

1 Implementierung mit Streamlit

Eine weitere Implementierung des Haar Cascade Klassifizierers wurde mit dem Dashboard Framework **Streamlit** als **Web-Anwendung** umgesetzt. Das Dashboard enthält zum einen einen Bereich mit unterschiedlichen Parametern für den Haar Cascade Klassifizierer, und zum anderen einen Bereich, in dem verschieden Bild-Quellen ausgewählt werden können. Dazu gehört neben eigenen oder vorinstallierten Bildern auch die Möglichkeit, ein Video oder das eigene Webcam-Bild zu verwenden.

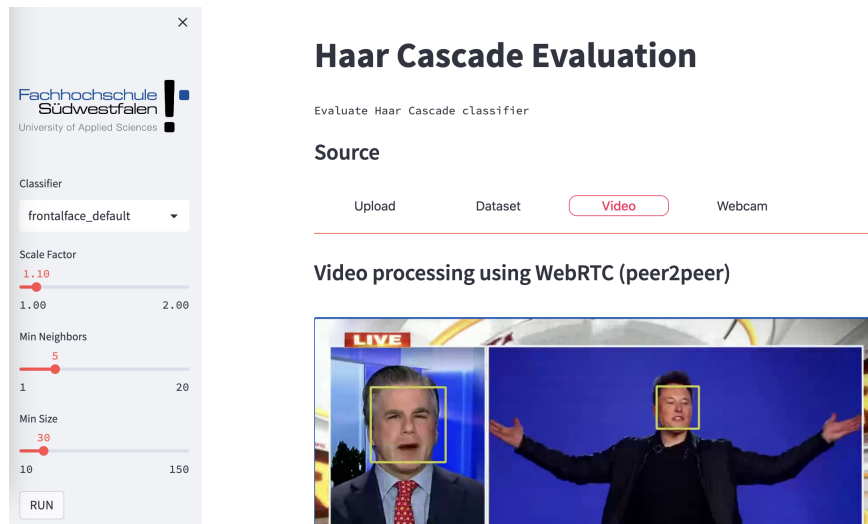


Abbildung 1: Streamlit Dashboard mit Haar Cascade Klassifizierer

Das Dashboard kann auch als Demo-Anwendung auf der Streamlit-Cloud ausprobiert werden.

1.1 Eingabequellen

Die Anwendung unterstützt diese bereits angesprochenen Eingabemöglichkeiten:

Upload

Hier kann ein einzelnes Bild hochgeladen und verwendet werden

Dataset

Unter diesem Punkt können mehrere vorkonfigurierte Bilder auf einmal verwendet werden

Video

Bei dieser Option wird ein vorkonfiguriertes Video abgespielt, auf dem der ausgewählte Klassifizierer angewandt wird

Webcam

Hier besteht die Möglichkeit, den Klassifizierer auf die eigene Webcam-Bilder anzuwenden

1.2 Optionen

Die verwendeten Optionen zur Konfiguration des Haar Cascade Klassifizierers **Scale Factor**, **Min Neighbors** und **Min Size** wurden bereits in den vorherigen Abschnitten im Rahmen der Tkinter Anwendung beschrieben.

1.3 Code

Der Programmcode für die Implementierung des Streamlit-Dashboards ist auf die folgenden Quelldateien verteilt:

```
1 # Hauptanwendung und Einstiegspunkt
2 app.py
3 # Funktion zur Ausfuehrung des eigentlichen Klassifizierers
4 pki_a22_app/haarcascades/haarcascades.py
5 # Interface fuer die unterschiedlichen Eingabequellen
6 pki_a22_app/dashboard/sources.py
7 # Eingabequelle mit einem einzelnen hochladbarem Bild
8 pki_a22_app/dashboard/source_upload.py
9 # Eingabequelle mit Auswahl eines Datensatzes mit mehreren
  Bildern
10 pki_a22_app/dashboard/source_dataset.py
11 # Eingabequelle zum Abspielen eines voreingestellten Videos
12 pki_a22_app/dashboard/source_video.py
13 # Eingabequelle bei der die eigene Webcam verwendet wird
14 pki_a22_app/dashboard/source_webcam.py
15 # Utility Modul, das vor allem Funktionen fuer Datei-Operationen
  enthaelt
16 pki_a22_app/utils/file_loader.py
```

2 Organisation

Besonders zu Beginn war es sehr hilfreich, eine Kollaborations-Plattform für das Projekt einzurichten, über die die unterschiedlichsten Informationen organisatorischer Natur ausgetauscht werden konnten. Wir haben uns für den Einsatz eines Miro-Boardes entschieden. das wir unter anderem nutzten für:

- Terminfindung
- Pinnwand
- Brainstorming
- Zeitplanung
- Wireframes

3 Continuos Integration

Um gemeinsam am Programmcode arbeiten zu können, haben wir uns ein Git-basiertes Codeverwaltungs-Tool entschieden. Hierfür haben wir auf GitHub ein eigenes Git-Repository eingerichtet.

Zur Qualitätssicherung haben wir uns dafür entschieden, für den Main-Branch Pull-Requests zu verwenden und die Erfolgreiche Ausführung eines GitHub-Actions- Workflows für Unit-Tests vorauszusetzen.

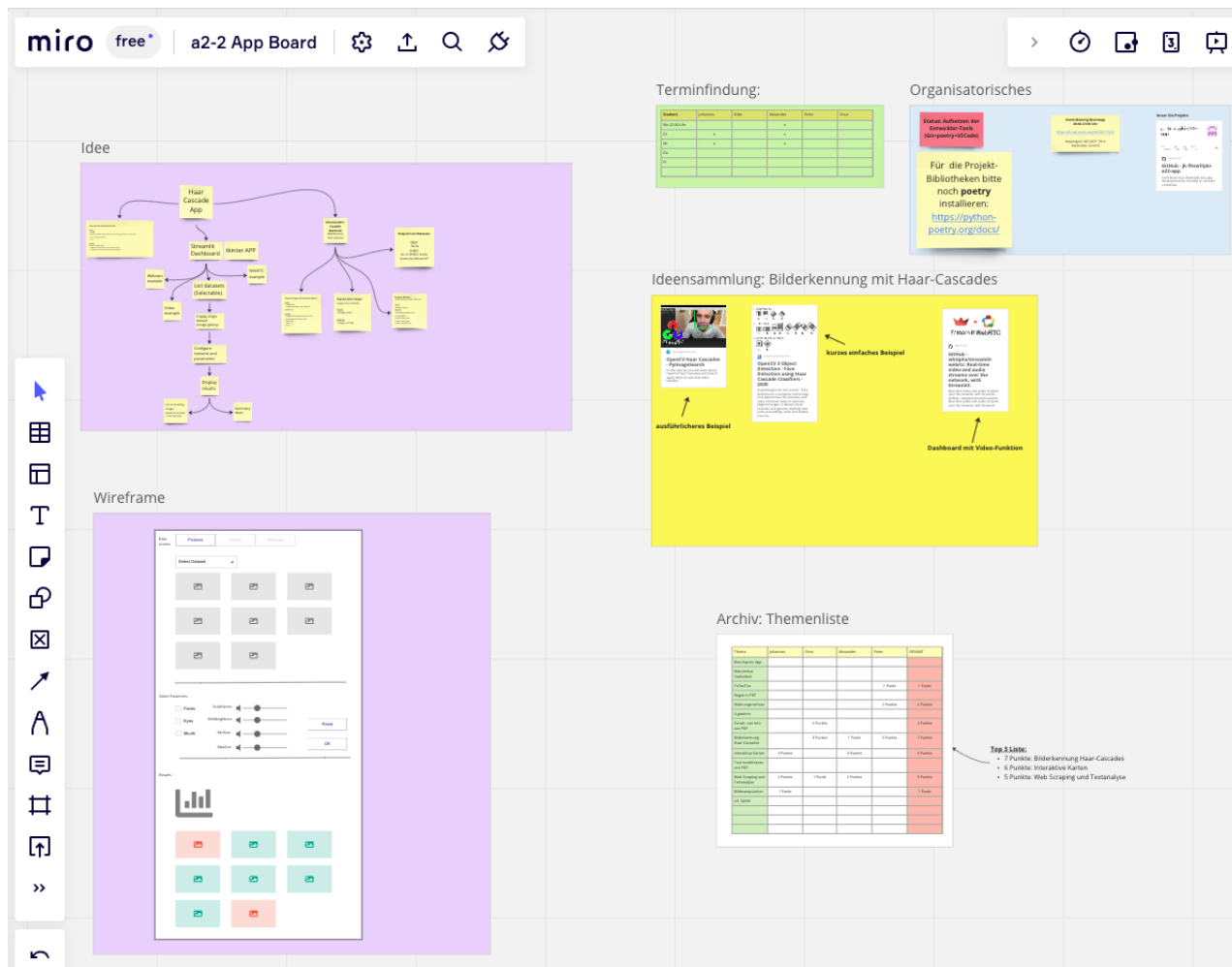



Abbildung 2: Miro-Board für organisatorische Aufgaben





[Pull requests](#)
[Issues](#)
[Codespaces](#)
[Marketplace](#)
[Explore](#)
















[jk-fhswf / pki-a22-app](#)
Public
Pin
Unwatch 2

[Code](#)
[Issues](#)
[Pull requests 2](#)
[Actions](#)
[Projects](#)
[Wiki](#)
[Security](#)
[Insights](#)
[Settings](#)

main
10 branches
0 tags
Go to file
Add file
Code



jk-fhswf docs(Streamlit):

 Added hackaday reference ...

✓ 901f56a
16 minutes ago
23 commits

	.github/workflows	Update main.yml	yesterday
	.vscode	Jp text (#10)	31 minutes ago
	doc	Jp text (#10)	31 minutes ago
	pki_a22_app	Jp text (#10)	31 minutes ago
	resources	Jp update readme (#9)	yesterday
	tests	updated image name	yesterday
	.gitignore	Jp text (#10)	31 minutes ago
	README.md	docs(Streamlit):  Added hackaday reference	16 minutes ago
	app.py	Jp update readme (#9)	yesterday
	app_tkinter.py	Bugfix: Screensize adaptiert für Unix Systeme	5 days ago
	packages.txt	Jp text (#10)	31 minutes ago
	poetry.lock	Jp update readme (#9)	yesterday
	poetry.toml	Init project with poetry (#1)	last month
	pyproject.toml	Jp update readme (#9)	yesterday

☰
README.md
✎

Haar Cascade Evaluation Apps with Streamlit and Tkinter


About


No description provided.

Readme


0 stars

2 watching


0 forks



Releases


No releases published
[Create a new release](#)



Packages

No packages published
[Publish your first package](#)


Contributors


 jk-fhswf


 peter-ak


 aLfu911



Languages

Abbildung 3: Git-Repository auf GitHub

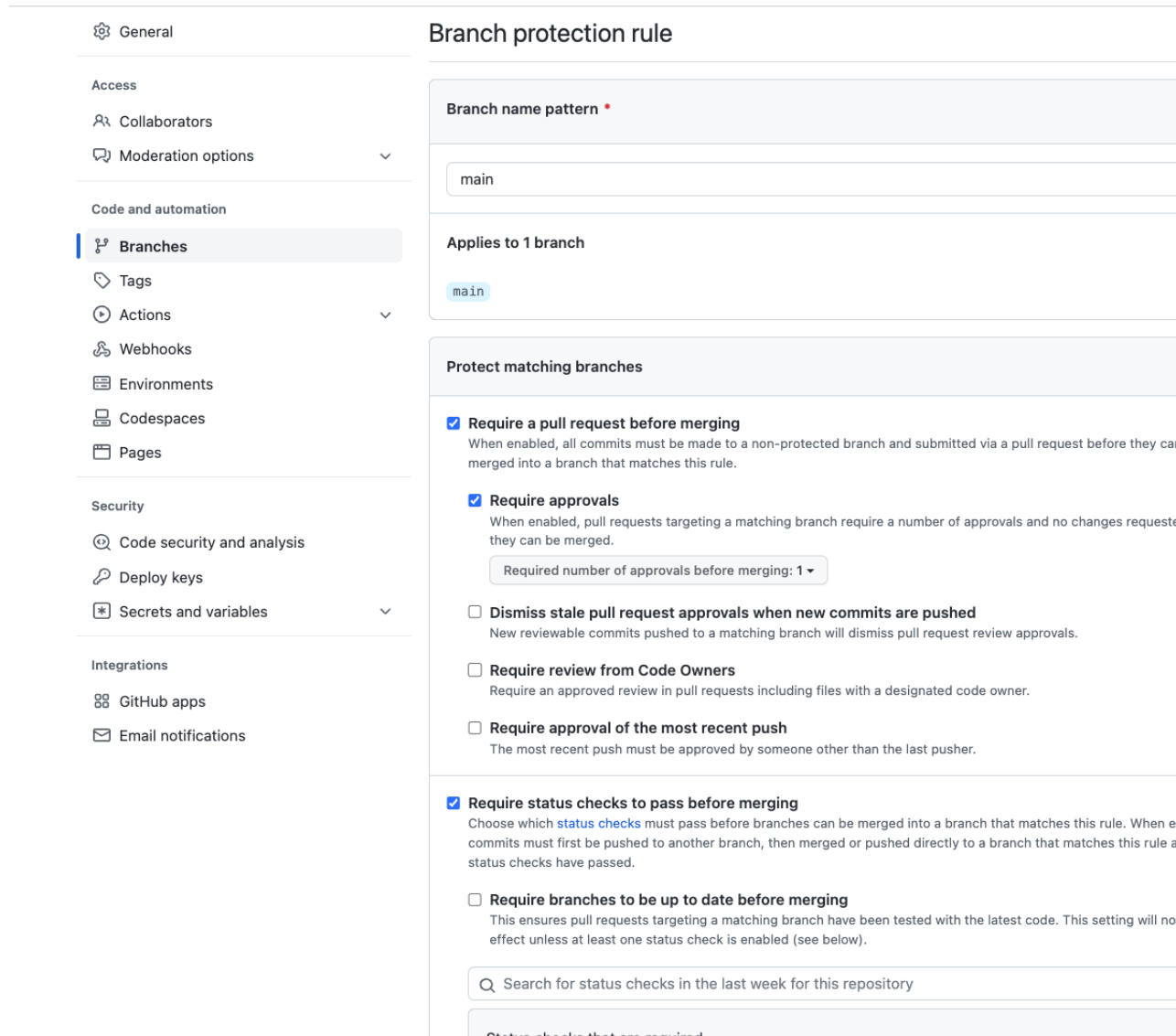


Abbildung 4: Konfiguration von QA-Optionen

GitHub interface showing the workflow execution for the repository `jk-fhswf / pki-a22-app`. The workflow is named `docs(Streamlit): :memo: Added hackaday reference #10`.

The workflow file is `.github/workflows/main.yml`.

The workflow summary shows the following jobs:

- `pytest check` (succeeded 12 minutes ago in 24s)

The workflow details show the following steps:

- Set up job
- Run actions/checkout@v2
- Run actions/setup-python@v2
- cache poetry install
- Run snok/install-poetry@v1
- cache deps
- Run poetry install --no-interaction --no-root
- Run poetry install --no-interaction
- Run poetry run pytest

The output of the `Run poetry run pytest` step is as follows:

```
1  ▶ Run poetry run pytest
8  ===== test session starts =====
9  platform linux -- Python 3.9.16, pytest-7.2.0, pluggy-1.0.0
10 rootdir: /home/runner/work/pki-a22-app/pki-a22-app
11 collected 5 items
12
13 tests/test_pki_a22_app.py . [ 20%]
14 tests/haarcascades/test_haarcascades.py . [ 40%]
15 tests/utils/test_file_loader.py ... [100%]
16
17 ===== 5 passed in 0.65s =====
```

Abbildung 5: Beispiel Ausführung von Tests mit pytest