

# Übung 7

## Table of Contents

1. Lösungsidee .....	2
1.1. Auflistung aller Files .....	2
1.2. Laden eines Files .....	2
1.3. Hinzufügen neuer Files .....	2
1.4. Löschen von Files .....	2
1.5. Exportieren von Files .....	2
1.6. Löschen von Zeilen oder Spalten .....	3
1.7. Editieren und Löschen von Einträgen .....	3
1.8. Anmerkung .....	3
2. Tests .....	4
2.1. Korrekte Anzeige der Übersicht .....	4
2.2. Hinzufügen eines gültigen Files .....	4
2.3. Hinzufügen eines ungültigen Files .....	5
2.4. Korrekte Anzeige einer Detailseite .....	5
2.5. Exportieren eines Files .....	6
2.6. Löschen von Spalten und Zeilen .....	6
2.7. Aktualisieren eines Wertes .....	8
3. Verwendete Bibliotheken .....	8
4. Quellcode .....	9

# 1. Lösungsidee

## 1.1. Auflistung aller Files

Um alle Files aufzulisten, wurde eine Methode mit der Route `/` erstellt. Alle Filenamen in dem Verzeichnis `/uploaded_files` werden geladen und `render_template` mitgegeben. Im `index.html` wird dann über diese Liste iteriert und sie werden in einer Tabelle angezeigt.

## 1.2. Laden eines Files

Um die Daten eines Files anzuzeigen, gibt es die Methode mit der Route `/<filename>`. Auf die Detailseite gelangt man über das Anklicken des Dateinamens. Das File wird entsprechend als Pandas-Dataframe geparsed und bei `render_template` zusätzlich mit dem Dateinamen mitgegeben. Falls ein Fehler auftreten sollte (z.B. Filename existiert nicht), wird auf die Startseite weitergeleitet. Ansonsten wird im `details.html` über das DataFrame iteriert. Mit einer zweidimensionalen Schleife kann über `df.values` auf die einzelnen Werte zugegriffen werden und diese entsprechend in einem `<td>` angezeigt werden. Für die Überschriften wird einfach über `df.columns` iteriert.

## 1.3. Hinzufügen neuer Files

Dazu gibt es im `index.html` ein Input-Element vom Typen `file`. Mittels Post-Methode kann ein File hinzugefügt werden. Mit `request.files['file']` kann auf das in der Form angegebene File zugegriffen werden. Es wird entsprechend überprüft, ob das File existiert und ob es hinzugefügt werden kann und einen gültigen Dateitypen aufweist. Anschließend wird das File im Verzeichnis `uploaded_files` gespeichert.

## 1.4. Löschen von Files

Dazu gibt es die Methode mit der Route `/delete/<filename>`. Das File wird einfach aus dem entsprechenden Verzeichnis entfernt.

## 1.5. Exportieren von Files

Hier lässt sich mittels Select für jede Zeile das entsprechende Format auswählen.

Beim Select wurden die Dateiformate statisch eingetragen, da das dynamische Setzen über einen Parameter beim 'render\_template' nicht funktioniert hat. Nach dem Drücken auf Export wird die Methode mit der Route `/export/<filename>` aufgerufen. Das File wird als Dataframe geladen und in das Verzeichnis `temp` mit entsprechendem Dateiformat gespeichert. Schlussendlich kann mit der Methode `send_from_directory` das File an den Client zum Download gesendet werden.

## 1.6. Löschen von Zeilen oder Spalten

In der Detailansicht gibt es bei jeder Zeile und Spalte einen Button zum Löschen. Mit `df.drop` kann der entsprechende Eintrag entfernt werden. Der Methode selbst wird die ausgewählte Zeile oder Spalte mitgegeben. Wenn es keine Einträge mehr gibt, wird das File gelöscht und auf die Startseite zurückgegangen.

## 1.7. Editieren und Löschen von Einträgen

Dazu gibt es eine eigene Ansicht `edit.html`. Wenn auf den Edit-Button bei einer der Zellen gedrückt wird, öffnet sich ein weiteres Fenster, wo der Wert bearbeitet werden kann. Der aktuelle Wert lässt sich mit Zeilen und Spaltenindex ermitteln. Der Wert kann auch komplett gelöscht werden, er wird dann auf `Nothing` gesetzt. Nach dem Editieren der Werte wird der DataFrame aktualisiert und entsprechend abgespeichert.

## 1.8. Anmerkung

- Als Trennzeichen für CSV-Files kann nur `,` verwendet werden
- Datenbankfiles müssen genau eine Tabelle beinhalten

## 2. Tests

### 2.1. Korrekte Anzeige der Übersicht

Übersicht wird korrekt angezeigt.

Choose File

No file chosen

Add

Filename

AnnualTicketSales.csv

Delete

CSV

Export

annual\_gold\_rate.txt

Delete

CSV

Export

belgium.db

Delete

CSV

Export

blog.json

Delete

CSV

Export

canada.db

Delete

CSV

Export

credit\_data.txt

Delete

CSV

Export

DOGE-INR.csv

Delete

CSV

Export

One\_Piece\_json.json

Delete

Sqlite

Export

Figure 1. Übersicht

### 2.2. Hinzufügen eines gültigen Files

File wird korrekt hinzugefügt.

Choose File

test.csv

Add

Filename

<a href="#">AnnualTicketSales.csv</a>	Delete	CSV	Export
<a href="#">annual_gold_rate.txt</a>	Delete	CSV	Export
<a href="#">belgium.db</a>	Delete	CSV	Export
<a href="#">blog.json</a>	Delete	CSV	Export
<a href="#">canada.db</a>	Delete	CSV	Export
<a href="#">credit_data.txt</a>	Delete	CSV	Export
<a href="#">DOGE-INR.csv</a>	Delete	CSV	Export
<a href="#">One Piece json.json</a>	Delete	CSV	Export

Figure 2. Übersicht vor Hinzufügen

Choose File	No file chosen	Add
Filename		
<a href="#">AnnualTicketSales.csv</a>	Delete	CSV Export
<a href="#">annual_gold_rate.txt</a>	Delete	CSV Export
<a href="#">belgium.db</a>	Delete	CSV Export
<a href="#">blog.json</a>	Delete	CSV Export
<a href="#">canada.db</a>	Delete	CSV Export
<a href="#">credit_data.txt</a>	Delete	CSV Export
<a href="#">DOGE-INR.csv</a>	Delete	CSV Export
<a href="#">One_Piece.json.json</a>	Delete	CSV Export
<a href="#">test.csv</a>	Delete	CSV Export

Figure 3. Übersicht nach Hinzufügen

## 2.3. Hinzufügen eines ungültigen Files

Ungültiges File wird nicht hochgeladen und hinzugefügt.

Choose File	Test.xlsx	Add
Filename		
<a href="#">AnnualTicketSales.csv</a>	Delete	CSV Export
<a href="#">annual_gold_rate.txt</a>	Delete	CSV Export
<a href="#">belgium.db</a>	Delete	CSV Export
<a href="#">blog.json</a>	Delete	CSV Export
<a href="#">canada.db</a>	Delete	CSV Export
<a href="#">credit_data.txt</a>	Delete	CSV Export
<a href="#">DOGE-INR.csv</a>	Delete	CSV Export
<a href="#">One_Piece.json.json</a>	Delete	CSV Export
<a href="#">valid_file.csv</a>	Delete	CSV Export

Figure 4. Übersicht vor Hinzufügen (xlsx)

```

127.0.0.1 - - [29/Jan/2022 10:04:14] "GET / HTTP/1.1" 200 -
Error: File Test.xlsx has invalid type
127.0.0.1 - - [29/Jan/2022 10:05:24] "POST / HTTP/1.1" 302 -
127.0.0.1 - - [29/Jan/2022 10:05:24] "GET / HTTP/1.1" 200 -

```

Figure 5. Konsole: Uploaden eines ungültigen Files

## 2.4. Korrekte Anzeige einer Detailseite

Daten werden in tabellarischer Form angezeigt.

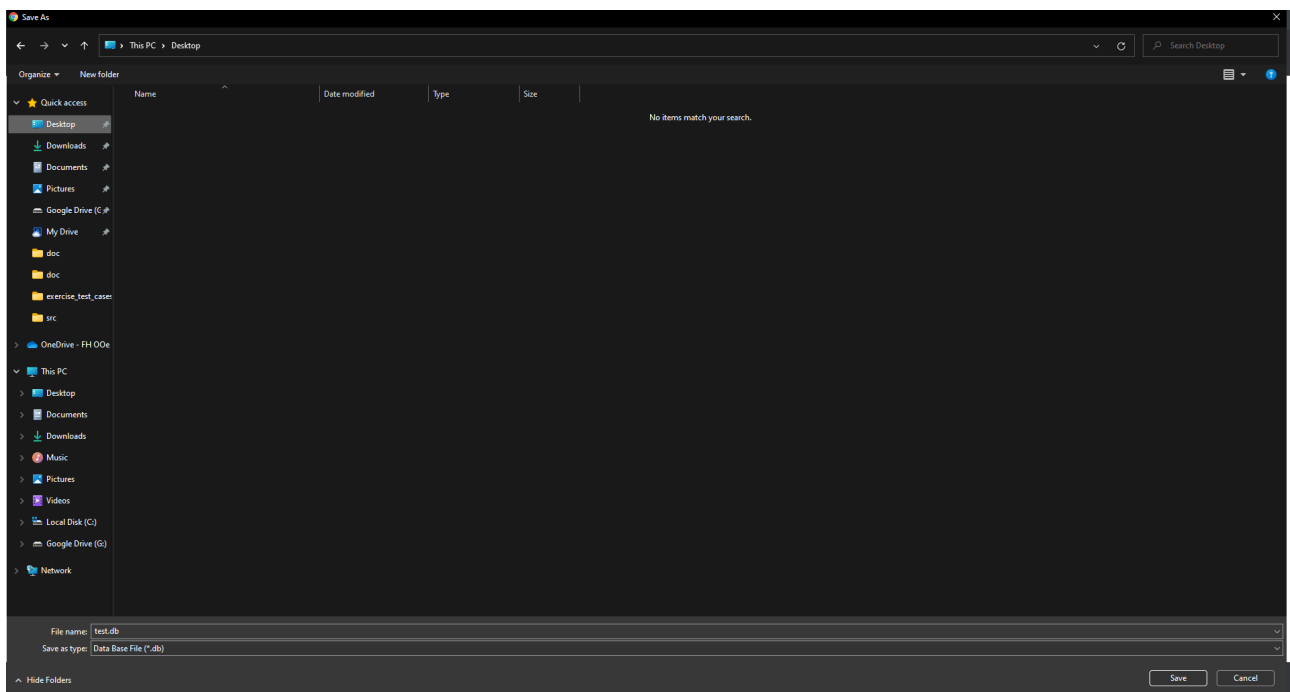
## File DOGE-INR.csv

Date	Open	High	Low	Close	Adj Close	Volume	
2017-11-11	0.074644	0.081936	0.074328	0.078261	0.078261	145377173.0	
2017-11-12	0.077447	0.078832	0.065263	0.067637	0.067637	214308634.0	
2017-11-13	0.068183	0.079276	0.066426	0.079244	0.079244	162349496.0	
2017-11-14	0.078565	0.081071	0.074006	0.077453	0.077453	173986236.0	
2017-11-15	0.07777	0.088193	0.077083	0.087366	0.087366	185350147.0	
2017-11-16	0.087965	0.092942	0.084053	0.090772	0.090772	223488323.0	
2017-11-17	0.090182	0.091046	0.082847	0.085371	0.085371	181214075.0	
2017-11-18	0.085034	0.090012	0.081523	0.089252	0.089252	107153383.0	
2017-11-19	0.088105	0.092407	0.087143	0.089469	0.089469	93076117.0	
2017-11-20	0.089352	0.092965	0.08762	0.090714	0.090714	139391829.0	
2017-11-21	0.091811	0.093843	0.084653	0.09011	0.09011	197701704.0	
2017-11-22	0.090895	0.133159	0.089072	0.118258	0.118258	1222261079.0	
2017-11-23	0.118015	0.145046	0.118015	0.122063	0.122063	1686596208.0	
2017-11-24	0.125266	0.127672	0.115624	0.123399	0.123399	619924033.0	
2017-11-25	0.124339	0.133976	0.121375	0.133976	0.133976	351218243.0	
2017-11-26	0.134035	0.137908	0.121568	0.131048	0.131048	496046708.0	

Figure 6. DODGE-INR.csv

## 2.5. Exportieren eines Files

File test.csv kann als test.db exportiert werden.



## 2.6. Löschen von Spalten und Zeilen

Spalte Low wird entfernt mittels X Button.

## File DOGE-INR.csv

Date	Open	High	Low	Close	Adj Close	Volume	
2017-11-11	0.074644	0.081936	0.074328	0.078261	0.078261	145377173.0	
2017-11-12	0.077447	0.078832	0.065263	0.067637	0.067637	214308634.0	
2017-11-13	0.068183	0.079276	0.066426	0.079244	0.079244	162349496.0	
2017-11-14	0.078565	0.081071	0.074006	0.077453	0.077453	173986236.0	
2017-11-15	0.07777	0.088193	0.077083	0.087366	0.087366	185350147.0	
2017-11-16	0.087965	0.092942	0.084053	0.090772	0.090772	223488323.0	
2017-11-17	0.090182	0.091046	0.082847	0.085371	0.085371	181214075.0	
2017-11-18	0.085034	0.090012	0.081523	0.089252	0.089252	107153383.0	
2017-11-19	0.088105	0.092407	0.087143	0.089469	0.089469	93076117.0	
2017-11-20	0.089352	0.092965	0.08762	0.090714	0.090714	139391829.0	
2017-11-21	0.091811	0.093843	0.084653	0.09011	0.09011	197701704.0	
2017-11-22	0.090895	0.133159	0.089072	0.118258	0.118258	1222261079.0	
2017-11-23	0.118015	0.145046	0.118015	0.122063	0.122063	1686596208.0	
2017-11-24	0.125266	0.127672	0.115624	0.123399	0.123399	619924033.0	
2017-11-25	0.124339	0.133976	0.121375	0.133976	0.133976	351218243.0	
2017-11-26	0.134035	0.137908	0.121568	0.131048	0.131048	496046708.0	

Figure 7. Aktuelle Ansicht

## File DOGE-INR.csv

Date	Open	High	Close	Adj Close	Volume	
2017-11-11	0.074644	0.081936	0.078261	0.078261	145377173.0	
2017-11-12	0.077447	0.078832	0.067637	0.067637	214308634.0	
2017-11-13	0.068183	0.079276	0.079244	0.079244	162349496.0	
2017-11-14	0.078565	0.081071	0.077453	0.077453	173986236.0	
2017-11-15	0.07777	0.088193	0.087366	0.087366	185350147.0	
2017-11-16	0.087965	0.092942	0.090772	0.090772	223488323.0	
2017-11-17	0.090182	0.091046	0.085371	0.085371	181214075.0	
2017-11-18	0.085034	0.090012	0.089252	0.089252	107153383.0	
2017-11-19	0.088105	0.092407	0.089469	0.089469	93076117.0	
2017-11-20	0.089352	0.092965	0.090714	0.090714	139391829.0	
2017-11-21	0.091811	0.093843	0.09011	0.09011	197701704.0	
2017-11-22	0.090895	0.133159	0.118258	0.118258	1222261079.0	
2017-11-23	0.118015	0.145046	0.122063	0.122063	1686596208.0	
2017-11-24	0.125266	0.127672	0.123399	0.123399	619924033.0	

Figure 8. Löschen der Spalte Low

Erste Zeile mit '2017-11-11' wird gelöscht.

## File DOGE-INR.csv



















Date 	Open 	High 	Close 	Adj Close 	Volume 	
2017-11-12 	0.077447 	0.078832 	0.067637 	0.067637 	214308634.0 	
2017-11-13 	0.068183 	0.079276 	0.079244 	0.079244 	162349496.0 	
2017-11-14 	0.078565 	0.081071 	0.077453 	0.077453 	173986236.0 	
2017-11-15 	0.07777 	0.088193 	0.087366 	0.087366 	185350147.0 	
2017-11-16 	0.087965 	0.092942 	0.090772 	0.090772 	223488323.0 	
2017-11-17 	0.090182 	0.091046 	0.085371 	0.085371 	181214075.0 	
2017-11-18 	0.085034 	0.090012 	0.089252 	0.089252 	107153383.0 	
2017-11-19 	0.088105 	0.092407 	0.089469 	0.089469 	93076117.0 	
2017-11-20 	0.089352 	0.092965 	0.090714 	0.090714 	139391829.0 	

Figure 9. Löschen der ersten Zeile

## 2.7. Aktualisieren eines Wertes

Wert Open in der ersten Zeile wird auf 1234.0 aktualisiert.

# Edit entry

Update value

Cancel

Delete value

Figure 10. Seite zum Editieren


































Date 	Open 	High 	Close 	Adj Close 	Volume 	
2017-11-12 	1234.0 	0.078832 	0.067637 	0.067637 	214308634.0 	
2017-11-13 	0.068183 	0.079276 	0.079244 	0.079244 	162349496.0 	
2017-11-14 	0.078565 	0.081071 	0.077453 	0.077453 	173986236.0 	
2017-11-15 	0.07777 	0.088193 	0.087366 	0.087366 	185350147.0 	

Figure 11. Nach Editieren (Wert wurde auf 1234.0 gesetzt)

## 3. Verwendete Bibliotheken

Listing 1. requirements.txt

```
pandas~=1.4.0
Flask~=2.0.2
```



## 4. Quellcode

Listing 2. main.py

```
import os
import pathlib
import sqlite3
from flask import Flask, render_template, request, redirect, send_from_directory
import pandas as pd
from pandas import DataFrame

app = Flask(__name__)

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

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, delimiter=separator)

def parse_json(filename) -> DataFrame:
    return pd.read_json(filename, 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)
    else:
        return parse_sqlite(name)
```

```
@app.route('/', methods=['GET'])
def get_files():
    files = [f for f in os.listdir(output_directory)]
    return render_template('index.html', files=files)

@app.route('/delete/<filename>', methods=['GET'])
def delete_file(filename: str):
    if os.path.exists(f"{output_directory}/{filename}"):
        os.remove(f"{output_directory}/{filename}")
    return redirect('/')

@app.route('/export/<filename>', methods=['POST'])
def export_file(filename: str):
    filename = os.path.join(output_directory, filename)
    filepath = pathlib.Path(filename)
    source_filetype = filepath.suffix
    df = parse_file(filename, source_filetype)
    if not os.path.exists('temp'):
        os.mkdir('temp')
    target_file_type = request.form['file_type']
    target_file_name = f"{filepath.stem}-{target_file_type}"
    save_file(df, 'temp', target_file_name, target_file_type)
    return send_from_directory('temp', target_file_name)

@app.route('/<filename>', methods=['GET'])
def get_file(filename: str):
    try:
        full_filename = os.path.join(output_directory, filename)
        file_path = pathlib.Path(full_filename)
        df = parse_file(full_filename, file_path.suffix)
        return render_template('details.html', filename=filename, data_frame=df)
    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 = delete_col_or_row(filename, int(index), is_row=False)

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

    return redirect(f'/{filename}')

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

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

```
return redirect(f'/{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 filetypes:
                raise ValueError(f'File {file.filename} has invalid type')
            file.save(file.filename)
            df = parse_file(filename, file_type)
            save_file(df, output_directory, filename, file_type)
    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):
    full_filename = os.path.join(output_directory, filename)
    file_path = pathlib.Path(full_filename)
    df = parse_file(full_filename, file_path.suffix)

    value = df.iloc[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):
    set_entry(filename, row_index, column_index, request.form["value"])
    return redirect(f'/{filename}')

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

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, output_directory, filename, file_type, table_name)

    return df


def save_file(df: DataFrame, output_dir: str, filename: str, file_type: str, table_name: str =
None):
    if file_type == '.csv' or file_type == '.txt':
        df.to_csv(os.path.join(output_dir, filename), index=False)
    elif file_type == '.json':
        df.to_json(os.path.join(output_dir, filename), orient="records")
    else:
        conn = sqlite3.connect(os.path.join(output_dir, 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 set_entry(filename: str, row_index: str, column_index: str, 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, output_directory, filename, file_type, table_name)


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

*Listing 3. base.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/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTWFspD3yD65VohhpuuC0mLASjC"
crossorigin="anonymous">
  <!-- <link rel="stylesheet" href="../static/css/main.css"> -->
  {% block head %} {% endblock %}
</head>
<body>
<div class="container">
  {% block body %} {% endblock %}
</div>
</body>
</html>
```

## Listing 4. 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 %}
            <tr>
                <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 %}
                <tr>
                    <td>{{ col }}
                        <a href="/delete-col/{{ filename }}/{{ loop.index - 1 }}" class="btn btn-sm
btn-danger">X</a>
                    </td>
                </tr>
            {% endfor %}
        </tbody>
    </table>
{% endblock %}

```

*Listing 5. 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="/{{ filename }}">Cancel</a>
        <a class="btn btn-sm btn-danger" href="/delete-value/{{ filename }}/{{ row_index }}/{{
column_index }}">Delete value</a>
    </form>
{% endblock %}
```

Listing 6. index.html

```

{% extends 'base.html' %}

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

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

    <form method="post" action="/" enctype="multipart/form-data" class="row">
        <div class="col-auto">
            <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</button>
        </div>
    </form>

    <table class="table table-striped table-sm mt-2 shadow">
        <thead>
            <tr>
                <th>Filename</th>
            </tr>
        </thead>
        <tbody>
            {% for file in files %}
                <tr>
                    <td> <a href="/{{ file }}">{{ file }}</a></td>
                    <td>
                        <div class="row">
                            <div class="col-auto">
                                <a href="/delete/{{ file }}" class="btn btn-sm btn-danger">
Delete</a>
                            </div>
                            <div class="col-auto">
                                <form action="/export/{{ file }}" 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 %}

```