Übung 9

Table of Contents

1. Lösungsidee 2
2. Tests 3
  2.1. Hinzufügen einer Github-Organisation 3
  2.2. Hinzufügen einer großen Github-Organization (dotnet) 3
  2.3. Hinzufügen mit einer ungültigen Url 4
3. Verwendete Bibliotheken 5
4. Quellcode 6

1. Lösungsidee

Für das Scraping von Github-Organizations wurde das Flask Projekt verwendet. Zum Eingeben der Github Url wurde ein eigenes Textfeld ergänzt.

Wenn eine Url angegeben wurde, wird mittels Action `github-url` die entsprechende Methode aufgerufen. Zu Beginn wird die Url validiert (muss mit https://github beginnen und beim Abruf den Status 200 zurückliefern), anschließend werden die Daten extrahiert. Daten-Scraping und Validierung erfolgt in einem eigenen File `github_helper.py`. Hier werden zu Beginn allgemeine Informationen zur Organisation und alle Repositories mittels Paged-Requests ermittelt. Anschließend wird jedes Repository durchgegangen und die spezifischen Informationen gescraped. Die Daten werden anschließend als JSON zurückgegeben und als .csv abgespeichert. Damit die allgemeinen Informationen in der Übersicht korrekt angezeigt werden, werden diese in einer eigenen Datenbank fileinfo gespeichert. Hier gibt es dann Spalten für name, url, languages, repository_count und members_count. Für die Übersicht werden diese Daten dann geladen und entsprechend in eigenen Spalten angezeigt.

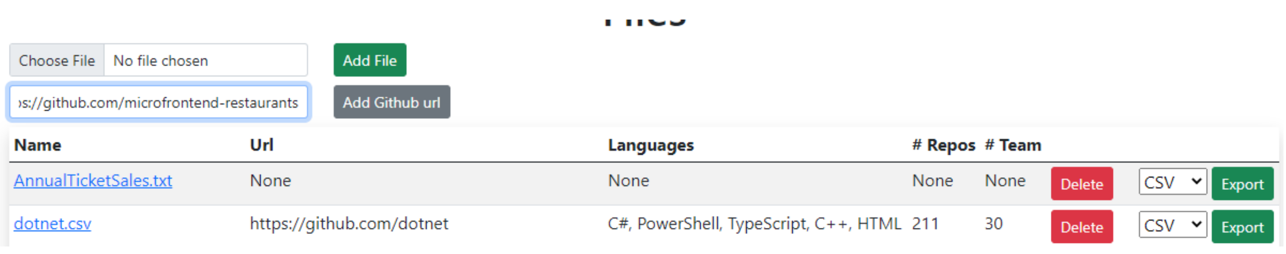
Annahmen:

- Files oder Github-Urls sind eindeutig
- Es werden bei der Url keine ungültigen Zeichen hinzugefügt
- Löschen von Files erfolgt rein über Gui, ansonsten muss das die Datenbank `fileinfo.db` angepasst werden.

2. Tests

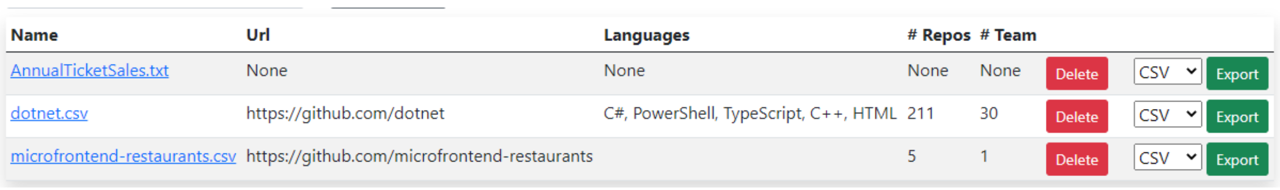
2.1. Hinzufügen einer Github-Organisation

Hinzufügen meiner Bachelorarbeit-Organisation von Micro-Frontends.



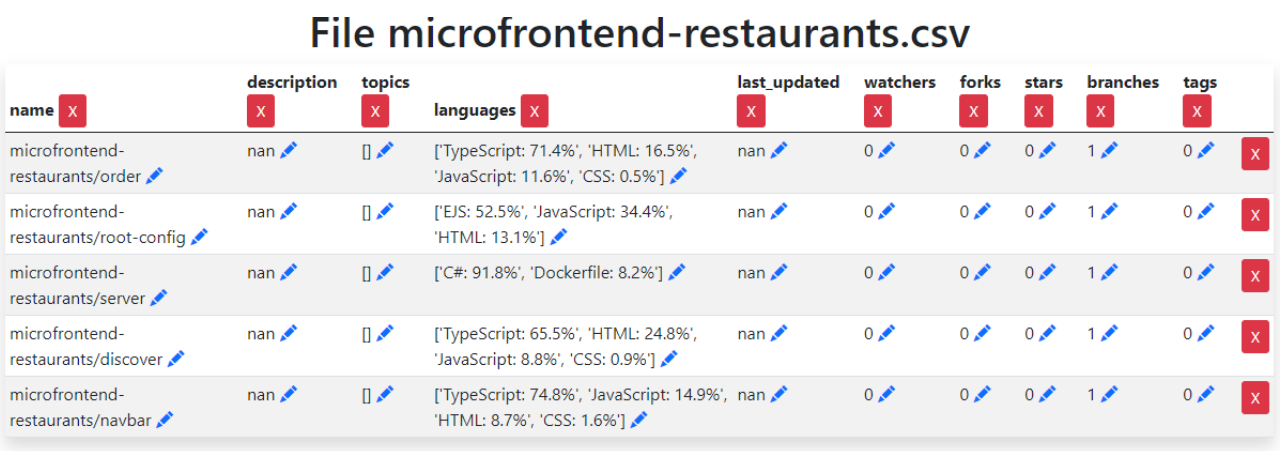
Name	Url	Languages	# Repos	# Team			
AnnualTicketSales.txt	None	None	None	None	Delete	CSV	Export
dotnet.csv	https://github.com/dotnet	C#, PowerShell, TypeScript, C++, HTML	211	30	Delete	CSV	Export

Figure 1. Übersicht



Name	Url	Languages	# Repos	# Team			
AnnualTicketSales.txt	None	None	None	None	Delete	CSV	Export
dotnet.csv	https://github.com/dotnet	C#, PowerShell, TypeScript, C++, HTML	211	30	Delete	CSV	Export
microfrontend-restaurants.csv	https://github.com/microfrontend-restaurants		5	1	Delete	CSV	Export

Figure 2. Übersicht nach Hinzufügen (Languages sind leer, da diese offensichtlich erst nachgeladen werden)



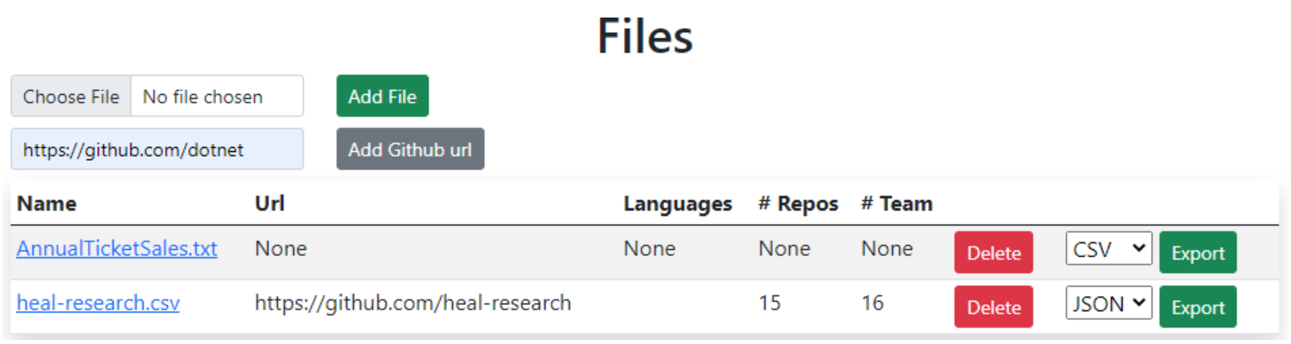
File microfrontend-restaurants.csv

name X	description X	topics X	languages X	last_updated X	watchers X	forks X	stars X	branches X	tags X	
microfrontend-restaurants/order ✏	nan ✏	[] ✏	['TypeScript: 71.4%', 'HTML: 16.5%', 'JavaScript: 11.6%', 'CSS: 0.5%'] ✏	nan ✏	0 ✏	0 ✏	0 ✏	1 ✏	0 ✏	X
microfrontend-restaurants/root-config ✏	nan ✏	[] ✏	['EJS: 52.5%', 'JavaScript: 34.4%', 'HTML: 13.1%'] ✏	nan ✏	0 ✏	0 ✏	0 ✏	1 ✏	0 ✏	X
microfrontend-restaurants/server ✏	nan ✏	[] ✏	['C#: 91.8%', 'Dockerfile: 8.2%'] ✏	nan ✏	0 ✏	0 ✏	0 ✏	1 ✏	0 ✏	X
microfrontend-restaurants/discover ✏	nan ✏	[] ✏	['TypeScript: 65.5%', 'HTML: 24.8%', 'JavaScript: 8.8%', 'CSS: 0.9%'] ✏	nan ✏	0 ✏	0 ✏	0 ✏	1 ✏	0 ✏	X
microfrontend-restaurants/navbar ✏	nan ✏	[] ✏	['TypeScript: 74.8%', 'JavaScript: 14.9%', 'HTML: 8.7%', 'CSS: 1.6%'] ✏	nan ✏	0 ✏	0 ✏	0 ✏	1 ✏	0 ✏	X

Figure 3. Tabelle

2.2. Hinzufügen einer großen Github-Organization (dotnet)

Hinzufügen von https://github.com/dotnet mit 211 Repositories. Nach mehreren Minuten waren alle Repositories gescraped und gespeichert.



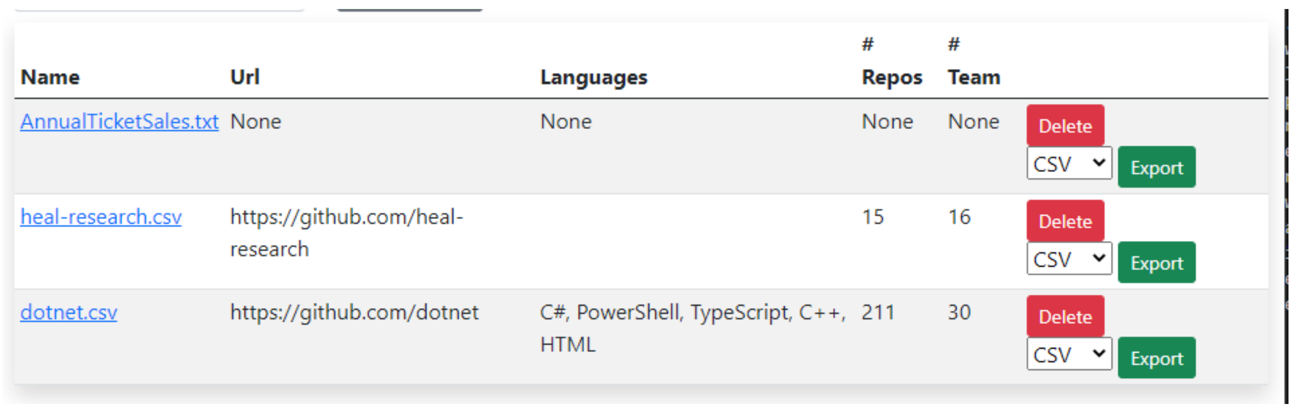
Files

Choose File No file chosen | Add File

https://github.com/dotnet | Add Github url

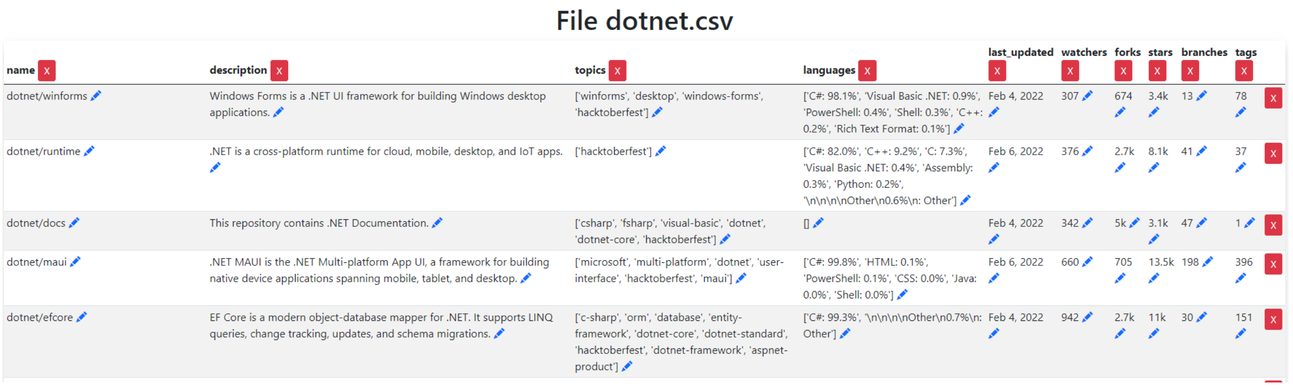
Name	Url	Languages	# Repos	# Team			
AnnualTicketSales.txt	None	None	None	None	Delete	CSV	Export
heal-research.csv	https://github.com/heal-research		15	16	Delete	JSON	Export

Figure 4. Übersicht



Name	Url	Languages	# Repos	# Team	
AnnualTicketSales.txt	None	None	None	None	Delete CSV Export
heal-research.csv	https://github.com/heal-research		15	16	Delete CSV Export
dotnet.csv	https://github.com/dotnet	C#, PowerShell, TypeScript, C++, HTML	211	30	Delete CSV Export

Figure 5. Übersicht nach Hinzufügen

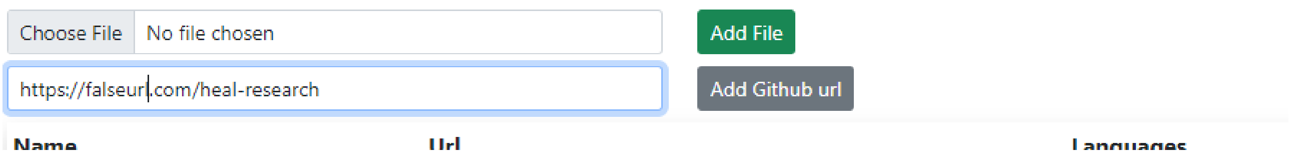


File dotnet.csv

name	description	topics	languages	last_updated	watchers	forks	stars	branches	tags
dotnet/winforms	Windows Forms is a .NET UI framework for building Windows desktop applications.	['winforms', 'desktop', 'windows-forms', 'hacktoberfest']	['C#: 98.1%', 'Visual Basic .NET: 0.9%', 'PowerShell: 0.4%', 'Shell: 0.3%', 'C++: 0.2%', 'Rich Text Format: 0.1%']	Feb 4, 2022	307	674	3.4k	13	78
dotnet/runtime	.NET is a cross-platform runtime for cloud, mobile, desktop, and IoT apps.	['hacktoberfest']	['C#: 82.0%', 'C++: 9.2%', 'C: 7.3%', 'Visual Basic .NET: 0.4%', 'Assembly: 0.3%', 'Python: 0.2%', '\n\n\n\nOther\n0.6%\n: Other']	Feb 6, 2022	376	2.7k	8.1k	41	37
dotnet/docs	This repository contains .NET Documentation.	['csharp', 'fsharp', 'visual-basic', 'dotnet', 'dotnet-core', 'hacktoberfest']	[]	Feb 4, 2022	342	5k	3.1k	47	1
dotnet/maui	.NET MAUI is the .NET Multi-platform App UI, a framework for building native device applications spanning mobile, tablet, and desktop.	['microsoft', 'multi-platform', 'dotnet', 'user-interface', 'hacktoberfest', 'maui']	['C#: 99.8%', 'HTML: 0.1%', 'PowerShell: 0.1%', 'CSS: 0.0%', 'Java: 0.0%', 'Shell: 0.0%']	Feb 6, 2022	660	705	13.5k	198	396
dotnet/efcore	EF Core is a modern object-database mapper for .NET. It supports LINQ queries, change tracking, updates, and schema migrations.	['c-sharp', 'orm', 'database', 'entity-framework', 'dotnet-core', 'dotnet-standard', 'hacktoberfest', 'dotnet-framework', 'aspnet-product']	['C#: 99.3%', '\n\n\n\nOther\n0.7%\n: Other']	Feb 4, 2022	942	2.7k	11k	30	151

Figure 6. Tabelle

2.3. Hinzufügen mit einer ungültigen Url

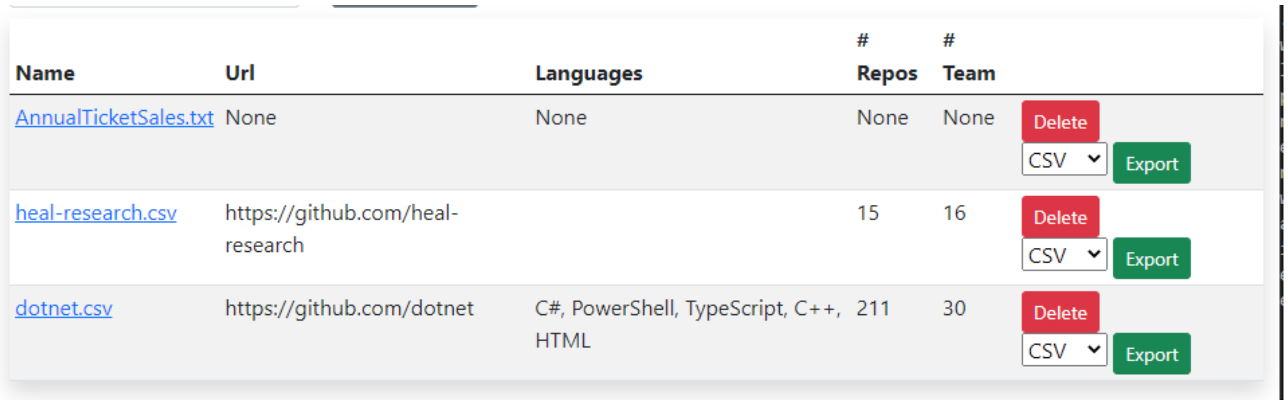


Choose File No file chosen | Add File

https://falseurl.com/heal-research | Add Github url

Name Url Languages

Figure 7. Hinzfügen einer ungültige Url



Name	Url	Languages	# Repos	# Team	
AnnualTicketSales.txt	None	None	None	None	Delete CSV Export
heal-research.csv	https://github.com/heal-research		15	16	Delete CSV Export
dotnet.csv	https://github.com/dotnet	C#, PowerShell, TypeScript, C++, HTML	211	30	Delete CSV Export

Figure 8. Ausgabe des Fehlers

3. Verwendete Bibliotheken

Listing 1. requirements.txt

```
pandas~=1.4.0
Flask~=2.0.2
requests~=2.27.1
beautifulsoup4~=4.10.0
```

4. Quellcode

Listing 2. app.py

```python
import os
import pathlib

from flask import Flask, render_template, request, redirect, send_from_directory
import pandas as pd

import file_helper
import github_helper

app = Flask(__name__)


@app.route('/', methods=['GET'])
def get_files():
    files = file_helper.get_fileinfos()
    return render_template('index.html', files=files)


@app.route('/delete/<filename>', methods=['GET'])
def delete_file(filename: str):
    file_helper.remove_file(filename)
    return redirect('/')


@app.route('/export/<filename>', methods=['POST'])
def export_file(filename: str):
    filename = os.path.join(file_helper.output_directory, filename)
    filepath = pathlib.Path(filename)
    source_filetype = filepath.suffix
    df = file_helper.parse_file(filename, source_filetype)
    target_file_type = request.form['file_type']
    target_file_name = f"{filepath.stem}{target_file_type}"
    file_helper.save_file(df, target_file_name, target_file_type, target_directory='temp')
    return send_from_directory('temp', target_file_name)


@app.route('/github-url', methods=['POST'])
def add_url():
    try:
        url = request.form['url']
        # Validate url
        if github_helper.is_valid_github_url(url):
            # Extract data
            data = github_helper.extract_github_data(url)
            title = data["title"]
            members = data["member_count"]
            languages = data["languages"]
            repositories = data["repositories"]

            # Create file
            df = pd.DataFrame(repositories)
            file_helper.save_file(df, f"{title}.csv", ".csv", None)
            file_helper.add_fileinfo_entry(f"{title}.csv", url, ', '.join(languages), len
(repositories), members)
```

```
        else:
            raise ValueError('Url is invalid')
    except BaseException as e:
        print("Error: ", e)

    return redirect('/')


@app.route('/details/<filename>', methods=['GET'])
def get_file(filename: str):
    try:
        return render_template('details.html', filename=filename, data_frame=file_helper
.get_dataframe(filename))
    except BaseException as e:
        print("Error: ", e)
        return redirect("/")


@app.route('/delete-col/<filename>/<index>', methods=['GET'])
def delete_col(filename: str, index: str):
    df = file_helper.delete_col_or_row(filename, int(index), is_row=False)

    if df.empty:
        delete_file(filename)
        return redirect("/")

    return redirect(f'/details/{filename}')


@app.route('/delete-row/<filename>/<index>', methods=['GET'])
def delete_row(filename: str, index: str):
    df = file_helper.delete_col_or_row(filename, int(index), is_row=True)

    if df.empty:
        delete_file(filename)
        return redirect("/")

    return redirect(f'/details/{filename}')


@app.route('/', methods=['POST'])
def add_file():
    file = request.files['file']
    filename = file.filename
    try:
        if filename == '':
            raise ValueError('No file specified')
        else:
            file_type = os.path.splitext(filename)[1]
            if file_type not in file_helper.filetypes:
                raise ValueError(f'File {file.filename} has invalid type')
            file.save(file.filename)
        df = file_helper.parse_file(filename, file_type)
        file_helper.save_file(df, filename, file_type)
        file_helper.add_fileinfo_entry(filename)
    except ValueError as e:
        print(f'Error: {e}')
    except BaseException as e:
        print(f'Exception details: {e}')
    finally:
        # Cleanup uploaded file
        if os.path.exists(filename):
            os.remove(filename)
    return redirect('/')


@app.route('/edit/<filename>/<row_index>/<column_index>', methods=['GET'])
def edit_entry(filename: str, row_index: str, column_index: str):
    value = file_helper.get_cell_value(filename, int(row_index), int(column_index))
    return render_template("edit.html", row_index=row_index, column_index=column_index,
filename=filename, value=value)


@app.route('/edit/<filename>/<row_index>/<column_index>', methods=['POST'])
def update_entry(filename: str, row_index: str, column_index: str):
    file_helper.update_cell_value(filename, int(row_index), int(column_index), request.form
["value"])
    return redirect(f'/details/{filename}')


@app.route('/delete/<filename>/<row_index>/<column_index>', methods=['GET'])
def delete_entry(filename: str, row_index: str, column_index: str):
    file_helper.update_cell_value(filename, int(row_index), int(column_index), None)
    return redirect(f'/details/{filename}')


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

Listing 3. file_helper.py

```
import os
import pathlib
import sqlite3

import pandas as pd
from pandas import DataFrame

output_directory = "uploaded_files"
filetypes = ('.csv', '.txt', '.json', '.db')


def remove_file(filename):
    full_filename = os.path.join(output_directory, filename)
    remove_fileinfo_entry(filename)
    if os.path.exists(full_filename):
        os.remove(full_filename)


def get_dataframe(filename):
    full_filename = os.path.join(output_directory, filename)
    file_path = pathlib.Path(full_filename)
    return parse_file(full_filename, file_path.suffix)


def save_file(df: DataFrame, filename: str, file_type: str, table_name: str = None,
target_directory: str = None):
    if target_directory is None:
        target_directory = output_directory

    if not os.path.exists(target_directory):
        os.mkdir(target_directory)

    if file_type == '.csv' or file_type == '.txt':
        df.to_csv(os.path.join(target_directory, filename), index=False)
    elif file_type == '.json':
        df.to_json(os.path.join(target_directory, filename), orient="records")
    else:
        conn = sqlite3.connect(os.path.join(target_directory, filename))

        # Generate table name
        if table_name is None:
            table_name = filename.removesuffix(file_type)

        df.to_sql(table_name, con=conn, index=False, if_exists='replace')
        conn.close()


def get_sqlite_table_name(filename, conn) -> str:
    c = conn.cursor()
    c = c.execute(f"SELECT name FROM sqlite_master WHERE type='table';")
    results = c.fetchall()
    if len(results) == 0:
        conn.close()
        raise ValueError(f"{filename} has no table")
    elif len(results) == 2:
        conn.close()
        raise ValueError(f"{filename} has more than one table")
    # Get first column in first row
    return results[0][0]


def parse_sqlite(filename: str) -> DataFrame:
    conn = sqlite3.connect(filename)
    table_name = get_sqlite_table_name(filename, conn)

    df = pd.read_sql(f"select * from {table_name}", con=conn)
    conn.close()
    return df


def parse_csv(filename: str, separator: str) -> DataFrame:
    return pd.read_csv(filename, encoding="latin-1", delimiter=separator)


def parse_json(filename) -> DataFrame:
    return pd.read_json(filename, encoding="latin-1", orient='records')


def parse_file(name: str, file_type: str) -> DataFrame:
    if not os.path.isfile(name):
        raise ValueError('File does not exist')

    if file_type == '.csv' or file_type == '.txt':
        return parse_csv(name, separator=',')
    elif file_type == '.json':
        return parse_json(name)
```

```python
    else:
        return parse_sqlite(name)


def add_fileinfo_entry(name: str, url=None, languages=None, repositories=None, members=None):
    conn = sqlite3.connect('fileinfo.db')
    c = conn.cursor()
    c.execute('''CREATE TABLE IF NOT EXISTS files (
        name text,
        url text,
        languages text,
        repository_count decimal,
        member_count decimal)''')

    c.execute('''INSERT INTO files (name, url, languages, repository_count, member_count)
                        VALUES(?,?,?,?,?)''', [name, url, languages, repositories, members])

    conn.commit()
    c.close()


def get_fileinfos():
    conn = sqlite3.connect('fileinfo.db')
    c = conn.cursor()
    c.execute('''CREATE TABLE IF NOT EXISTS files (
            name text,
            url text,
            languages text,
            repository_count decimal,
            member_count decimal)''')
    c.execute('''SELECT * FROM files''')
    results = c.fetchall()
    c.close()
    return results


def remove_fileinfo_entry(name: str):
    conn = sqlite3.connect('fileinfo.db')
    c = conn.cursor()
    c.execute('''DELETE FROM files WHERE name = ?''', [name])
    conn.commit()
    c.close()


def get_cell_value(filename: str, row_index: int, column_index: int):
    full_filename = os.path.join(output_directory, filename)
    file_path = pathlib.Path(full_filename)
    df = parse_file(full_filename, file_path.suffix)

    return df.iloc[row_index, column_index]


def update_cell_value(filename: str, row_index: int, column_index: int, value: str | None):
    full_filename = os.path.join(output_directory, filename)
    file_path = pathlib.Path(full_filename)
    file_type = file_path.suffix
    df = parse_file(full_filename, file_path.suffix)

    df.iloc[int(row_index), int(column_index)] = value
    if file_type == '.db':
        conn = sqlite3.connect(full_filename)
        table_name = get_sqlite_table_name(full_filename, conn)
        conn.close()
    else:
        table_name = None

    save_file(df, filename, file_type, table_name)


def delete_col_or_row(filename: str, index: int, is_row: bool):
    full_filename = os.path.join(output_directory, filename)
    file_path = pathlib.Path(full_filename)
    file_type = file_path.suffix
    df = parse_file(full_filename, file_path.suffix)

    if is_row:
        df.drop([index], inplace=True)
    else:
        df.drop(df.columns[index], axis=1, inplace=True)

    # Get table name if necessary
    if file_type == '.db':
        conn = sqlite3.connect(full_filename)
        table_name = get_sqlite_table_name(full_filename, conn)
        conn.close()
    else:
        table_name = None

    save_file(df, filename, file_type, table_name)

    return df
```

Listing 4. github_helper.py

```
import re

import requests
from bs4 import BeautifulSoup

github_base_url = "https://github.com"


def get_github_members(url: str):
    page = 0
    all_members = []
    url = f"{url.replace(github_base_url, github_base_url + '/orgs')}/people"

    while True:
        page = page + 1
        paged_url = f"{url}?page={page}"
        doc = BeautifulSoup(requests.get(paged_url).text, 'html.parser')
        members = doc.select('a[id*="member-"]')
        if len(members) == 0:
            break
        for member in members:
            all_members.append(member.text.strip())
    return all_members


def get_github_repositories(url: str):
    page = 0
    all_repositories = []
    url = f"{url.replace(github_base_url, github_base_url + '/orgs')}/repositories"
    while True:
        page = page + 1
        paged_url = f"{url}?page={page}"
        response = requests.get(paged_url)
        doc = BeautifulSoup(response.text, 'html.parser')
        repository_tags = doc.find("div", {"id": "org-repositories"}).find_all("li")
        repository_count = len(repository_tags)
        if repository_count == 0:
            break
        for repo in repository_tags:
            repository = f'{github_base_url}{repo.find("a")["href"]}'
            all_repositories.append(repository)

    return all_repositories


def get_github_title_and_languages(url: str):
    response = requests.get(url)
    doc = BeautifulSoup(response.text, 'html.parser')
    page_content = doc.find('main')
    title = doc.find('a', {'class': 'Header-link'}).text
    all_language_tags = page_content.find('h4', text="Top languages").parent.find_all('span',
{'itemprop': 'programmingLanguage'})
    languages = [tag.text for tag in all_language_tags]
    return title, languages


def extract_github_data(url: str):
    title, all_languages = get_github_title_and_languages(url)
    members = get_github_members(url)
    repositories = get_github_repositories(url)

    repository_data = []

    for repo in repositories:
        response = requests.get(repo)
        content = BeautifulSoup(response.text, 'html.parser')

        repository_name = repo.removeprefix(github_base_url + "/")

        aboutbox = content.find("h2", text='About').findParent()
        description_info = aboutbox.find('p', {'class': 'f4'})
        if description_info is not None:
            description = description_info.text.strip()
        else:
            description = None

        # Values from infobox
        infobox_numbers = aboutbox.findAll('strong')
        stars = infobox_numbers[0].text.strip()
        watchers = infobox_numbers[1].text.strip()
        forks = infobox_numbers[2].text.strip()
        branches_and_tags = content.find('span', {'class': 'color-fg-muted'}, text=re.compile
("branch(es)?")).parent.parent.findAll(
            "strong")
        branches = branches_and_tags[0].text
        tags = branches_and_tags[1].text

        languages_title_tag = content.find("h2", {"class": "h4"}, text="Languages")
        languages = []
        if languages_title_tag is not None:
            for language_item in languages_title_tag.parent.findAll("li"):
                language_spans = language_item.findAll('span')
                language = f"{language_spans[0].text}: {language_spans[1].text}"
                languages.append(language)

        last_updated_tag = content.find("relative-time")
        if last_updated_tag is not None:
            last_updated = last_updated_tag.text
        else:
            last_updated = None

        topics = []
        for topic_tag in content.findAll('a', {'class': 'topic-tag-link'}):
            topics.append(topic_tag.text.strip())

        repository_data.append({
            "name": repository_name,
            "description": description,
            "topics": topics,
            "languages": languages,
            "last_updated": last_updated,
            "watchers": watchers,
            "forks": forks,
            "stars": stars,
            "branches": branches,
            "tags": tags
        })

    return {
        "title": title,
        "member_count": len(members),
        "languages": all_languages,
        "repositories": repository_data
    }


def is_valid_github_url(url):
    try:
        if not re.match(f'^{github_base_url}/*', url):
            raise ValueError('Url is invalid.')

        response = requests.get(url)

        if not response.ok:
            raise ValueError('Invalid response')

        return True
    except:
        return False
```

Listing 5. base.html

```html
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="icon" type="image/x-icon" href="/static/favicon.ico">
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-
icons@1.7.2/font/bootstrap-icons.css">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet"
          integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
crossorigin="anonymous">
    <!-- <link rel="stylesheet" href="../static/css/main.css"> -->
    {% block head %} {% endblock %}
</head>
<body>
<div class="container-fluid">
    {% block body %} {% endblock %}
</div>
</body>
</html>
```

Listing 6. details.html

```
{% extends 'base.html' %}

{% block head %}
    <title>Files Storage</title>
{% endblock %}

{% block body %}
    <h1 class="text-center">File {{ filename }}</h1>

    <table class="table table-striped table-sm shadow">
        <thead>
        <tr>
            {% for col in data_frame.columns %}
                <th>{{ col }}
                    <a href="/delete-col/{{ filename }}/{{ loop.index - 1 }}" class="btn btn-sm
btn-danger">X</a>
                </th>
            {% endfor %}
            <th></th>
        </tr>
        </thead>
        <tbody>
        {% for row in data_frame.values %}
            {% set outer_loop = loop %}
            <tr>
                {% for value in row %}
                    <td>{{ value }}
                        <a href="/edit/{{ filename }}/{{ outer_loop.index - 1 }}/{{ loop.index -
1 }}"><i
                                class="bi bi-pencil-fill"></i></a>
                    </td>
                {% endfor %}
                <td><a href="/delete-row/{{ filename }}/{{ loop.index - 1 }}" class="btn btn-sm
btn-danger">X</a></td>
            </tr>
        {% endfor %}
        {% for col in data_frame.columns %}
        {% endfor %}
        </tbody>
    </table>
{% endblock %}
```

Listing 7. edit.html

```
{% extends 'base.html' %}

{% block head %}
    <title>Files Storage</title>
{% endblock %}

{% block body %}
    <h1 class="text-center">Edit entry</h1>
    <form action="/edit/{{ filename }}/{{ row_index }}/{{ column_index }}" method="post">

        <label for="entry"></label>
        <input id="entry" type="text" name="value" value="{{ value }}">

        <button class="btn btn-sm btn-success" type="submit">Update value</button>
        <a class="btn btn-sm btn-secondary" href="/details/{{ filename }}">Cancel</a>
        <a class="btn btn-sm btn-danger" href="/delete-value/{{ filename }}/{{ row_index }}/{{
column_index }}">Delete value</a>
    </form>
{% endblock %}
```

Listing 8. index.html

```
{% extends 'base.html' %}

{% block head %}
    <title>Files Storage</title>
{% endblock %}

{% block body %}
    <h1 class="text-center">Files</h1>

    <div class="row justify-content-between">
        <div class="col-12 mb-2">
            <form method="post" action="/" enctype="multipart/form-data" class="row">
                <div class="col-3">
                    <input type="file" name="file" class="form-control form-control-sm"
                           accept=".csv,.txt,.json,.db">
                </div>
                <div class="col-auto">
                    <button class="btn btn-success btn-sm">Add File</button>
                </div>
            </form>
        </div>
        <div class="col-12">
            <form method="post" action="/github-url" class="row">
                <div class="col-3">
                    <input type="text" name="url" class="form-control form-control-sm">
                </div>
                <div class="col-auto">
                    <button class="btn btn-secondary btn-sm">Add Github url</button>
                </div>
            </form>
        </div>
    </div>

    <table class="table table-striped table-sm mt-2 shadow">
```

```
        <thead>
        <tr>
            <th>Name</th>
            <th>Url</th>
            <th>Languages</th>
            <th># Repos</th>
            <th># Team</th>
            <th></th>
        </tr>
        </thead>
        <tbody>
        {% for file in files %}
            <tr>
                <td><a href="/details/{{ file[0] }}">{{ file[0] }}</a></td>
                <td>{{ file[1] }}</td>
                <td>{{ file[2] }}</td>
                <td>{{ file[3] }}</td>
                <td>{{ file[4] }}</td>
                <td>
                    <div class="row">
                        <div class="col-auto">
                            <a href="/delete/{{ file[0] }}" class="btn btn-sm btn-danger">
Delete</a>
                        </div>
                        <div class="col-auto">
                            <form action="/export/{{ file[0] }}" method="post">
                                <select name="file_type">
                                    <option value=".csv">CSV</option>
                                    <option value=".txt">Text</option>
                                    <option value=".db">Sqlite</option>
                                    <option value=".json">JSON</option>
                                </select>
                                <button class="btn btn-sm btn-success">Export</button>
                            </form>
                        </div>
                    </div>
                </td>
            </tr>
        {% endfor %}
        </tbody>
    </table>
{% endblock %}
```