MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

NATIONAL TECHNICAL UNIVERSITY OF UKRAINE

" IHORY SIKORSKY KYIV POLYTECHNIC INSTITUTE"

Kostiantyn Radchenko

Modern Software Development Technologies

Laboratory Work 3

AI-Powered Software Development

kulubecioglu mehmet

IM-14 FIOT

Github: https://github.com/mklbc/AI_project

Kyiv

IHORY SIKORSKY KYIV POLYTECHNIC INSTITUTE

2024

1. Introduction This report presents the implementation and deployment of an AI-powered software development project. The objective of this laboratory work is to develop an AI-based application using FastAPI, integrate machine learning for sentiment analysis, and deploy it using GitHub for version control. The report also provides answers to the questions included in the laboratory assignment.

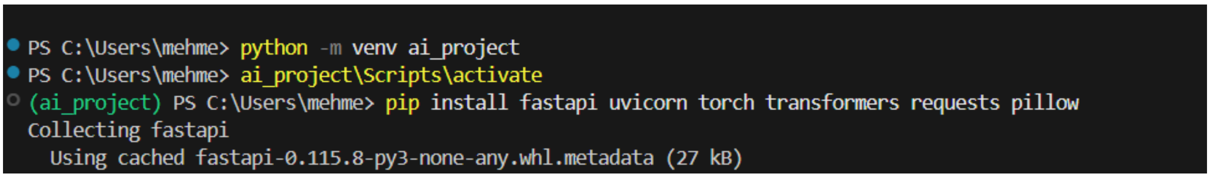
2. Implementation

Technologies Used:

- **FastAPI**: A modern web framework for building APIs with Python.
- **Uvicorn**: An ASGI server to run the FastAPI application.
- **Transformers & PyTorch**: For implementing an AI-powered sentiment analysis model.
- **GitHub**: For version control and collaboration.

Development Steps:

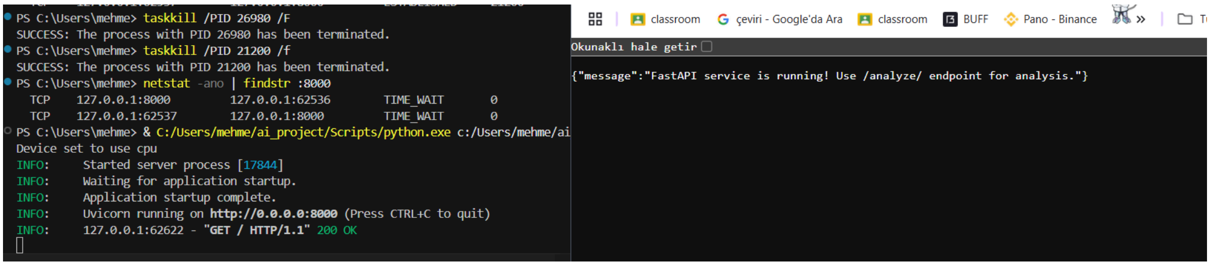
1. **Setting Up the FastAPI Application**
    - Installed FastAPI and Uvicorn using:



```
PS C:\Users\mehme> python -m venv ai_project
PS C:\Users\mehme> ai_project\Scripts\activate
(ai_project) PS C:\Users\mehme> pip install fastapi uvicorn torch transformers requests pillow
Collecting fastapi
  Using cached fastapi-0.115.8-py3-none-any.whl.metadata (27 kB)
```

    - Created `ai_service.py`, which initializes and runs the API.
2. **Implementing AI Functionalities**
    - Integrated a text classification model using the Transformers library.
    - Used `distilbert-base-uncased-finetuned-sst-2-english` for sentiment analysis.
3. **Testing Locally**
    - Ran the application using:

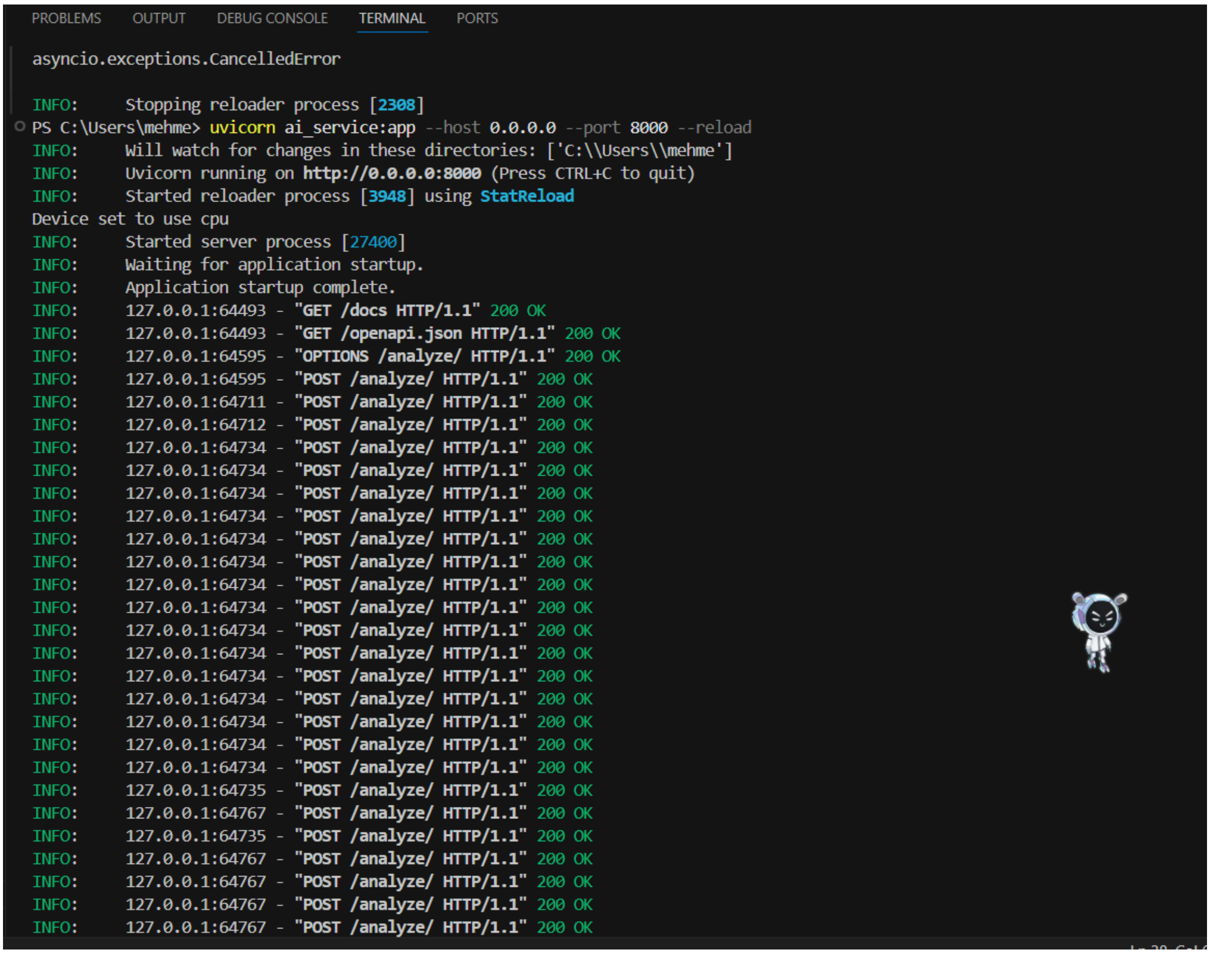
```
uvicorn ai_service:app --host 0.0.0.0 --port 8000
```



```
PS C:\Users\mehme> taskkill /PID 26980 /F
SUCCESS: The process with PID 26980 has been terminated.
PS C:\Users\mehme> taskkill /PID 21200 /f
SUCCESS: The process with PID 21200 has been terminated.
PS C:\Users\mehme> netstat -ano | findstr :8000
  TCP    127.0.0.1:8000         127.0.0.1:62536        TIME_WAIT       0
  TCP    127.0.0.1:62537        127.0.0.1:8000         TIME_WAIT       0
PS C:\Users\mehme> & C:/Users/mehme/ai_project/Scripts/python.exe c:/Users/mehme/ai
Device set to use cpu
INFO:     Started server process [17844]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO:     127.0.0.1:62622 - "GET / HTTP/1.1" 200 OK
```

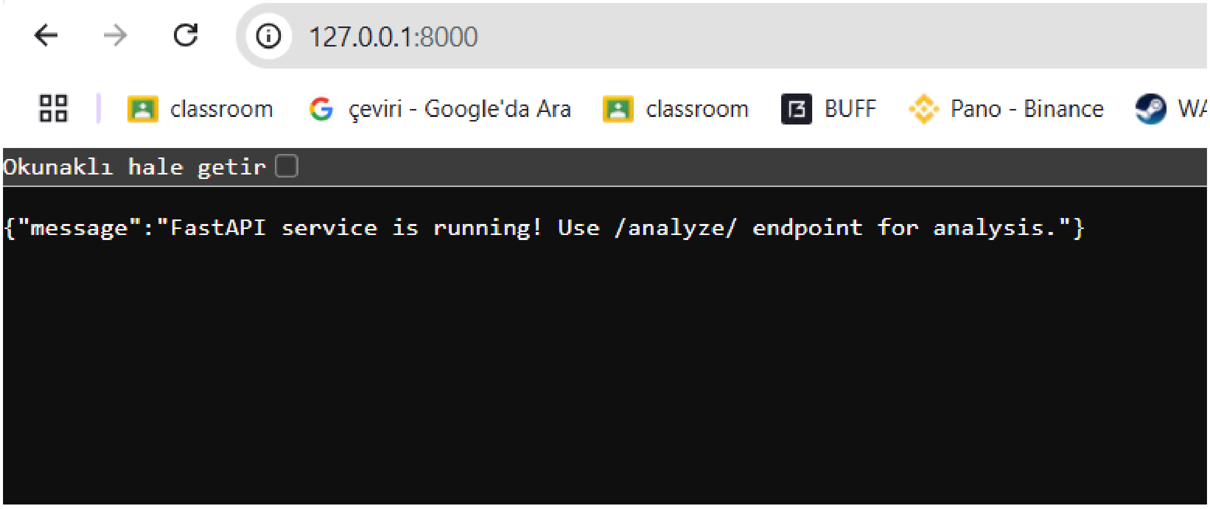
Okunaklı hale getir ☐

```
{"message":"FastAPI service is running! Use /analyze/ endpoint for analysis."}
```



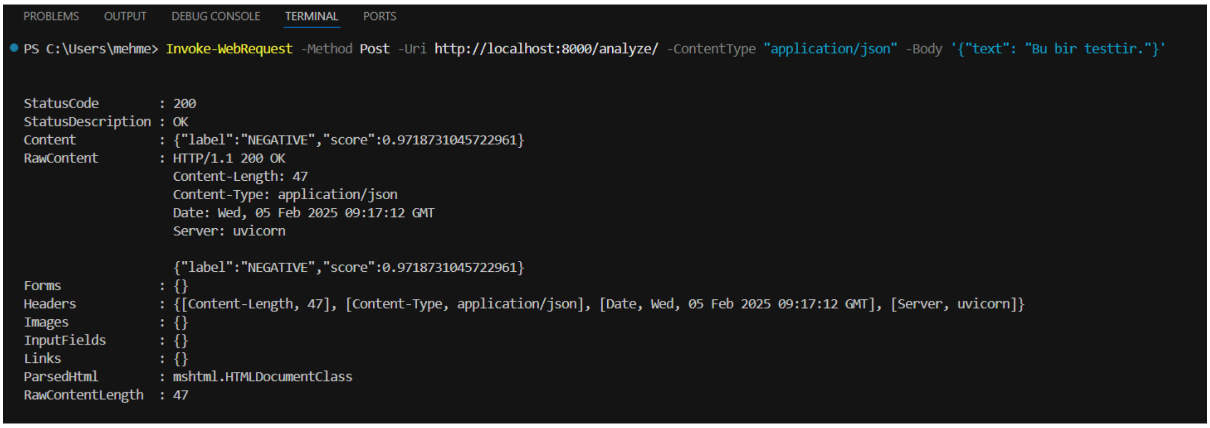
```
asyncio.exceptions.CancelledError

INFO:     Stopping reloader process [2308]
PS C:\Users\mehme> uvicorn ai_service:app --host 0.0.0.0 --port 8000 --reload
INFO:     Will watch for changes in these directories: ['C:\\Users\\mehme']
INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO:     Started reloader process [3948] using StatReload
Device set to use cpu
INFO:     Started server process [27400]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     127.0.0.1:64493 - "GET /docs HTTP/1.1" 200 OK
INFO:     127.0.0.1:64493 - "GET /openapi.json HTTP/1.1" 200 OK
INFO:     127.0.0.1:64595 - "OPTIONS /analyze/ HTTP/1.1" 200 OK
INFO:     127.0.0.1:64595 - "POST /analyze/ HTTP/1.1" 200 OK
INFO:     127.0.0.1:64711 - "POST /analyze/ HTTP/1.1" 200 OK
INFO:     127.0.0.1:64712 - "POST /analyze/ HTTP/1.1" 200 OK
INFO:     127.0.0.1:64734 - "POST /analyze/ HTTP/1.1" 200 OK
INFO:     127.0.0.1:64734 - "POST /analyze/ HTTP/1.1" 200 OK
INFO:     127.0.0.1:64734 - "POST /analyze/ HTTP/1.1" 200 OK
INFO:     127.0.0.1:64734 - "POST /analyze/ HTTP/1.1" 200 OK
INFO:     127.0.0.1:64734 - "POST /analyze/ HTTP/1.1" 200 OK
INFO:     127.0.0.1:64734 - "POST /analyze/ HTTP/1.1" 200 OK
INFO:     127.0.0.1:64734 - "POST /analyze/ HTTP/1.1" 200 OK
INFO:     127.0.0.1:64734 - "POST /analyze/ HTTP/1.1" 200 OK
INFO:     127.0.0.1:64734 - "POST /analyze/ HTTP/1.1" 200 OK
INFO:     127.0.0.1:64734 - "POST /analyze/ HTTP/1.1" 200 OK
INFO:     127.0.0.1:64734 - "POST /analyze/ HTTP/1.1" 200 OK
INFO:     127.0.0.1:64734 - "POST /analyze/ HTTP/1.1" 200 OK
INFO:     127.0.0.1:64734 - "POST /analyze/ HTTP/1.1" 200 OK
INFO:     127.0.0.1:64734 - "POST /analyze/ HTTP/1.1" 200 OK
INFO:     127.0.0.1:64734 - "POST /analyze/ HTTP/1.1" 200 OK
INFO:     127.0.0.1:64735 - "POST /analyze/ HTTP/1.1" 200 OK
INFO:     127.0.0.1:64767 - "POST /analyze/ HTTP/1.1" 200 OK
INFO:     127.0.0.1:64735 - "POST /analyze/ HTTP/1.1" 200 OK
INFO:     127.0.0.1:64767 - "POST /analyze/ HTTP/1.1" 200 OK
INFO:     127.0.0.1:64767 - "POST /analyze/ HTTP/1.1" 200 OK
INFO:     127.0.0.1:64767 - "POST /analyze/ HTTP/1.1" 200 OK
INFO:     127.0.0.1:64767 - "POST /analyze/ HTTP/1.1" 200 OK
```

1. **Uploading to GitHub**
    - Used `git` commands to push the project repository.
2. **Deployment Considerations**
    - Configured `requirements.txt` for dependency management.

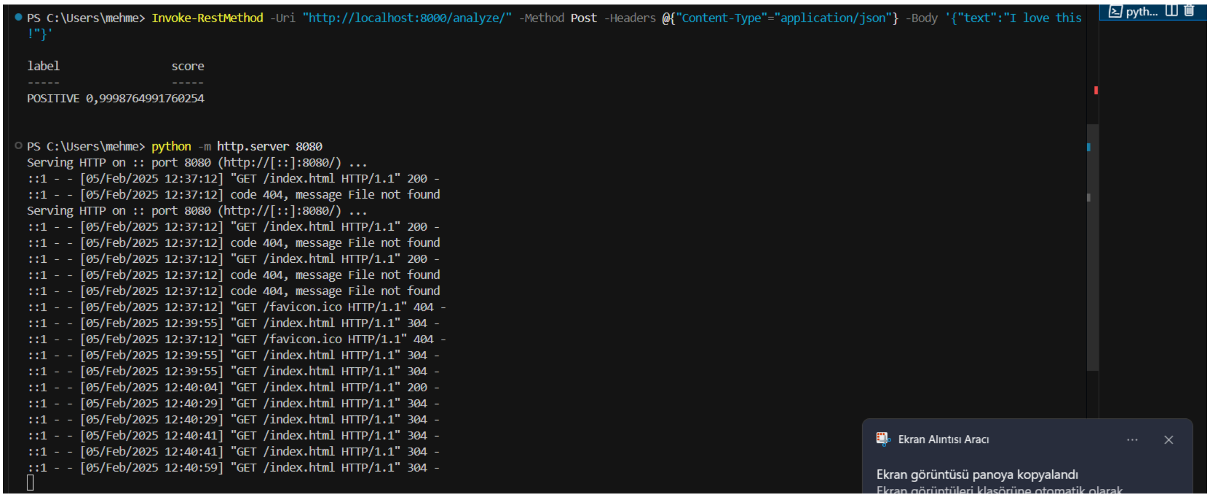


Okunaklı hale getir ☐

```
{"message":"FastAPI service is running! Use /analyze/ endpoint for analysis."}
```



```
● PS C:\Users\mehme> Invoke-WebRequest -Method Post -Uri http://localhost:8000/analyze/ -ContentType "application/json" -Body '{"text": "Bu bir testtir."}'


StatusCode        : 200
StatusDescription : OK
Content           : {"label":"NEGATIVE","score":0.9718731045722961}
RawContent        : HTTP/1.1 200 OK
                    Content-Length: 47
                    Content-Type: application/json
                    Date: Wed, 05 Feb 2025 09:17:12 GMT
                    Server: uvicorn

                    {"label":"NEGATIVE","score":0.9718731045722961}
Forms             : {}
Headers           : {[Content-Length, 47], [Content-Type, application/json], [Date, Wed, 05 Feb 2025 09:17:12 GMT], [Server, uvicorn]}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 47
```

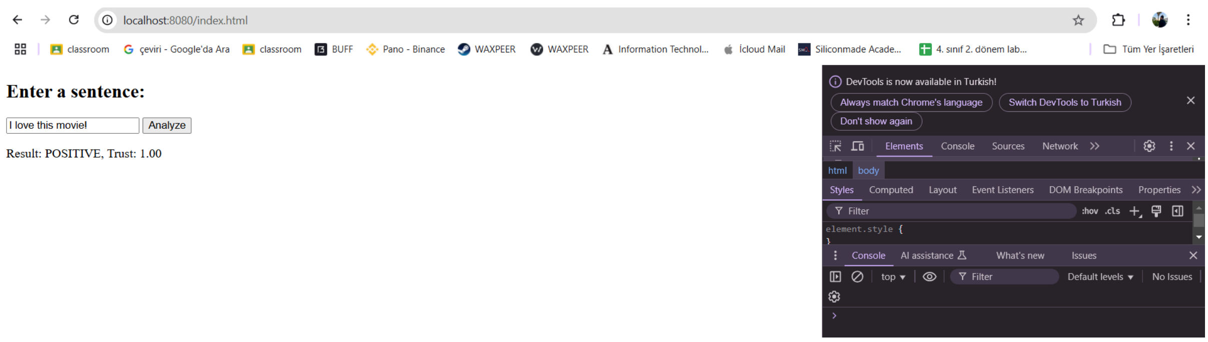


```
● PS C:\Users\mehme> Invoke-RestMethod -Uri "http://localhost:8000/analyze/" -Method Post -Headers @{"Content-Type"="application/json"} -Body '{"text":"I love this
  !"}'

  label                score
  -----                -----
  POSITIVE 0,9998764991760254


○ PS C:\Users\mehme> python -m http.server 8080
  Serving HTTP on :: port 8080 (http://[::]:8080/) ...
  ::1 - - [05/Feb/2025 12:37:12] "GET /index.html HTTP/1.1" 200 -
  ::1 - - [05/Feb/2025 12:37:12] code 404, message File not found
  Serving HTTP on :: port 8080 (http://[::]:8080/) ...
  ::1 - - [05/Feb/2025 12:37:12] "GET /index.html HTTP/1.1" 200 -
  ::1 - - [05/Feb/2025 12:37:12] code 404, message File not found
  ::1 - - [05/Feb/2025 12:37:12] "GET /index.html HTTP/1.1" 200 -
  ::1 - - [05/Feb/2025 12:37:12] code 404, message File not found
  ::1 - - [05/Feb/2025 12:37:12] code 404, message File not found
  ::1 - - [05/Feb/2025 12:37:12] "GET /favicon.ico HTTP/1.1" 404 -
  ::1 - - [05/Feb/2025 12:39:55] "GET /index.html HTTP/1.1" 304 -
  ::1 - - [05/Feb/2025 12:37:12] "GET /favicon.ico HTTP/1.1" 404 -
  ::1 - - [05/Feb/2025 12:39:55] "GET /index.html HTTP/1.1" 304 -
  ::1 - - [05/Feb/2025 12:39:55] "GET /index.html HTTP/1.1" 304 -
  ::1 - - [05/Feb/2025 12:40:04] "GET /index.html HTTP/1.1" 200 -
  ::1 - - [05/Feb/2025 12:40:29] "GET /index.html HTTP/1.1" 304 -
  ::1 - - [05/Feb/2025 12:40:29] "GET /index.html HTTP/1.1" 304 -
  ::1 - - [05/Feb/2025 12:40:41] "GET /index.html HTTP/1.1" 304 -
  ::1 - - [05/Feb/2025 12:40:41] "GET /index.html HTTP/1.1" 304 -
  ::1 - - [05/Feb/2025 12:40:59] "GET /index.html HTTP/1.1" 304 -
```

Ekran Alıntısı Aracı

Ekran görüntüsü panoya kopyalandı

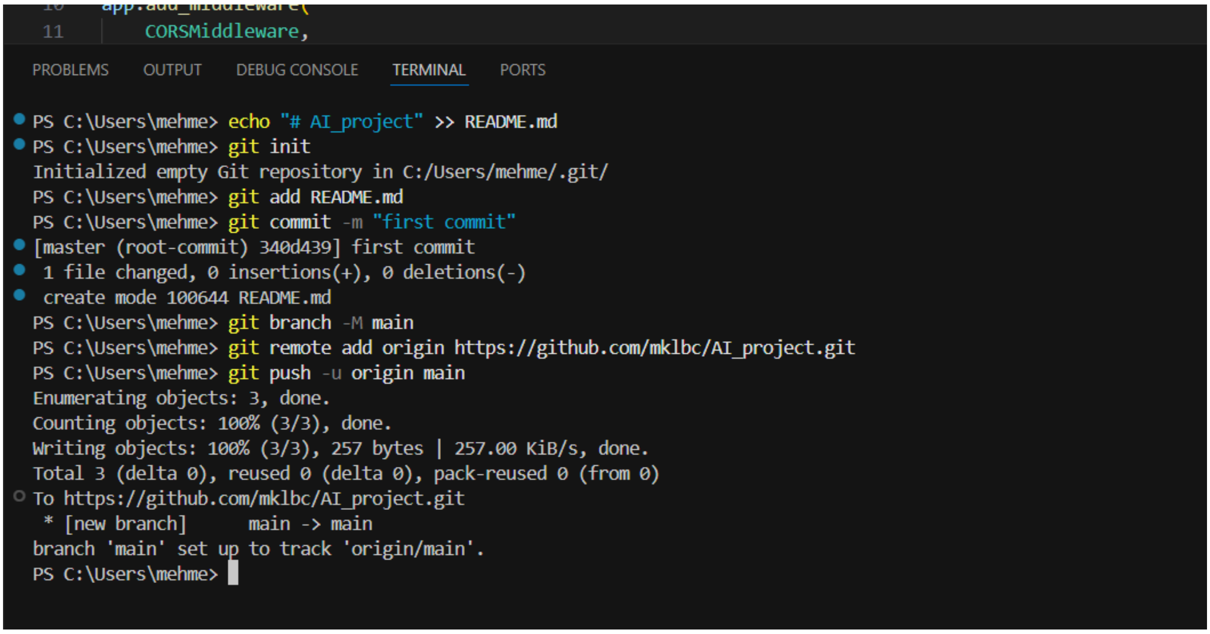
Ekran görüntüleri klasörüne otomatik olarak



Enter a sentence:

I love this movie! Analyze

Result: POSITIVE, Trust: 1.00



```
    app.add_middleware(
        CORSMiddleware,
```

```
PS C:\Users\mehme> echo "# AI_project" >> README.md
PS C:\Users\mehme> git init
Initialized empty Git repository in C:/Users/mehme/.git/
PS C:\Users\mehme> git add README.md
PS C:\Users\mehme> git commit -m "first commit"
[master (root-commit) 340d439] first commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 README.md
PS C:\Users\mehme> git branch -M main
PS C:\Users\mehme> git remote add origin https://github.com/mklbc/AI_project.git
PS C:\Users\mehme> git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 257 bytes | 257.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/mklbc/AI_project.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
PS C:\Users\mehme> 
```

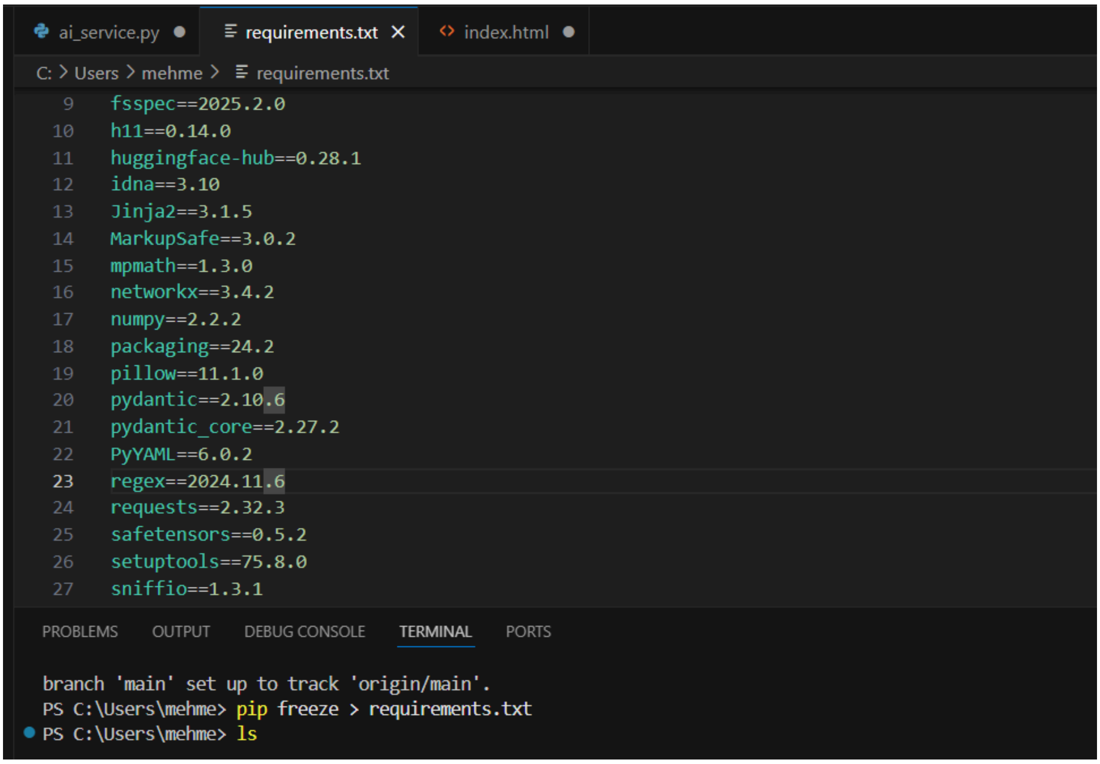
Enter a sentence:

impossible Analyze

Result: NEGATIVE, Trust: 0.99

Project Files:

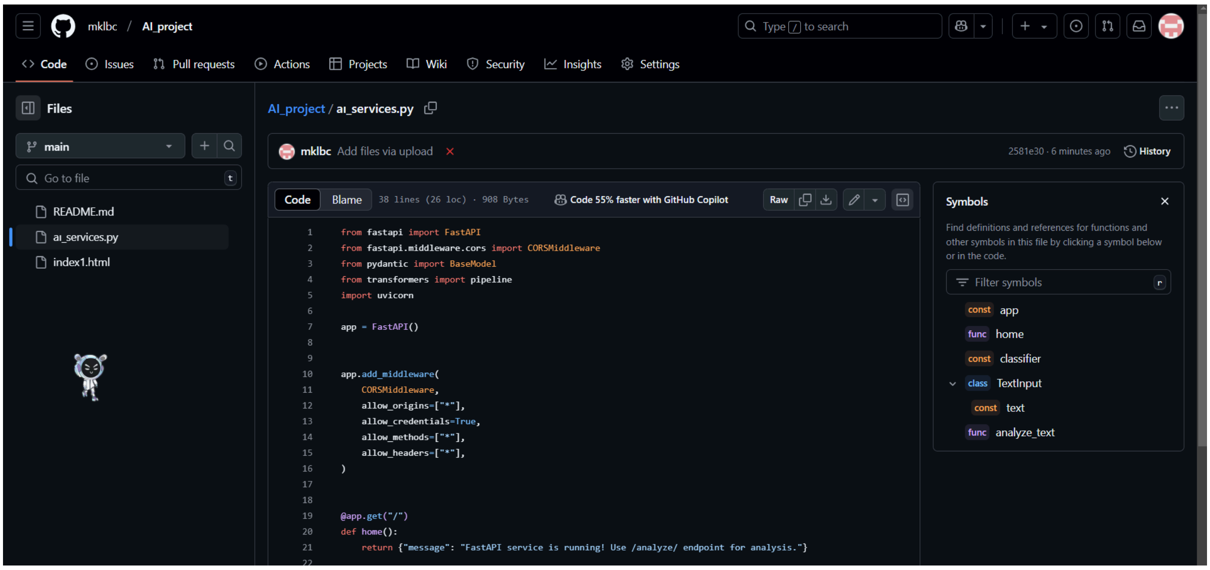
requirements.txt



```
fsspec==2025.2.0
h11==0.14.0
huggingface-hub==0.28.1
idna==3.10
Jinja2==3.1.5
MarkupSafe==3.0.2
mpmath==1.3.0
networkx==3.4.2
numpy==2.2.2
packaging==24.2
pillow==11.1.0
pydantic==2.10.6
pydantic_core==2.27.2
PyYAML==6.0.2
regex==2024.11.6
requests==2.32.3
safetensors==0.5.2
setuptools==75.8.0
sniffio==1.3.1
```

```
branch 'main' set up to track 'origin/main'.
PS C:\Users\mehme> pip freeze > requirements.txt
PS C:\Users\mehme> ls
```

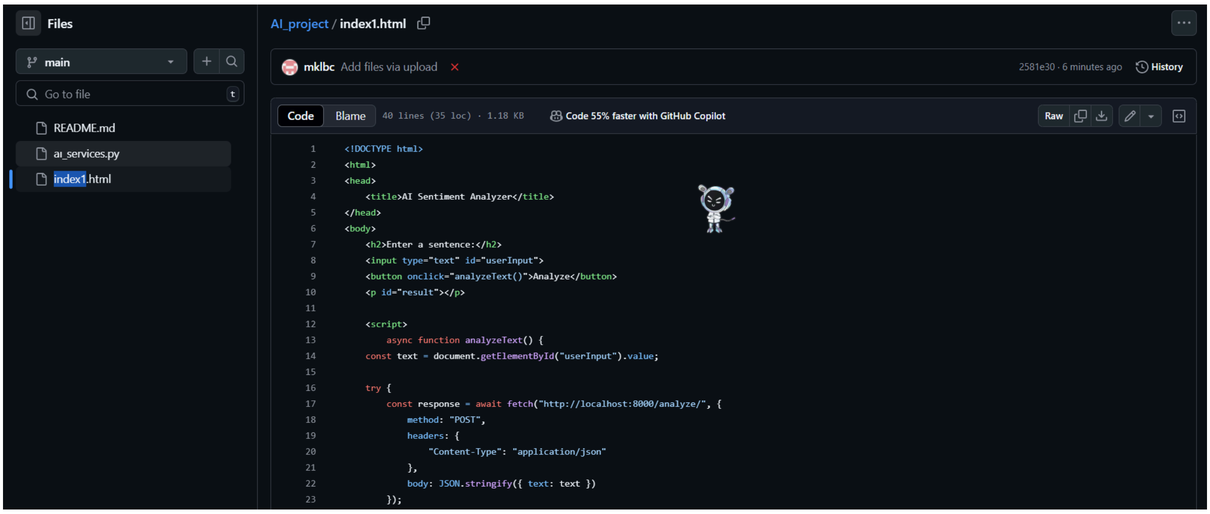
ai_service.py



```python
from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware
from pydantic import BaseModel
from transformers import pipeline
import uvicorn

app = FastAPI()


app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)


@app.get("/")
def home():
    return {"message": "FastAPI service is running! Use /analyze/ endpoint for analysis."}
```

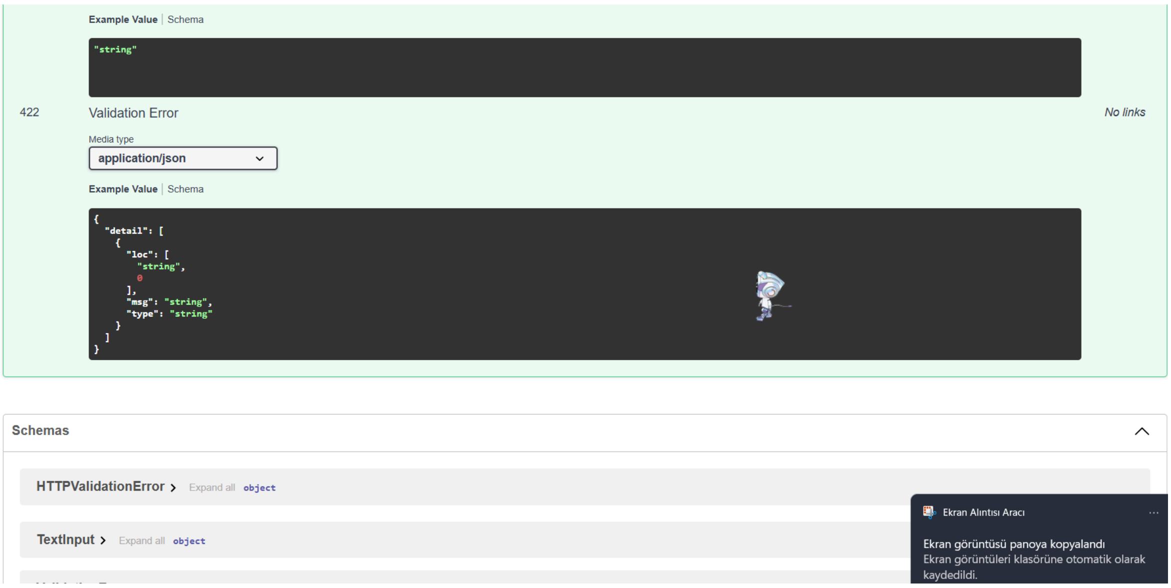
```python
@app.get("/")
def home():
    return {"message": "FastAPI service is running! Use /analyze/ endpoint for analysis."}


classifier = pipeline("text-classification", model="distilbert-base-uncased-finetuned-sst-2-english")


class TextInput(BaseModel):
    text: str

# Analiz endpoint'i
@app.post("/analyze/")
def analyze_text(input: TextInput):
    result = classifier(input.text)
    return {"label": result[0]['label'], "score": result[0]['score']}


if __name__ == "__main__":
    uvicorn.run(app, host="0.0.0.0", port=8000, reload=True)
```

index.html



```html
<!DOCTYPE html>
<html>
<head>
    <title>AI Sentiment Analyzer</title>
</head>
<body>
    <h2>Enter a sentence:</h2>
    <input type="text" id="userInput">
    <button onclick="analyzeText()">Analyze</button>
    <p id="result"></p>

    <script>
        async function analyzeText() {
    const text = document.getElementById("userInput").value;

    try {
        const response = await fetch("http://localhost:8000/analyze/", {
            method: "POST",
            headers: {
                "Content-Type": "application/json"
            },
            body: JSON.stringify({ text: text })
        });
```

```
        if (!response.ok) {
            const errorData = await response.json();
            throw new Error(`HTTP error! status: ${response.status}, detail: ${JSON.stringify(errorData)}`);
        }

        const data = await response.json();
        document.getElementById("result").innerText = `Result: ${data.label}, Trust: ${data.score.toFixed(2)}`;
    } catch (error) {
        console.error("Error:", error);
        document.getElementById("result").innerText = "An error has occurred. Check console.";
    }
}

    </script>
</body>
</html>
```

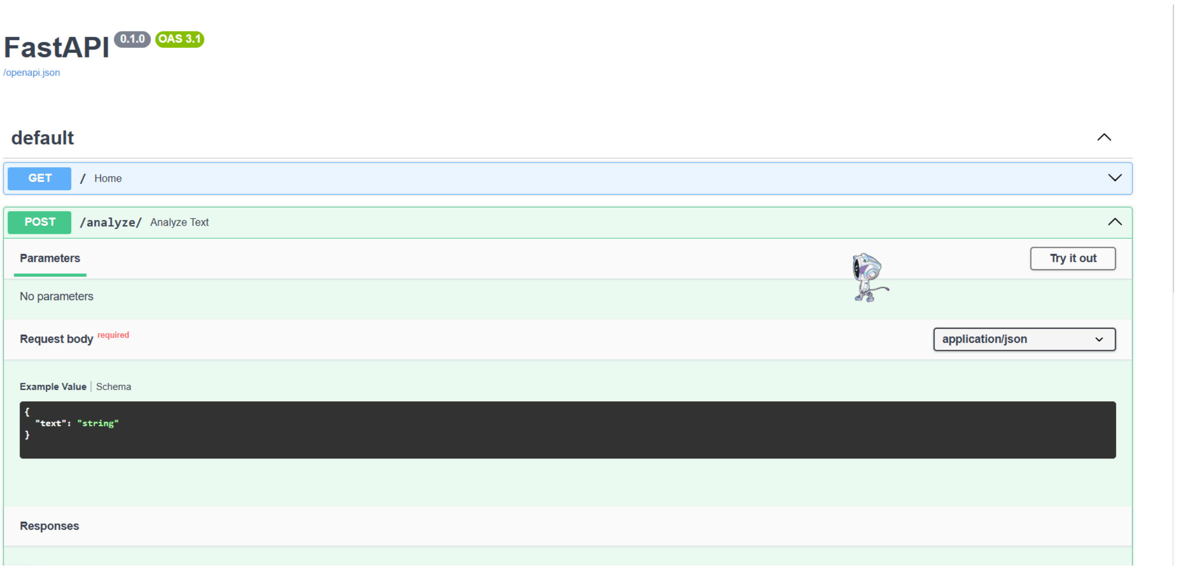


```
"string"
```

422 Validation Error

```
{
  "detail": [
    {
      "loc": [
        "string",
        0
      ],
      "msg": "string",
      "type": "string"
    }
  ]
}
```

Ekran görüntüsü panoya kopyalandı
Ekran görüntüleri klasörüne otomatik olarak kaydedildi.



No parameters

```
{
  "text": "string"
}
```

3. Answers to Questions

1. AI enhances automation, improves accuracy, and enables better decision-making.
2. AI models learn from data and generalize patterns, whereas traditional algorithms follow predefined rules.
3. Common AI-powered features include recommendation systems, chatbots, and speech recognition.
4. Pre-trained models reduce development time and computational costs.
5. Transfer learning allows AI models to adapt learned features from one task to another.
6. REST APIs enable seamless integration of AI functionalities into various applications.
7. Synchronous inference processes requests sequentially, whereas asynchronous inference handles multiple requests simultaneously.
8. Performance challenges include model latency, scalability, and resource-intensive computations.
9. FastAPI provides better performance and async support compared to Flask for AI applications.
10. GPU acceleration speeds up AI inference by leveraging parallel processing.
11. Docker ensures environment consistency and simplifies deployment.
12. Model quantization reduces model size and speeds up inference.
13. Security risks include unauthorized access, data privacy issues, and model exploitation.
14. Ethical concerns include bias in AI models and data privacy issues.
15. AI applications can be monitored using logging, feedback loops, and continuous model improvements.

Conclusion This laboratory work provided valuable experience in developing AI-powered web applications, integrating machine learning models into APIs, and managing code with version control. Understanding dependency management and API deployment best practices were key takeaways from this project.