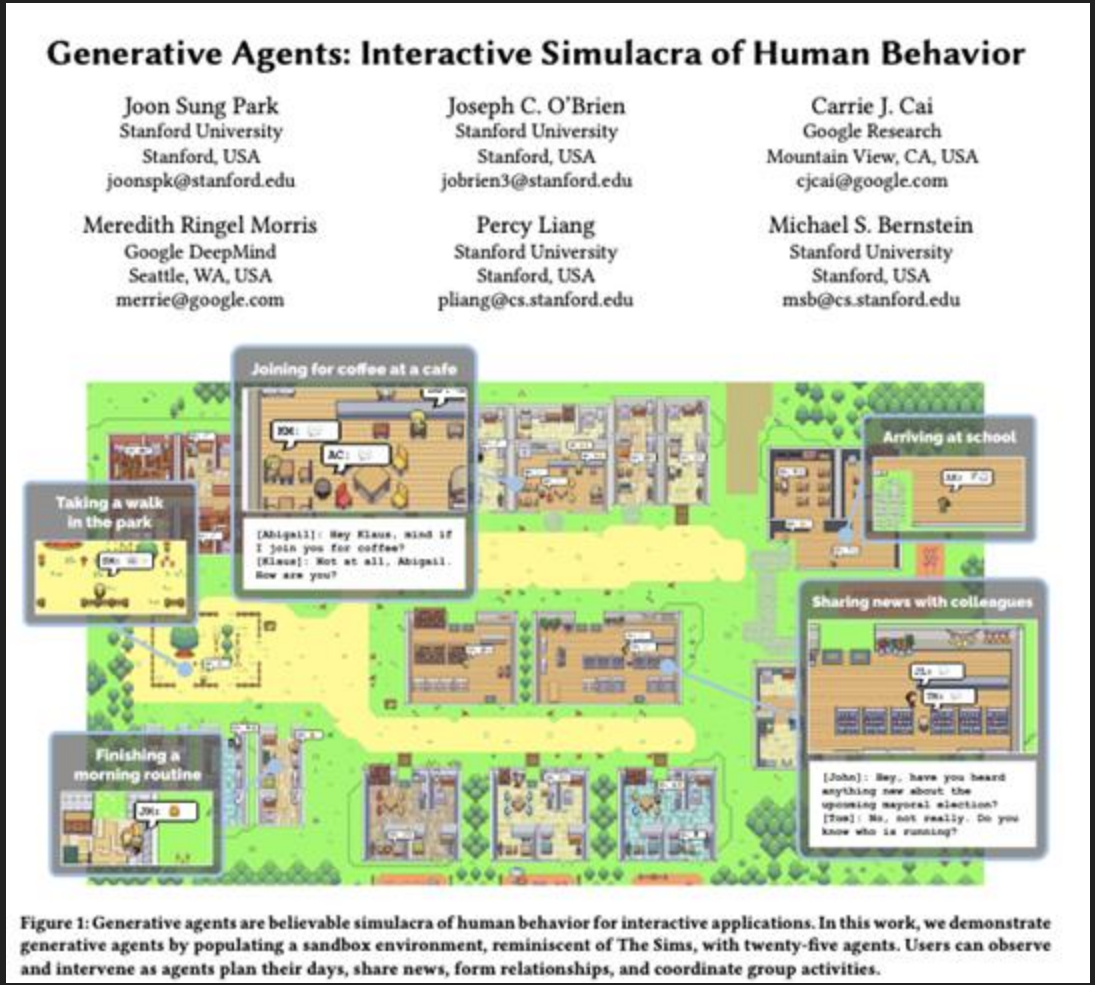
Existierende Projekte

Park, Joon Sung, et al. "Generative agents: Interactive simulacra of human behavior." Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology. 2023. Paper Link

Demo

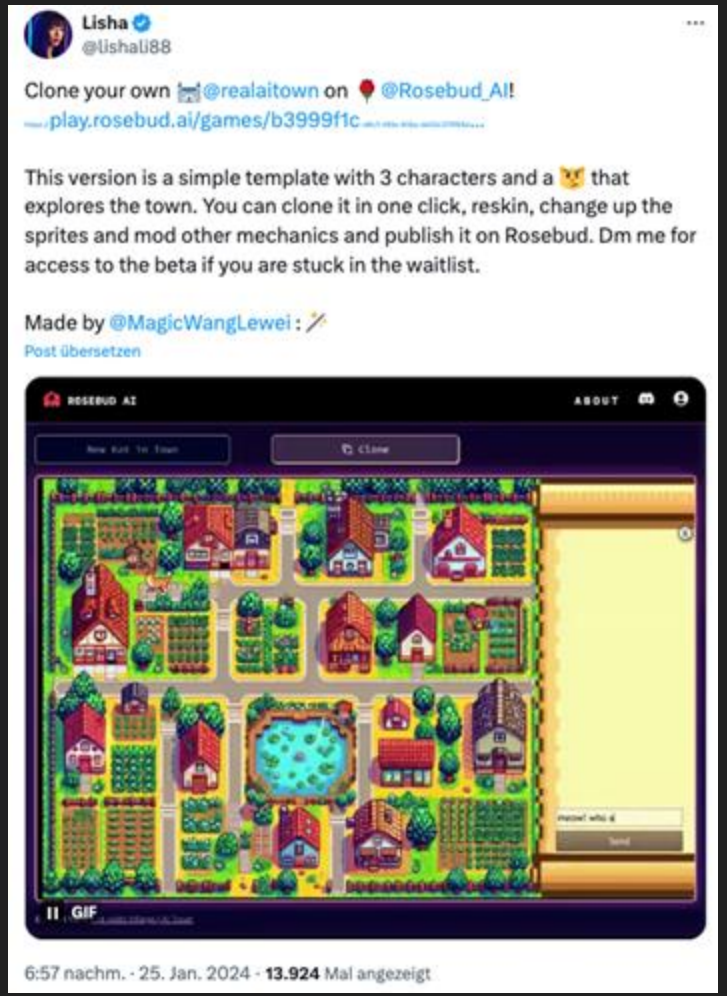
Source Code

Working Code



Existierende Projekte

- AI Town Clone
- https://twitter.com/lishali88/status/1750578635101344252



Scenario / Task / Challenge

Was würden Sie tun wollen?

Wie soll die Welt aussehen?

Vorschlag: Web-basierte Welt mit Logging Output zur Inspektion der Agenten

Wie kooperativ/kompetitiv soll es sein?

Vorschlag: Challenges über geteilte Szenarien (bspw. Netzwerk von Firmen), die von allen Teams entwickelt werden können. Zielsetzung festlegen, bspw. wie weit Informationen gekommen sind oder ob bestimmte Agenten ein Ziel (Trade) erreicht haben.

Was sind die Minimalanforderungen?

Vorschlag: Umgang mit einem LLM und zeigen, dass es besser funktioniert als andere Versionen.

Alternative Scenario (Deadline 08.04.2024)

World:

- Grid-World with Birds-Eye-View

Task: Trade or Information Spread Party) or?

Metric/Evaluation: ?

Communication Style: 1:1 or 1:1:broadcast

Minimal Acceptance Criteria:

- 2 agents
- “UI” to compare worlds
- Evaluation of the agents behaviour

Was würden Sie tun wollen?

Wie groß sollen die Gruppen sein?

Vorschlag: 4-6 Personen pro Gruppe

Homework: We would like to get to know you better

Fill out that questionaire and send it to us by Sunday Evening 6pm via Email (Ricardo):
https://csp.uber.space/phhd/umfrage-zu-semesterstart.pdf

Week 2

Technical Setup

HTML Web Pages

html Copy code

```html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Minimal HTML File</title>
</head>
<body>

<h1>Hello, World!</h1>

</body>
</html>
```

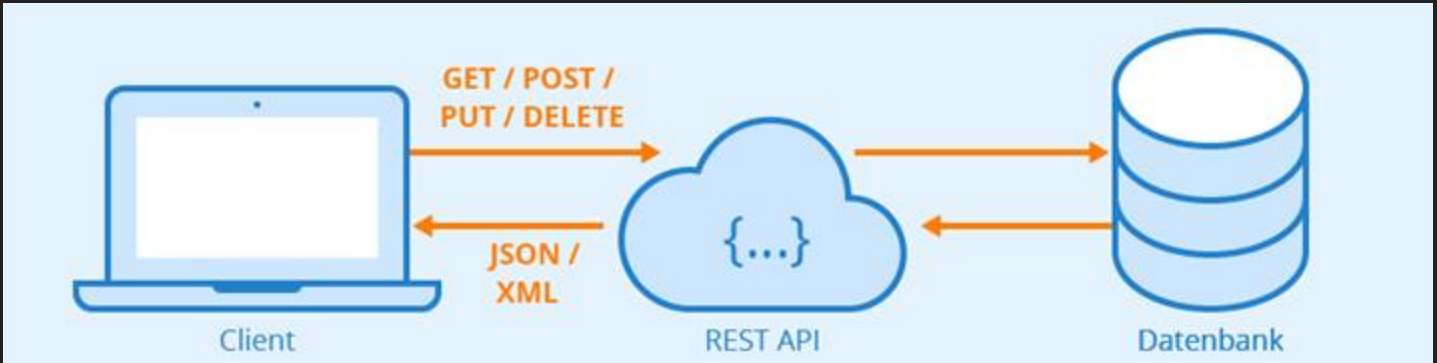
HTML Web Pages

```html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>HTML Page with Flask Backend</title>
</head>
<body>

<h1>Communication with Flask Backend</h1>

<button onclick="getData()">Get Data from Flask</button>
<div id="output"></div>

<script>
function getData() {
    fetch('/get_data')
    .then(response => response.json())
    .then(data => {
        document.getElementById('output').innerText = JSON.stringify(data);
    });
}
</script>

</body>
</html>
```

HTML Web Pages

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>HTML Page with Flask Backend</title>
</head>
<body>

<h1>Communication with Flask Backend</h1>

<button onclick="getData()">Get Data from Flask</button>
<div id="output"></div>

<script>
function getData() {
    fetch('/get_data')
    .then(response => response.json())
    .then(data => {
        document.getElementById('output').innerText = JSON.stringify(data);
    });
}
</script>

</body>
</html>
```

What is an API?



https://www.seobility.net/de/wiki/images/f/f1/Rest-API.png

What is an API?

python

Copy code

```python
from flask import Flask, jsonify

app = Flask(__name__)

@app.route('/')
def index():
    return app.send_static_file('index.html')

@app.route('/get_data')
def get_data():
    data = {'message': 'Hello from Flask backend!'}
    return jsonify(data)

if __name__ == '__main__':
    app.run(debug=True)
```

What is an API?

https://flask.palletsprojects.com/en/2.3.x/quickstart/

What is an API?

index.html | backend.py | README.md

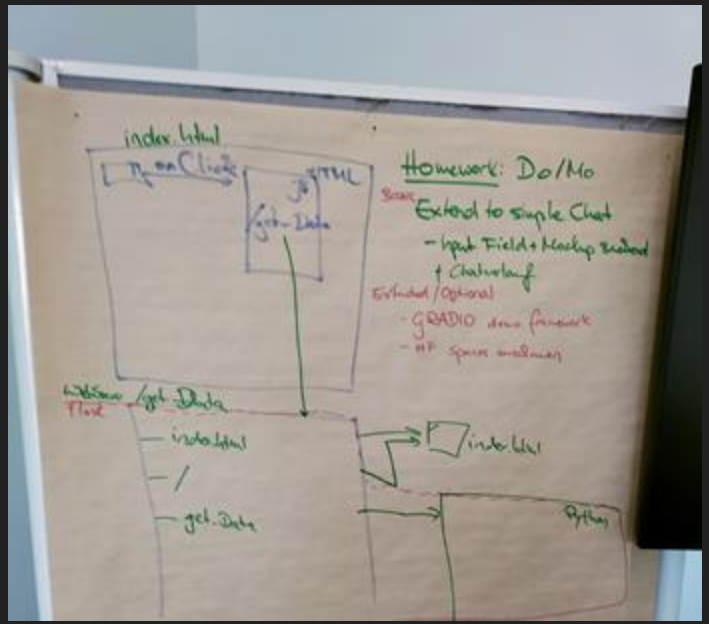
```python
from flask import Flask
from flask import render_template
from flask import jsonify

# creates a Flask application
app = Flask(__name__)


@app.route("/")
def hello():
    message = "Hello, World"
    return render_template('index.html',
                           message=message)

@app.route('/get_data')
def get_data():
    data = {'message': 'Hello from Flask backend!'}
    return jsonify(data)

# run the application
if __name__ == "__main__":
    app.run(debug=True)
```

https://flask.palletsprojects.com/en/2.3.x/quickstart/

What is an API?

```
from flask import Flask
from flask import render_template
from flask import jsonify
# creates a Flask application
app = Flask(__name__)
@app.route("/")
def hello():
    message = "Hello, World"
    return render_template('index.html',
                           message=message)
@app.route('/get_data')
def get_data():
    data = {'message': 'Hello from Flask backend!'}
    return jsonify(data)
# run the application
if __name__ == "__main__":
    app.run(debug=True)
```

Homework for week 3



Git

```
# mkdir project    (creates project directory)
# cd project/      (go inside the project directory)
# touch code.py    (creates a file named code.py, add code in an editor and save)
# git init         (starts a git repository in this folder, locally)
# git add code.py (tells git to consider this file next time you commit)
# git commit -m “My first commit”    (commits this file to git)
# git log  (show all commits)
# git status  (show which files have changes, have been added, not added)
```

Now go and create a project on github.com or gitlab on academiccloud.de
Follow the instructions there to add a remote origin to your local git folder.
Then every time you do a ‘git commit’ follow with a ‘git push origin main’.

Make sure you regularly commit and push code to protect from accidental loss of data.

Week 3

Monday only until 9.30 am

Homework Status - How is your demo looking?

- 3,4 funktionierende Demos
- Wir werden eine funktionierende Demo auf GitHub einstellen

Group Status - Has everyone found a group?

- Form group tables and get a piece of poster paper!

Task Definition

Is it clear?

Sketch a UI! 10mins

Let’s discuss the UI, what modules are needed and add them to the task definition

Modus Operandi Seminar from our Perspective

Monday (Scrum update, What?):

- FAQ: Listen to problems, requests, comments
- Scrum:
  - Get project updates (2 mins) per team
  - Serve as Product Owner - ask me anything
- Discuss further task specifications

Thursday (Technical Session, How?):

- Teach something
- Answer questions
- Sit with people who need to solve problems with code/bugs

This is a project seminar - you must work, become aware of issues and ask us!

Scrum Short Introduction

- Read an introduction to Scrum:
  - https://www.atlassian.com/de/agile/scrum

Homework zu Montag Woche 4

Wireframe für die UI anlegen

Thursday 17.04.2024

Topics to cover today:

1. A working Flask application
2. A working Streamlit application

Topics covered:

POST/GET, HTTP/HTTPS, gitlab, web servers, JSON

https://gitlab.gwdg.de/AIX/simulation-ki-agent-2024

Task Definition

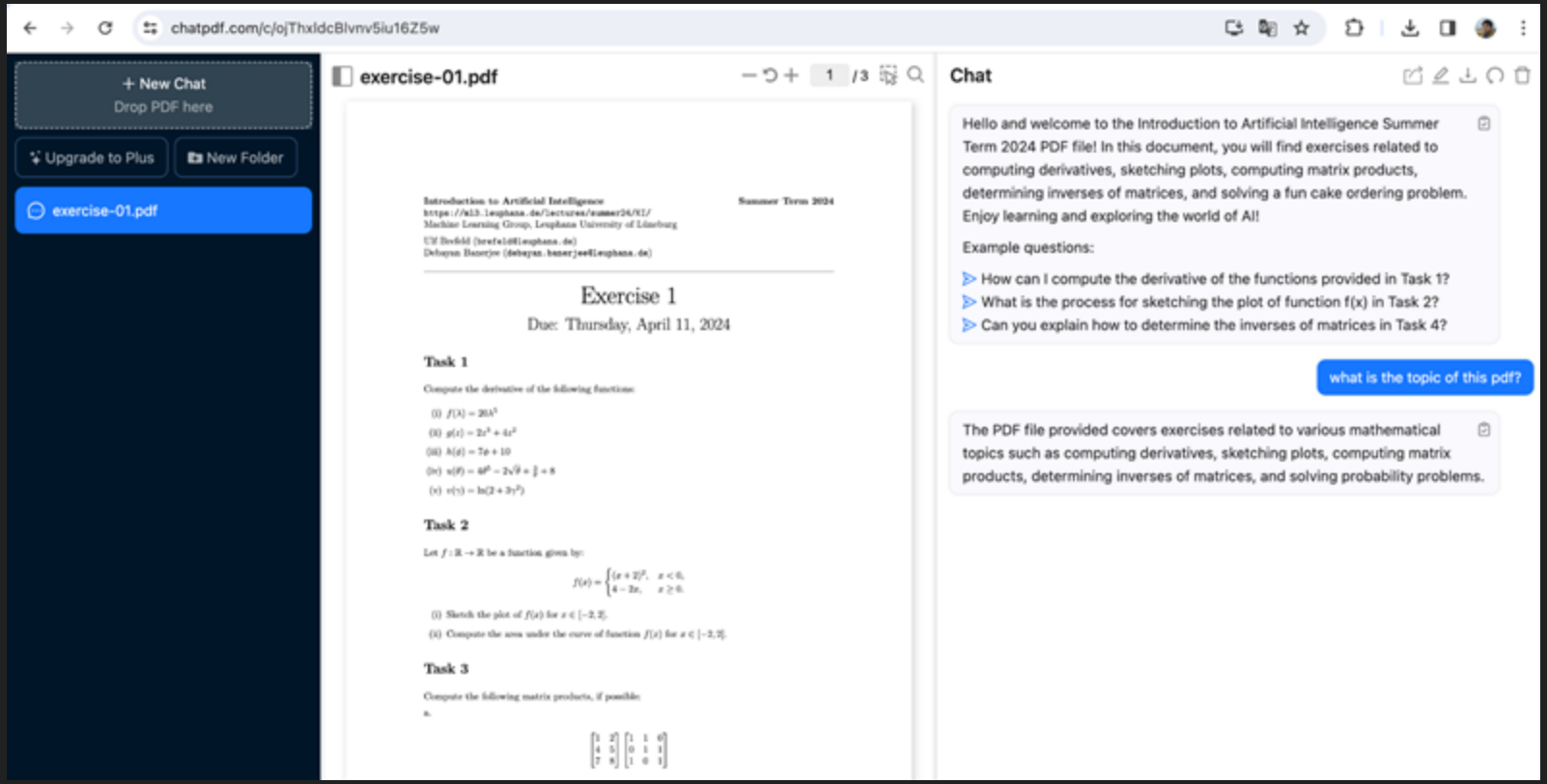
Task Definition: Chat to PDF (Sustainability Reports)

- Upload a sustainability report via PDF in a webU I
- Use an LLM to chat with the PDF and extract data
- Enable logging of the extracted contents (e.g. amount of CO2, waste strategy)
- Download logs as JSON

Data Source: 100 preprocessed sustainability reports, Repository

Paper: https://arxiv.org/abs/2310.05628

Task: Chat with a PDF (Example UI)



Task: Chat with a PDF (Example Architecture)

Pieces required?

1. Frontend (HTML/CSS/javascript) (Bootstrap)
    a. Upload PDF
    b. Chat interface (Gradio/Streamlit)
2. Backend
    a. Accept pdf upload
    b. Convert pdf to text (python library)
    c. Load LLM
    d. Use pdf text as context and answer chat using LLM (RAG)

        Retrieval Augmented Generation (python libraries)

Week 4

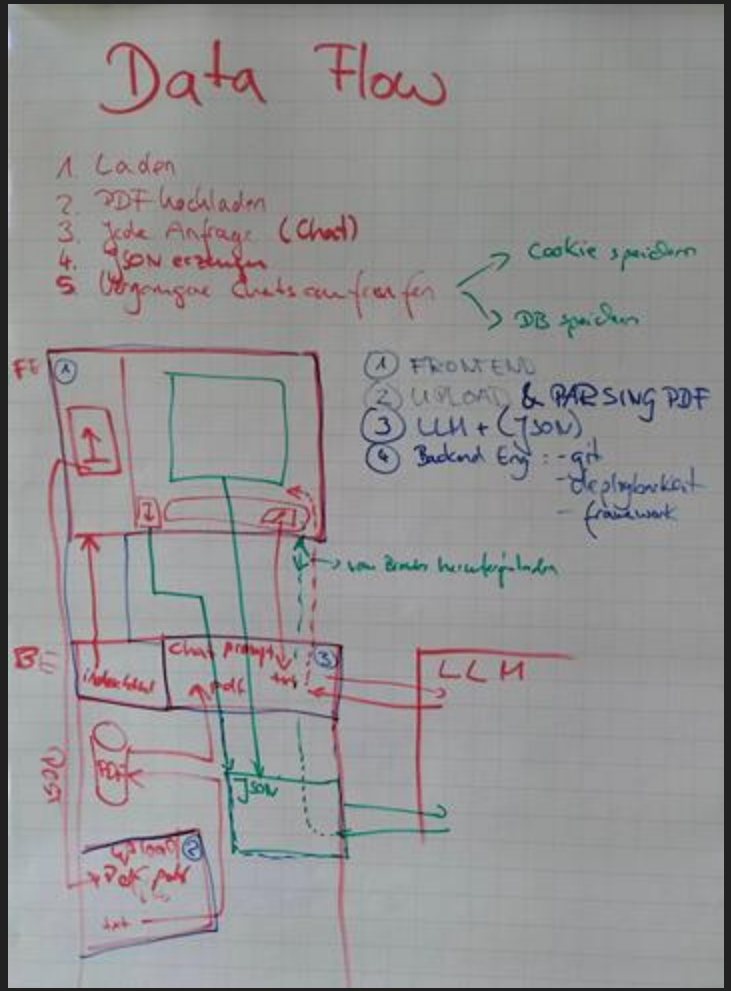
Milestones

Milestone	Features	Description	Dates
1	Basic web demo	Consists of a working frontend and backend, where information exchange takes place (POST request). Code must be on gitlab/github, with install instructions in a readme file. Every team member must be able to show a running demo on their machine.	6th May 2024
2	Backend talks to LLM	Modify your frontend to accept an input text. Send it to backend. The backend should send it to an LLM with an instruction, and send the result back to the frontend.	
3	Frontend accepts file upload	Modify frontend to allow a PDF upload, send it to backend. Convert PDF to text in backend and send it to frontend and display the text.	
4	Chat interface	Now your frontend should do the following: 1. Accept PDF upload 2. Present a chat interface 3. Working chat (not necessarily with LLM or PDF)	
5	PDF Chat + JSON Download	Allow the chat to take place with an LLM where the user can ask questions about the content of the PDF. Use an LLM to produce a downloadable JSON version of the chat. Well-written README.	

Homework Status - How is your UI wireframe looking?

- 1) Discuss wireframes
- 2) Which features are needed? What is the data flow for each feature, i.e., which functions are needed?
- 3) In which order do you want to do them, why?

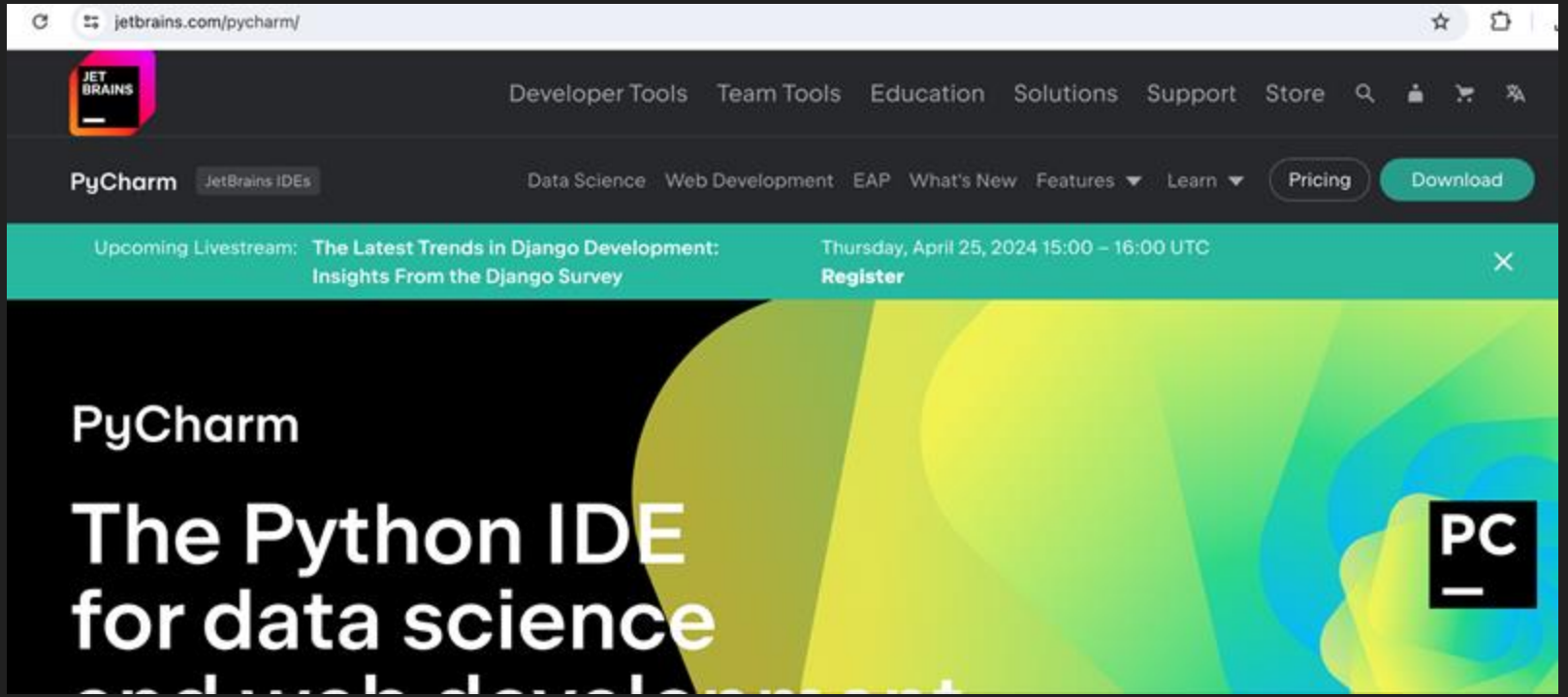
Dataflow & Roles



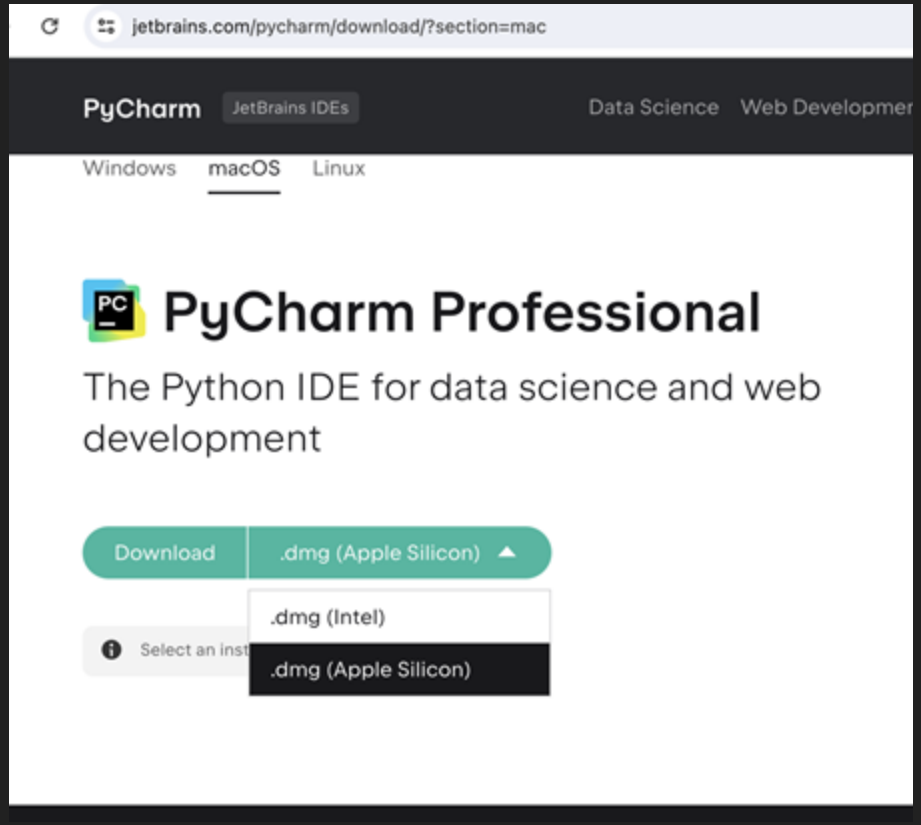
Timeline

Week 1	-	Introduction to the challenge (Today)
Week 2	Full-Stack Architecture, HTML, Flask	Git, Task Introduction
Week 3 🤠	Status Demo, Status Teams, Definition of Task, Modus der Veranstaltungen	Python, CURL, APIs via Python - Where do you need the biggest help technically?, JSON in Python read and write, Flask, IDE
Week 4	Discussion of Wireframes, Data Flow, Roles	Streamlit, RAG, Help with demos
Week 5	Show your demo!	
Week 6	MS1 - running web demo	
Week 7		
Week 8	Mid-Term Presentations	
Week 9		
Week 10		
Week 11		
Week 12		
Week 13		
Week 14	Final Presentations	

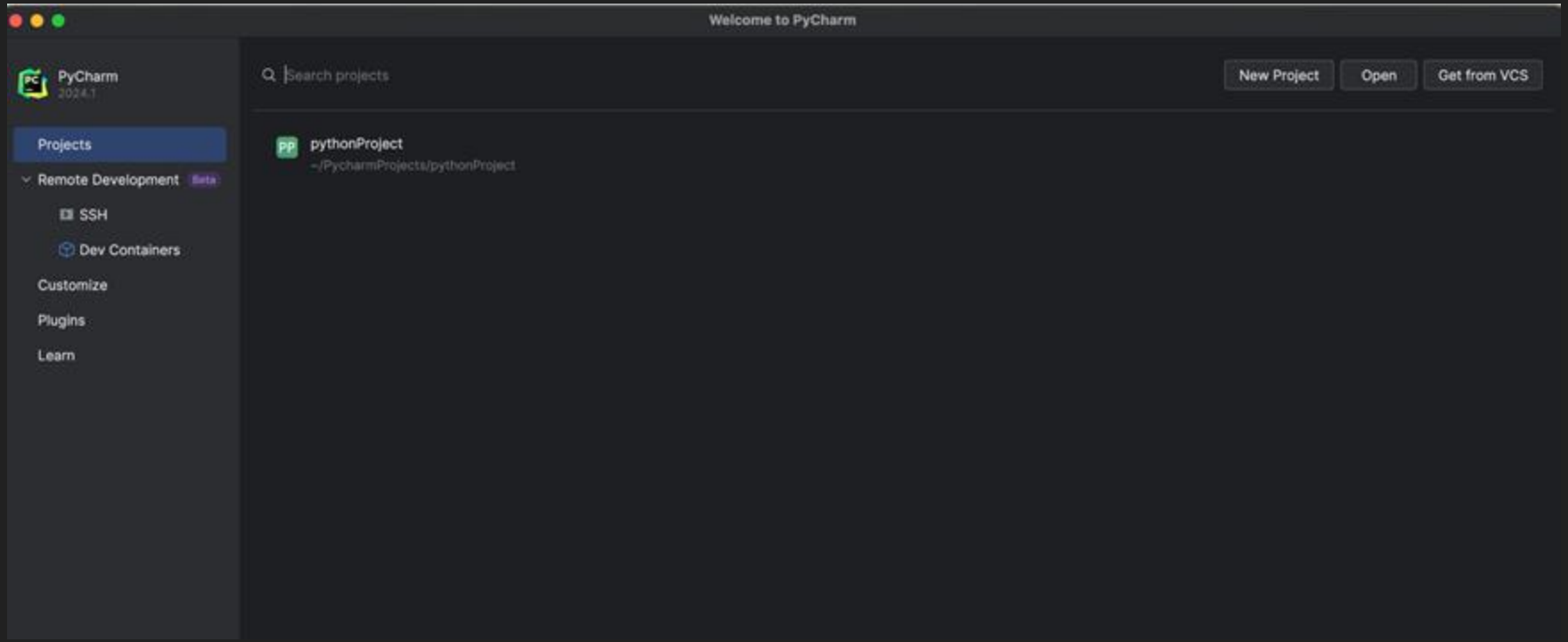
Using Co-Pilot with PyCharm



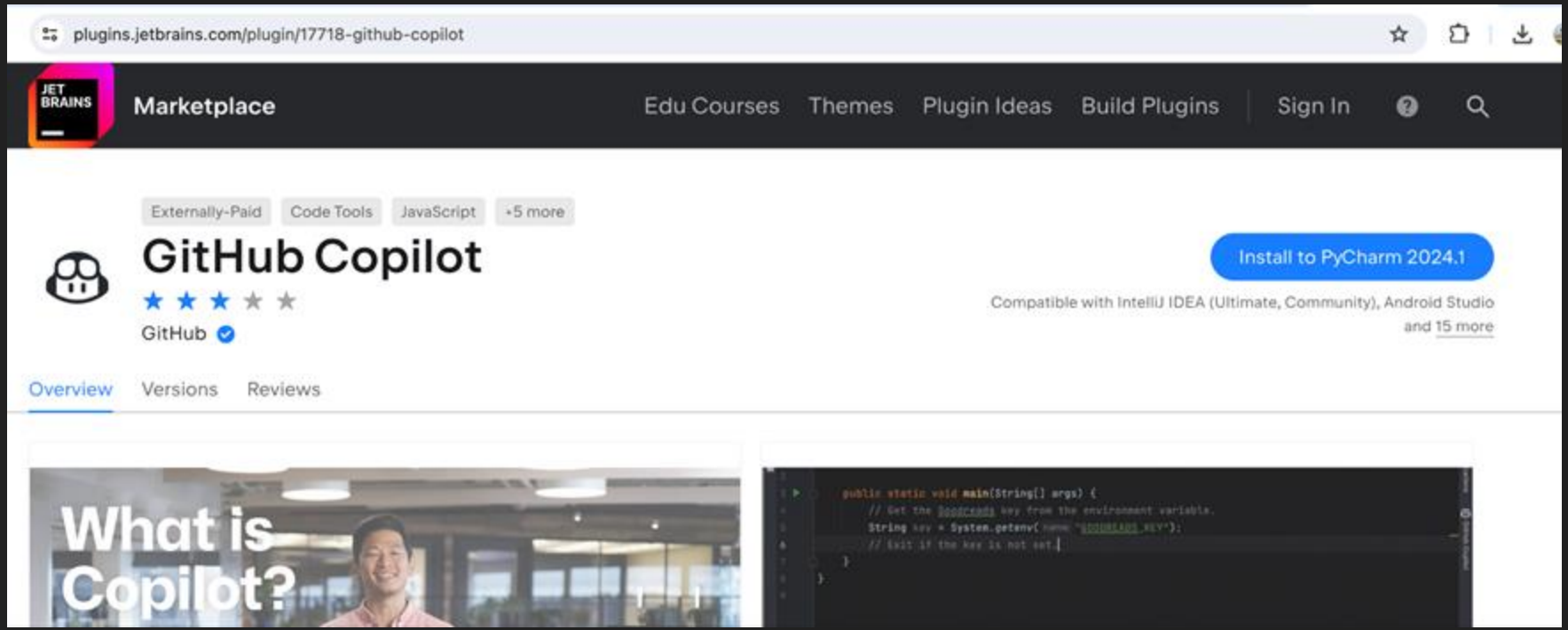
Download and Install PyCharm



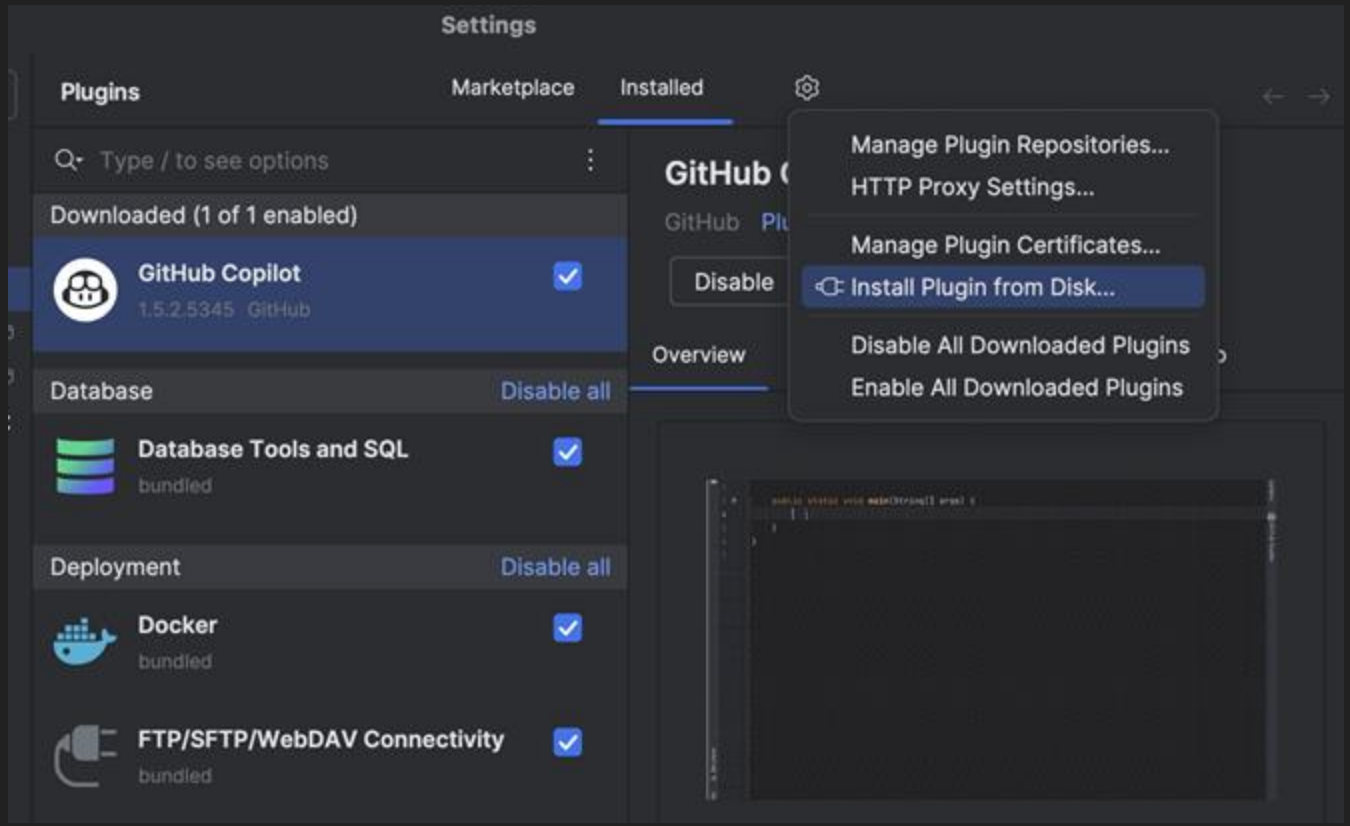
Run PyCharm


Welcome to PyCharm
PyCharm
2024.1
Search projects
New Project
Open
Get from VCS
Projects
Remote Development
Beta
SSH
Dev Containers
Customize
Plugins
Learn
pythonProject
~/PycharmProjects/pythonProject

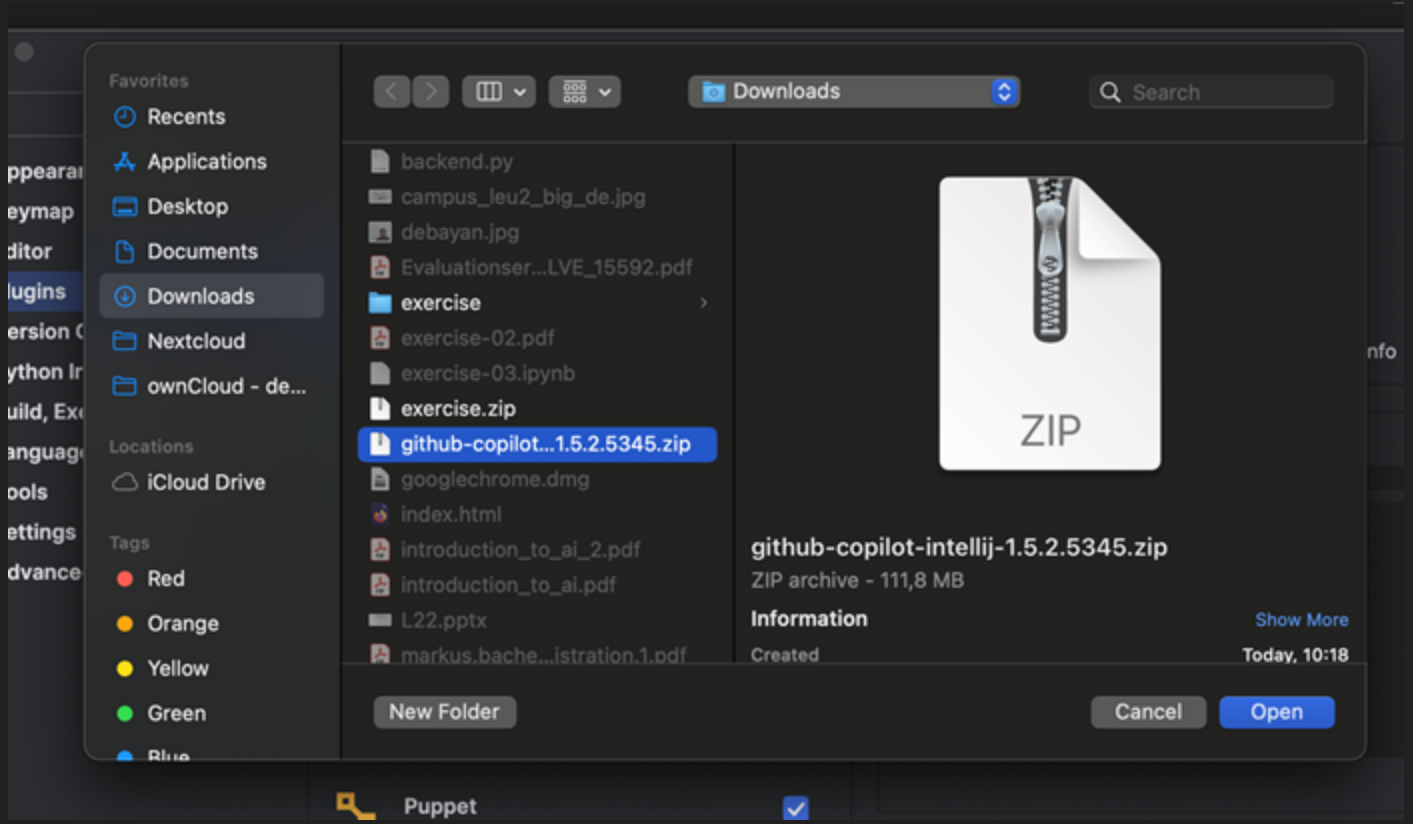
Download PyCharm Github Copilot Plugin



Open PyCharm Settings (Command + ,) on macOS



Select downloaded plugin and install

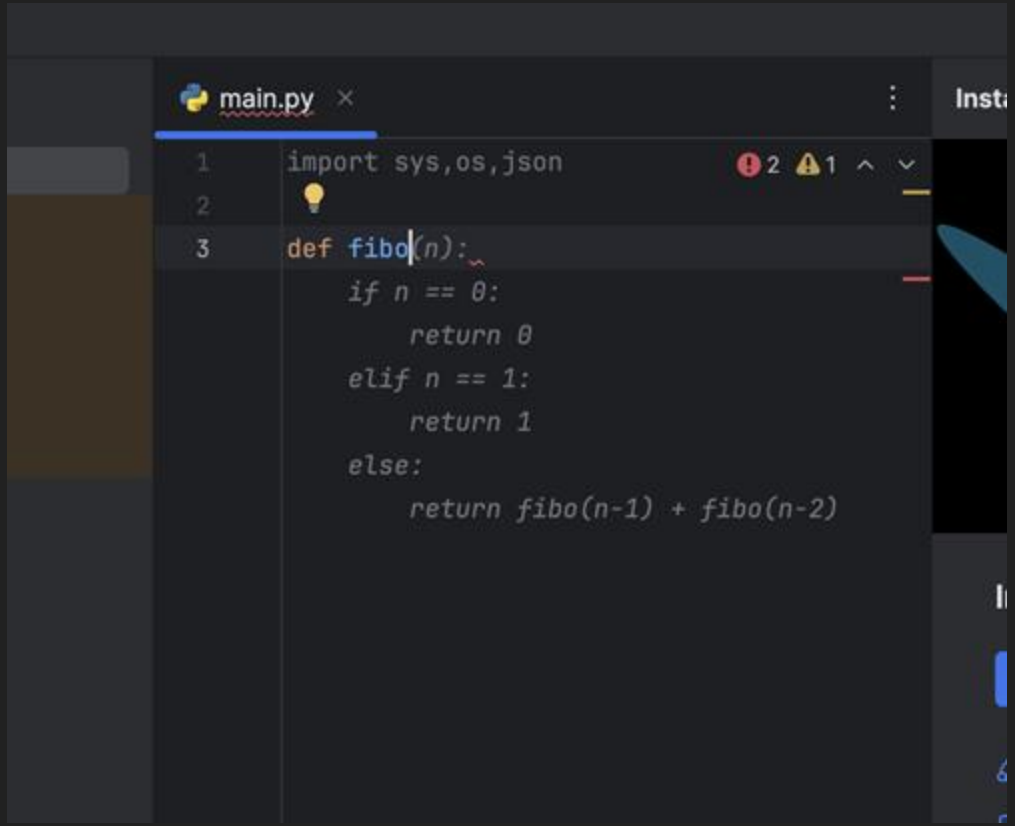


Activate Github Copilot Account

This varies from person to person.

Normally, PyCharm will give you prompts with links to click and activate your copilot account.

Start new project and use Copilot



Streamlit Tutorial

Summary of 25.04.2024

1. Presented github copilot + pycharm integration.
    a. follow https://education.github.com/pack#:~:text=Free%20access%20to%20Copilot%20Individual,button%20to%20claim%20the%20offer. for instructions on how to enable student pack on github.
2. Streamlit tutorial with chat interface and chatGPT integration. Students have been told that each group will be provided with an OpenAPI key which they must NOT commit to git. Secret keys and API keys must remain on disk and not in public.
3. Students needed help with setting up public-private keys and adding it on github for cloning and pushing to work.
    a. see https://jdblischak.github.io/2014-09-18-chicago/novice/git/05-sshkeys.html for instructions on how to.

Week 5

Student groups as of 29.04.2024

Die Drei Musketiere
Mika
Walid
Jonathan

Team 1.0
Tolga - PDF2Text
Noure - Der Rest
Tom - FE
Max - BE
Albion - FE
Nawid - PDF2Text

Team North
Roman
Ali
Aichan
Refki

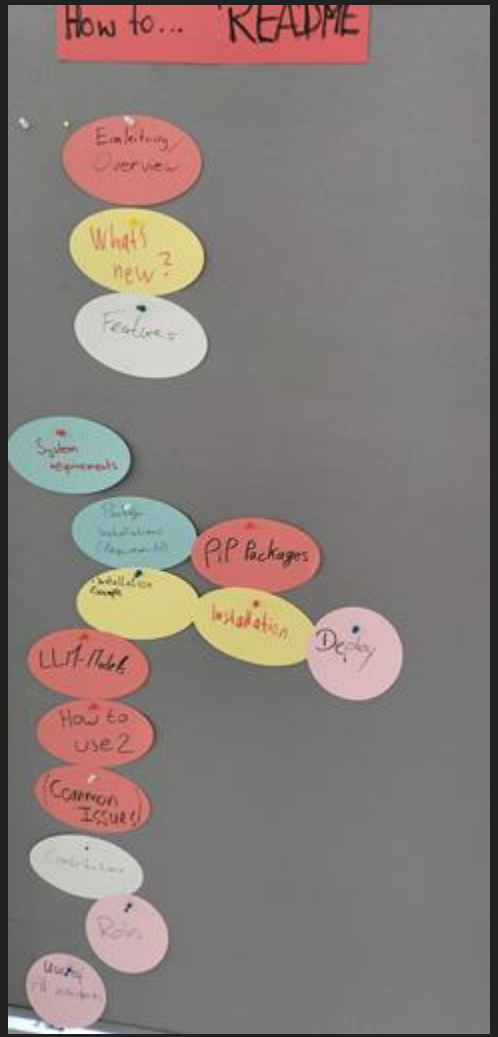
PyCharming
Alexander - BE&LLM
Timo - PDF2Text
Jakob - FE
Leo - FE
Oliver - BE&LLM
Jonne - BE&LLM

Algorithms-Alchemisten
Gerrit - FE
Friedrich - FE
Leon
Julian - PDF2Text
Raoul - BE
Kjell

Team Sunrise
Andrea
Anusha
Ivo
Fabian
Ahmad

What must go into the ReadMe?

- Description of functionality (descriptive text and diagram and screenshot)
- Roles (Name + 2-3 sentences what the person does)
- How to deploy?
- What can it do?
- Use of AI assistants

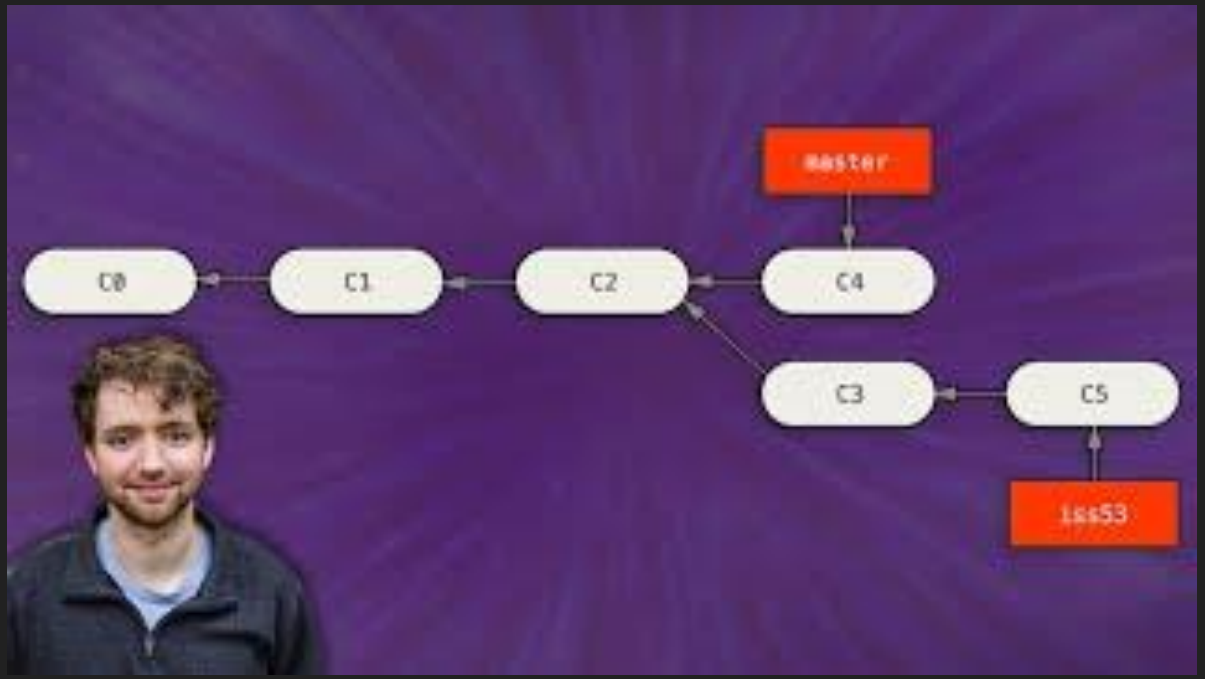


Show your demo

Expectation for May 6th

Basic web demo: Consists of a working frontend and backend, where information exchange takes place (POST request). Code must be on gitlab/github, with install instructions in a readme file. Every team member must be able to show a running demo on their machine.

02.05.2024 github fork, branch, remote



02.05.2024 github issues

02.05.2024 .gitignore

Timeline

Week 1	-	Introduction to the challenge (Today)
Week 2	Full-Stack Architecture, HTML, Flask	Git, Task Introduction
Week 3 🤠	Status Demo, Status Teams, Definition of Task, Modus der Veranstaltungen	Python, CURL, APIs via Python - Where do you need the biggest help technically?, JSON in Python read and write, Flask, IDE
Week 4	Discussion of Wireframes, Data Flow, Roles	Streamlit, Help with demos
Week 5	Show your demo! What must go into the ReadMe?	GitHub copilot + pycharm, ssh keys, streamlit + chatgpt tutorial
Week 6	MS1 - running web demo, Theory of LLMs	Github branch/fork/pull/push/issues/.gitignore
Week 7		HOLIDAY
Week 8	Mid-Term Presentations	TODO: RAG
Week 9		TODO: RAG
Week 10		
Week 11		
Week 12		
Week 13		
Week 14	Final Presentations	

Week 6

Student groups as of 06.05.2024

Die Drei Musketiere
Mika
Walid
Jonathan

Team 1.0
Tolga - PDF2Text
Noure - Der Rest
Tom - FE
Max - BE
Albion - FE
Nawid - PDF2Text

Team North
Roman
Ali
Aichan
Refki

PyCharming
Alexander - BE&LLM
Timo - PDF2Text
Jakob - FE
Leo - FE
Oliver - BE&LLM
Jonne - BE&LLM

Algorithms-Alchemisten
Gerrit - FE
Friedrich - FE
Leon
Julian - PDF2Text
Raoul - BE
Kjell

Team Sunrise
Andrea
Anusha
Ivo
Fabian
Ahmad

MS 1 Demo (Mon)day

Team 1 - North

- Entschuldigt

Team North
Roman
Ali
Aichan
Refki

Team 2 - PyCharming

PyCharming
Alexander - BE&LLM
Timo - PDF2Text
Jakob - FE
Leo - FE
Oliver - BE&LLM
Jonne - BE&LLM

- Vollfunktionsfähige Demo gemäß MS 1
- Flask-basierte Demo
- Bug: Seiten mit PDF und Chatbot links-rechts vertauscht, aber einfaches Vertauschen macht Sinn
- Bug: LLM Version soll noch von Link auf Dropdown umgestellt werden
- Bug: Scrollbar im Input-Feld
- Feature: Clear Button für Chat und PDF

- Feature: Google WebFonts

Team 3 - 1.0

Team 1.0
Tolga - PDF2Text
Noure - Der Rest
Tom - FE
Max - BE
Albion - FE
Nawid - PDF2Text

- Vollfunktionsfähige Demo gemäß MS 1
- Flask-basierte Demo
- Handgeschriebenes CSS für Styling wie WhatsApp
- Bug: -
- Feature: Clickable Examples als Buttons
- Feature: LLM Version soll noch von Link auf Dropdown umgestellt werden
- Feature: Clear Button für Chat und PDF

Team 4 - Algorithmus Alchemisten

Algorithms-Alchemisten
Gerrit - FE
Friedrich - FE
Leon
Julian - PDF2Text
Raoul - BE
Kjell

- Vollfunktionsfähige Demo gemäß MS 1
- Gradio-basierte Demo
- FITZ als PDF Viewer Library
- Bug: Side-by-Side View
- Bug: Nutze anders Standardbild bevor ein Dokument geladen wird
- Feature: Drag-and-Drop PDF
- Feature: Clickable Examples als Buttons
- Feature: LLM Version soll noch von Link auf Dropdown umgestellt werden
- Feature: Clear Button für Chat und PDF

Team 5 - Sunrise

Team Sunrise
Andrea
Anusha
Ivo
Fabian
Ahmad

- Vollfunktionsfähige Demo gemäß MS 1
- Flask-basierte Demo
- Drag&Drop funktioniert
- Try-outs: Login feature mit vergangenen Chats auf Basis eines JSOn Backends
- Bug: PDF Viewer noch nicht implementiert
- Bug: landing page und main page gleiches design
- Feature: Clickable Examples als Buttons
- Feature: LLM Version soll noch von Link auf Dropdown umgestellt werden
- Feature: Clear Button für Chat und PDF

Team 6 - Die Drei Musketiere

Die Drei Musketiere
Mika
Walid
Jonathan

- Fokus der Demo basiert auf Anwendungsprozess und deswegen nicht MS1
- Gradio-basierte Demo
- Full functioning PDF Viewer
- Bug: No Chat-Window yet (chronologische Entwicklungsreihenfolge)
- Feature: Clickable Examples als Buttons
- Feature: LLM Version soll noch von Link auf Dropdown umgestellt werden
- Feature: Clear Button für Chat und PDF
- Feature: ? User-basiertes Anzeigen nur der Nutzereigenen PDFs

Warum laufen Demos nicht auf den eigenen Systemen

- Windows/ Mac Installation von Flask (‘no module name found’) macht Probleme
- Port besetzt (5000)

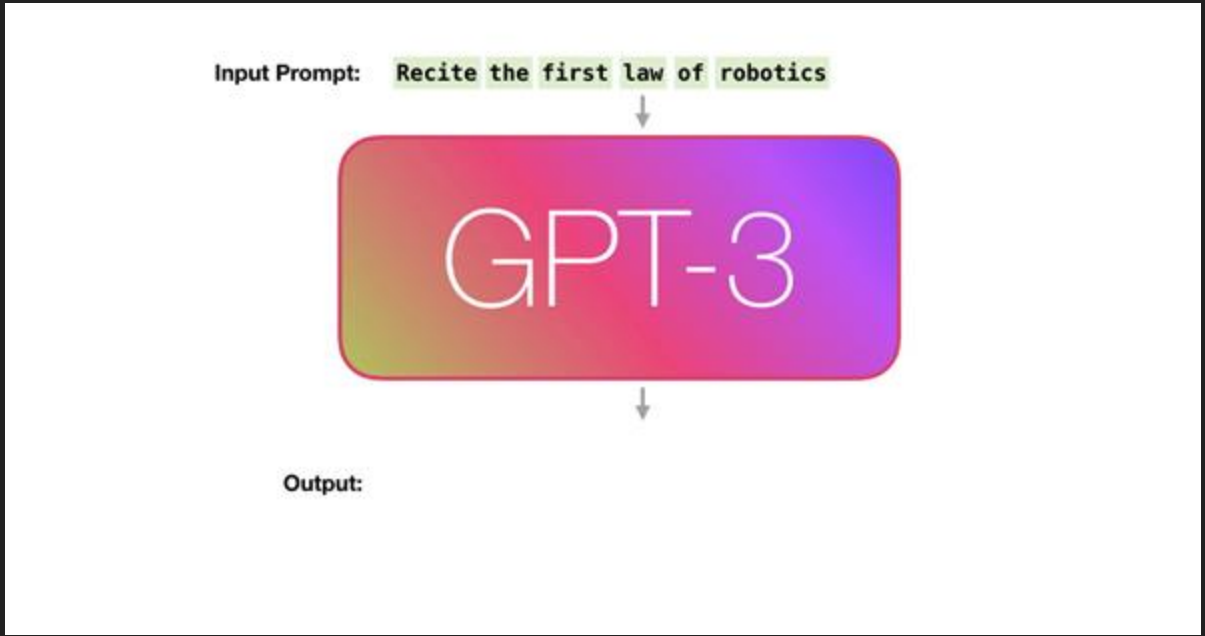
Milestones

Milestone	Features	Description	Dates
1	Basic web demo	Consists of a working frontend and backend, where information exchange takes place (POST request). Code must be on gitlab/github, with install instructions in a readme file. Every team member must be able to show a running demo on their machine.	6th May 2024 ✅
2	Backend talks to LLM	Modify your frontend to accept an input text. Send it to backend. The backend should send it to an LLM with an instruction, and send the result back to the frontend.	16th May 2024
3	Frontend accepts file upload	Modify frontend to allow a PDF upload, send it to backend. Convert PDF to text in backend and send it to frontend and display the text.	27th May 2024
4	Chat interface	Now your frontend should do the following: 1. Accept PDF upload 2. Present a chat interface 3. Working chat (not necessarily with LLM or PDF)	
5	PDF Chat + JSON Download	Allow the chat to take place with an LLM where the user can ask questions about the content of the PDF. Use an LLM to produce a downloadable JSON version of the chat. Well-written README.	

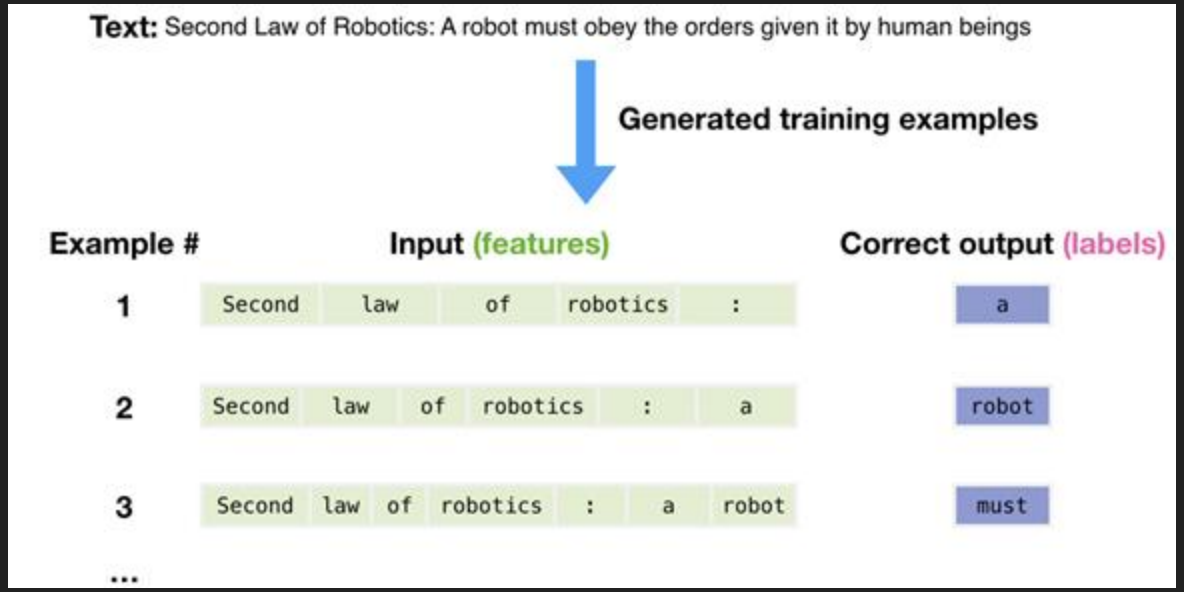
Week 7

What is an LLM

What is an LLM



What is an LLM



LLM First Steps: Your (Possible) Local Setup

- Autogen (https://github.com/microsoft/autogen) for multi-agent setup (or Langroid)
- Ollama (https://ollama.com/blog/openai-compatibility) for local LLMs
- Autogen + Ollama (https://ollama.com/blog/openai-compatibility) scroll down to Autogen section
- With Gemma:2b model, works on 4 core CPU and 8 GB RAM
- Try different models with ollama, which fit on your computer. Use Autogen over ollama for handling multi-agent communication.
- Try the Group Chat features https://microsoft.github.io/autogen/docs/tutorial/conversation-patterns
- Sample notebooks https://microsoft.github.io/autogen/docs/notebooks/

AIX LLM API

- You can use a local LLM (Mistral) hosted by us on a server via an API.
- You can access via code or on the command line using curl.
- You can make unlimited calls for free.

```
curl -X POST -u llm-chat-user:Nr17TT5m8Qj1oHS1   https://llm-
chat.skynet.coypu.org/generate_text
-H 'Content-Type: application/json'
-d '{
        "messages": [{"role": "user", "content": "What is your favourite condiment?"},
                            {"role": "assistant", "content": "It is lemon juice."},
                {"role": "user", "content": "Do you have mayonnaise recipes?"}]
 }'
```

curl -X POST -u llm-chat-user:Nr17TT5m8Qj1oHS1 https://llm-chat.skynet.coypu.org/generate_text -H 'Content-Type: application/json' -d'{ "messages": [{"role": "user", "content": "What is the fibonacci series?"}] }'

GDWG API

API key: bd5e3db58ba166a8edf4649aabd6a8f1

API-Endpunkt: https://chat-ai.academiccloud.de/v1

Verfügbare Modelle:

- intel-neural-chat-7b

- mixtral-8x7b-instruct

- qwen1.5-72b-chat

- meta-llama-3-70b-instruct

Bitte beachten Sie, dass sich die Modellliste in der Zukunft ändern kann. Sie werden über alle Änderungen informiert.

Unser Dienst ist OpenAI-kompatibel. Daher bieten wir, ähnlich wie OpenAI, zwei Haupt-APIs an:

- v1/completions für Textgenerierung und -vervollständigung

- v1/chat/completions für Benutzer-Assistenten-Gespräche

GDWG API

Um Ihren API-Zugang zu testen, können Sie diesen Befehl auf einer UNIX-Maschine ausführen:

text-completion API:

```
curl -i -N -X POST \
  --url https://chat-ai.academiccloud.de/v1/completions \
  --header 'Accept: application/json' \
  --header 'Authorization: Bearer <key>' \
  --header 'Content-Type: application/json'\
  --data '{
  "model": "intel-neural-chat-7b",
  "prompt": "San Fransico is a",
  "max_tokens": 7,
  "temperature": 0
}'
```

- chat-completion API:
- curl -i -N -X POST \
- --url https://chat-ai.academiccloud.de/v1/chat/completions \
- --header 'Accept: application/json' \
- --header 'Authorization: Bearer <key>' \
- --header 'Content-Type: application/json'\
- --data '{
- "model": "intel-neural-chat-7b",
- "messages": [{"role":"system","content":"You are a helpful assistant"},{"role":"user","content":"How tall is the Eiffel tower?"}],
- "temperature": 0
- }'

GDWG API

Bitte beachten Sie, <key> durch Ihren tatsächlichen API-Schlüssel zu ersetzen. Sobald Sie überprüft haben, dass der Schlüssel funktioniert, können Sie ihn in jeder OpenAI-kompatiblen Software oder jedem Tool verwenden, indem Sie einfach den API-Schlüssel und den Endpunkt einstellen. Hier ist ein Beispiel für einen Python-Client, der das openai-Paket verwendet:

```
from openai import OpenAI

# API configuration
api_key = '<api_key>' # Replace with your API key
base_url = "https://chat-ai.academiccloud.de/v1"
model = "intel-neural-chat-7b" # Choose any available model

# Start OpenAI client
client = OpenAI(
    api_key = api_key,
    base_url = base_url
)

# Get response
chat_completion = client.chat.completions.create(
        messages=[{"role":"system","content":"You are a helpful assistant"},{"role":"user","content":"How tall is the Eiffel tower?"}],
        model= model,
    )

# Print full response as JSON
print(chat_completion) # You can extract the response text from the JSON object
```

GPT 4 LLM API

- GPT 4 API access can also be provided selectively
- Please contact us for such API access
- Not everyone may receive this access because it is not free, or usage limits may be imposed

16.05.2024 Vector Databases and RAG

Vector Databases: https://www.youtube.com/watch?v=dN0lsF2cvm4

RAG: https://www.youtube.com/watch?v=T-D1OfcDW1M

16.05.2024 Milestone Achievement

Die Drei Musketiere

PDF upload works
Gradio based
Flask based
Hugging face basis
Problems faced
Works on localhost but not on cloud
Flask accepts pdf upload
Gradio is in an iframe and has chat
Unable to transfer pdf from flask to gradio
When gradio opens chat is possible.

Team Sunrise

Good looking UI
Upload of pdf works
Chat interface exists
Chat with LLM works (coypu API)
Preview of pdf does not work at the moment
Flask backend and web frontend html/javascript/css

Team North

Basic UI, chat window on right, pdf upload on left
Backend flask, but only empty functions

Team Pycharming

Flask backend Coypu LLM API
Frontend works, llm chat works Pdf upload works

PDF visualisation is not there yet

Team 1.0

Clean UI
Chat with llm works
Can upload, but doesn't visualise
Flask backend and html frontend

Week 8

23.05.2024

- Ensure no one is left behind
- Worked with teams and fixed issues
  - Python not working with pycharm, not finding flask
  - How to maintain separate config files on local and production environments
  - Git reset command for deleting bad commits
- In the next class, plan is to present some working RAG code

Team Alchemisten
Working web demo based on gradio
Pdf upload UI and chat UI exists
Chat with LLM works on console
Has been asked to change to the GWDG backend
PDF upload, conversion to text, and display of text works.
LLM chat does not work on UI console yet.

Presented by Leon, Julian.

Milestones

Milestone	Features	Description	Dates
1	Basic web demo	Consists of a working frontend and backend, where information exchange takes place (POST request). Code must be on gitlab/github, with install instructions in a readme file. Every team member must be able to show a running demo on their machine.	6th May 2024 ✅
2	Backend talks to LLM	Modify your frontend to accept an input text. Send it to backend. The backend should send it to an LLM with an instruction, and send the result back to the frontend.	16th May 2024 ✅
3	Frontend accepts file upload	Modify frontend to allow a PDF upload, send it to backend. Convert PDF to text in backend and send it to frontend and display the text.	27th May 2024
4	Chat interface	Now your frontend should do the following: 1. Accept PDF upload 2. Present a chat interface 3. Working chat (not necessarily with LLM or PDF)	
5	PDF Chat + JSON Download	Allow the chat to take place with an LLM where the user can ask questions about the content of the PDF. Use an LLM to produce a downloadable JSON version of the chat. Well-written README.	

Week 9

Milestones

Milestone	Features	Description	Dates
1	Basic web demo	Consists of a working frontend and backend, where information exchange takes place (POST request). Code must be on gitlab/github, with install instructions in a readme file. Every team member must be able to show a running demo on their machine.	6th May 2024 ✅
2	Backend talks to LLM	Modify your frontend to accept an input text. Send it to backend. The backend should send it to an LLM with an instruction, and send the result back to the frontend.	16th May 2024 ✅
3	Frontend accepts file upload	Modify frontend to allow a PDF upload, send it to backend. Convert PDF to text in backend and send it to frontend and display the text.	27th May 2024
4	Chat interface	Now your frontend should do the following: 1. Accept PDF upload 2. Present a chat interface 3. Working chat (not necessarily with LLM or PDF)	
5	PDF Chat + JSON Download	Allow the chat to take place with an LLM where the user can ask questions about the content of the PDF. Use an LLM to produce a downloadable JSON version of the chat. Well-written README.	

27.05.2024 Milestone Achievements

Die Drei Musketiere

Works on localhost but not on cloud => moved to one page GRADIO
PDF upload works
Gradio based
Flask based
~~Hugging face basis~~
Flask accepts pdf upload

Team Sunrise

Good looking UI
Upload of pdf works, and to text works
Chat interface exists
Chat with LLM works (intel GDWG API)
Preview works
Flask backend and web frontend html/javascript/css
Typewriter in Chat
Bug: Chat History + PDF are not given to LLM

Team Pycharming

Flask backend Coypu LLM API
Frontend works, llm chat works Pdf upload works
PDF visualisation is not there yet

Team Alchemisten

Working web demo based on gradio
Pdf upload UI and chat UI exists
Has been asked to change to the GWDG backend
PDF upload, conversion to text, and display of text works.
Bug: Chat with LLM works on console, LLM chat does not work on UI console yet
Bug: Save data in a session cookie, e.g. PDF text, is not possible

Presented by Leon, Julian.

Team North

Basic UI, chat window on right, pdf upload on left
Backend flask, but only empty functions

Team 1.0

Clean UI
Chat with LLM works
Can upload, but doesn't visualise in text box
Flask backend and html frontend

Bug (?): llama3 funktioniert nicht

Milestones

Milestone	Features	Description	Dates
1	Basic web demo	Consists of a working frontend and backend, where information exchange takes place (POST request). Code must be on gitlab/github, with install instructions in a readme file. Every team member must be able to show a running demo on their machine.	6th May 2024 ✅
2	Backend talks to LLM	Modify your frontend to accept an input text. Send it to backend. The backend should send it to an LLM with an instruction, and send the result back to the frontend.	16th May 2024 ✅
3	Frontend accepts file upload	Modify frontend to allow a PDF upload, send it to backend. Convert PDF to text in backend and send it to frontend and display the text.	27th May 2024 ✅
4	Chat interface	Now your frontend should do the following: 1. Accept PDF upload 2. Present a chat interface 3. Working chat **with the PDF**	10th June 2024
5	PDF Chat + JSON Download	Allow the chat to take place with an LLM where the user can ask questions about the content of the PDF. Use an LLM to produce a downloadable JSON version (see **specification**). Well-written README. **Eval numbers.**	24th June 2024

Specification of the "Download Button"

- Use an LLM to extract 10 key values from given PDF via one click and download the result into a JSON TODO: Key Definition
- Quantitative Features: CO2 in t/annum, NOX in t/annum, Number of Electric Vehicles
- Qualitative Features:
  - Impact: Negative impact on climate change from a company's activities that the company addresses in the report.
  - Risks: Material risks from impact on the climate change.
  - Opportunities: Financial materiality from company's activities related to climate change.
  - Strategy: Company's strategy and business model in line with the transition to a sustainable economy.
  - Actions: Actions and resources in relation to material sustainability matters.
  - Adopted policies: Policies adopted to manage material sustainability matters
  - Targets: Company's goals towards a sustainable economy.

Specification of the "Download Button" - Key-Value

```
{
"name" : "NAME_OF_THE_DOC",
"CO2" : "…",
"NOX" : "…",
"Number_of_Electric_Vehicles" : "…",
"Impact" : "…",
"Risks" : "…",
"Opportunities" : "…",
"Strategy" : "…",
"Actions" : "…",
"Adopted_policies" : "…",
"Targets" : "…"
}
```

Grading Scale

- Software Project with Presentation (30%)
  - Structure, Unified Design and Readability of slides and its figures (Story, Long Sentences,...)
  - Performance of the presenter (Freely, Clearly, Examples, Excitement, Formality, Embedded,...)
  - Correctness of the presentation and answers to questions
- Project Documentation (70%)
  - How good does the code work? (Milestone requirements)
  - Evaluation numbers over the PDF's Download button, ReadMe completeness, code style (comments), deployability
- Attention!
  - Presentation of the project in the last lecture week
  - Hand in of everything in the last lecture week (link to the repo incl. README, Download results and presentation as an email to both lecturers)