Backend:

import base64, certifi, json

import datetime

import bcrypt

import jwt

from flask import Flask, request, jsonify

from flask\_cors import CORS

from pymongo import MongoClient

import gridfs

from detect\_nails import detect\_and\_annotate

from predict\_hb import predict\_hb

app = Flask(\_\_name\_\_)

CORS(app)

uri = 'mongodb+srv://jakobscharf:mIFL16blqCB3O9V1@onlynails.zjtlwkc.mongodb.net/?retryWrites=true&w=majority&appName=OnlyNails'

client = MongoClient(uri, tlsCAFile=certifi.where())

db = client["onlynails"]

fs = gridfs.GridFS(db, collection="pictures")

# Key zur Generierung vom Login-Token -- nicht ändern!

SECRET\_KEY = "deinemamaistdickeralsmeinemamaaberdasistgarnichtschlimmweilmenschistmenschausserinkendieistkackiundfaul"

'''

Wichtiges To-Do vor Vorstellung: die 'best.pt' und 'model\_rf\_rgb\_avg.pkl' gegen richtig trainierte austauschen

CHECK -- Bilder anonym in DB ablegen für KI-Training Zwecke

CHECK -- Visualisierung HG-Werte

CHECK -- Hb-Werte in DB hinterlegen

CHECK -- API Endpoint für die Auswertung

CHECK -- Profil: Werte direkt bei Aufruf; Meds und Krankheiten werden doppelt eingetragen

CHECK -- Einheitliche Schriftarten

CHECK -- Daily Stuff: In DB hinterlegen + Speicher-Funktion

CHECK -- Passwort Hashing (per bcrypt) -> nur wirklich sicher, wenn Passwort mind. 12 Zeichen und Sonderzeichen etc.

CHECK -- Code dokumentieren und aufräumen

CHECK -- Analyse: Werte, Bilder etc. im sessionStorage speichern

CHECK -- Token Logik implementieren für Security der Endpoints (Enkrypting per HS256 und geheimen Key)

- Erkannte Boxen löschen implementieren (über Nummerierung der Boxen)

Zusatz CHECK: Autologin, wenn username, password noch lokal gespeichert sind

'''

def check\_token(auth\_header):

try:

token = auth\_header.split(" ")[1]

jwt.decode(token, SECRET\_KEY, algorithms=['HS256'])

return "success"

except jwt.ExpiredSignatureError:

return "Token abgelaufen"

except jwt.InvalidTokenError:

return "Token ungültig"

@app.route('/userdata', methods=['GET'])

def get\_userdata():

token\_response = check\_token(request.headers.get('Authorization'))

if token\_response == "success":

collection = db["userdata"]

user\_id = request.args.get("uid")

user\_dok = collection.find\_one({"uid": user\_id})

if user\_dok:

user\_dok["\_id"] = str(user\_dok["\_id"])

return jsonify(user\_dok), 200

else:

collection.insert\_one({"uid": user\_id})

return jsonify({"uid": user\_id}), 201

else:

return jsonify({"error": token\_response}), 401

@app.route('/userdata', methods=['POST'])

def update\_userdata():

token\_response = check\_token(request.headers.get('Authorization'))

if token\_response == "success":

collection = db["userdata"]

if not request.is\_json:

return jsonify({"error": "Missing JSON body"}), 400

data = request.get\_json()

user\_id = data.get("uid")

try:

collection.update\_one(

{"uid": user\_id},

{"$set": {

"name": data.get("name"),

"birthday": data.get("birthday"),

"gender": data.get("gender"),

"vorerkrankung": data.get("vorerkrankung"),

"medikamente": data.get("medikamente")

}}

)

return jsonify({"success": True}), 200

except Exception as e:

return jsonify({'success': False, 'error': e}), 401

else:

return jsonify({"error": token\_response}), 401

@app.route('/login', methods=['POST'])

def login():

collection = db["onlynails"]

if not request.is\_json:

return jsonify({"error": "Missing JSON in request"}), 415

data = request.get\_json()

username = data.get('username')

password = data.get('password')

try:

user\_dok = collection.find\_one({"username": username})

if user\_dok and bcrypt.checkpw(password.encode(), user\_dok["password"].encode('utf-8')):

payload = {

'username': username,

'exp': datetime.datetime.utcnow() + datetime.timedelta(hours=24)

}

token = jwt.encode(payload, SECRET\_KEY, algorithm='HS256')

user\_id = str(user\_dok['\_id'])

return jsonify({'success': True, 'id': user\_id, 'token': token}), 200

else:

return jsonify({'success': False}), 401

except Exception as e:

return jsonify({'success': False, 'error': e}), 401

@app.route('/register', methods=['POST'])

def register():

collection = db["onlynails"]

if not request.is\_json:

return jsonify({"error": "Missing JSON in request"}), 415

data = request.get\_json()

username = data.get('username')

password = data.get('password')

existing\_user = collection.find\_one({"username": username})

if not existing\_user:

try:

hashed\_pw = bcrypt.hashpw(password.encode('utf-8'), bcrypt.gensalt()).decode('utf-8')

user\_dok = {

"username": username,

"password": hashed\_pw

}

new\_user = collection.insert\_one(user\_dok)

return jsonify({'success': True, 'id': str(new\_user.inserted\_id)}), 200

except Exception as e:

return jsonify({'success': False, 'error': e}), 401

else:

return jsonify({'success': False}), 401

@app.route("/nails/detect", methods=["POST"])

def route\_detect\_nails():

"""

Multipart POST (image)

→ YOLO erkennt Fingernägel, gibt Box-Koordinaten & Preview zurück.

"""

token\_response = check\_token(request.headers.get('Authorization'))

if token\_response == "success":

if "file" not in request.files:

return jsonify({"error": 'Field "file" missing'}), 400

img\_bytes = request.files["file"].read()

# Upload des Bildes in DB für Training

fs.put(img\_bytes)

boxes, jpg\_bytes, \_ = detect\_and\_annotate(img\_bytes)

b64 = base64.b64encode(jpg\_bytes).decode("ascii")

return jsonify({

"bboxes": boxes, # [{id,x1,y1,x2,y2,score}, …]

"annotated": f"data:image/jpeg;base64,{b64}"

}), 200

else:

return jsonify({"error": token\_response}), 401

@app.route("/hb/predict", methods=["POST"])

def route\_predict\_hb():

"""

Multipart POST (image)

→ Hb-Schätzung unter Einbeziehung \*aller\* erkannten Boxen.

"""

token\_response = check\_token(request.headers.get('Authorization'))

if token\_response == "success":

if "file" not in request.files:

return jsonify({"error": 'Field "file" missing'}), 400

elif "uid" not in request.form:

return jsonify({"error": 'Field "uid" missing'}), 400

elif "date" not in request.form:

return jsonify({"error": 'Field "date" missing'}), 400

try:

hb\_val = predict\_hb(request.files["file"].read())

hb = round(hb\_val, 1)

safe\_hb\_in\_db(hb, request.form.get("uid"), request.form.get("date"))

return jsonify({"hb": hb}), 200

except Exception as e:

return jsonify({"error": str(e)}), 500

else:

return jsonify({"error": token\_response}), 401

# anonymously saving the hb value in the db

def safe\_hb\_in\_db(hb, uid, date):

db["hb\_values"].insert\_one({

"uid": uid,

"date": date,

"hb": hb,

})

@app.route("/hb/predict-custom", methods=["POST"])

def predict\_hb\_custom():

"""

Multipart POST

image : JPEG

keep\_ids : JSON-Liste (optional) → nur diese Box-IDs verwenden

drop\_ids : JSON-Liste (optional) → bestimmte Box-IDs ausschließen

"""

token\_response = check\_token(request.headers.get('Authorization'))

if token\_response == "success":

if "image" not in request.files:

return jsonify({"error": 'Field "image" missing'}), 400

keep\_ids = drop\_ids = None

if "keep\_ids" in request.form:

keep\_ids = json.loads(request.form["keep\_ids"])

elif "drop\_ids" in request.form:

drop\_ids = json.loads(request.form["drop\_ids"])

try:

hb\_val = predict\_hb(request.files["image"].read(),

keep\_ids=keep\_ids,

drop\_ids=drop\_ids)

return jsonify({"hb": round(hb\_val, 1)}), 200

except Exception as e:

return jsonify({"error": str(e)}), 500

else:

return jsonify({"error": token\_response}), 401

@app.route("/daily", methods=["POST"])

def post\_daily():

token\_response = check\_token(request.headers.get('Authorization'))

if token\_response == "success":

if not request.is\_json:

return jsonify({"error": "Missing JSON body"}), 400

data = request.get\_json()

db["daily"].insert\_one({

"uid": data.get("uid"),

"mood": data.get("mood"),

"symptoms": data.get("symptoms"),

"period": data.get("period"),

"sport": {

"type": data.get("sport").get("type"),

"intensity": data.get("sport").get("intensity"),

"duration": data.get("sport").get("duration"),

"comment": data.get("sport").get("comment"),

},

"date": data.get("date"),

})

return jsonify({"success": True}), 200

else:

return jsonify({"error": token\_response}), 401

@app.route("/hb", methods=["GET"])

def get\_hb():

uid = request.args.get('uid')

if not uid:

return jsonify({"error": "Parameter 'uid' fehlt"}), 400

token\_response = check\_token(request.headers.get('Authorization'))

if token\_response == "success":

cursor = db["hb\_values"].find({"uid": uid})

matching\_values = list(cursor)

if not matching\_values:

return jsonify({"error": "No values found"}), 404

data = [{"date": doc["date"], "value": doc["hb"]} for doc in matching\_values]

print(f"Return data {data}")

return jsonify(data), 200

else:

return jsonify({"error": token\_response}), 401

@app.route("/last-hb", methods=["GET"])

def get\_last\_hb():

uid = request.args.get('uid')

if not uid:

return jsonify({"error": "Parameter 'uid' fehlt"}), 400

token\_response = check\_token(request.headers.get('Authorization'))

if token\_response == "success":

cursor = db["hb\_values"].find({"uid": uid})

matching\_values = list(cursor)

for doc in matching\_values:

doc["parsed\_date"] = datetime.datetime.strptime(doc["date"], "%d.%m.%Y")

max\_date = max(doc["parsed\_date"] for doc in matching\_values)

latest\_docs = [doc for doc in matching\_values if doc["parsed\_date"] == max\_date]

latest\_doc = latest\_docs[0]

data = {"date": latest\_doc["date"], "value": latest\_doc["hb"]}

return jsonify(data), 200

else:

return jsonify({"error": token\_response}), 401

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

# --------------------------------------------------------------

# YOLO-v8 Fingernagel-Erkennung + Vorschaubild

# --------------------------------------------------------------

from pathlib import Path

import cv2, numpy as np

from ultralytics import YOLO

# Ordner, in dem dieses Skript liegt → dort liegt auch best.pt

ROOT = Path(\_\_file\_\_).resolve().parent

YOLO\_WEIGHTS = ROOT / "best.pt" # <-- dein YOLO-Gewicht

# Modell lazy-laden (= erst beim 1. Aufruf)

\_yolo\_model = None

def \_get\_model():

global \_yolo\_model

if \_yolo\_model is None:

\_yolo\_model = YOLO(str(YOLO\_WEIGHTS))

return \_yolo\_model

def detect\_and\_annotate(img\_bytes: bytes,

conf\_thres: float = 0.25):

"""

• Liefert:

boxes : Liste Dikt {"id", "x1","y1","x2","y2","score"}

jpg\_bytes : Annotiertes JPEG

avg\_rgbs : Liste Tuple (R,G,B) pro Box

"""

img\_arr = np.frombuffer(img\_bytes, dtype=np.uint8)

img = cv2.imdecode(img\_arr, cv2.IMREAD\_COLOR) # BGR

model = \_get\_model()

result = model(img, conf=conf\_thres, verbose=False)[0]

boxes, avg\_rgbs = [], []

annotated = img.copy()

for i, box in enumerate(result.boxes.xyxy.cpu().numpy()):

x1, y1, x2, y2 = map(int, box[:4])

score = float(result.boxes.conf[i])

# Box speichern

boxes.append({

"id": i,

"x1": x1, "y1": y1, "x2": x2, "y2": y2,

"score": round(score, 3)

})

# Mittelwert-RGB im Crop

crop = img[y1:y2, x1:x2]

mean\_bgr = crop.mean(axis=(0,1))

avg\_rgbs.append(tuple(float(c) for c in mean\_bgr[::-1])) # → RGB

# Box ins Vorschaubild zeichnen

cv2.rectangle(annotated, (x1,y1), (x2,y2), (0,255,0), 2)

cv2.putText(annotated, f"{i}:{score:.2f}",

(x1, y1-5), cv2.FONT\_HERSHEY\_SIMPLEX,

0.5, (0,255,0), 1, cv2.LINE\_AA)

# BGR → JPEG-Bytes

\_, jpg\_bytes = cv2.imencode(".jpg", annotated)

return boxes, jpg\_bytes.tobytes(), avg\_rgbs

# onlynails\_cli.py – Python 3.8-kompatibel

# --------------------------------------------------------------

import sys, json, base64, tempfile, webbrowser

from io import BytesIO

from pathlib import Path

from typing import Union, List # <-- wichtig für 3.8

import requests

from PIL import Image, ImageDraw

# ---------- Konfiguration --------------------------------------

BACKEND = "http://localhost:5000"

ENDPOINT\_DETECT = f"{BACKEND}/detect\_nails"

ENDPOINT\_PRED = f"{BACKEND}/predict\_hb\_custom"

# --------------------------------------------------------------

# ---------- kleine Utils --------------------------------------

def open\_temp\_jpg(pil\_im: Union[Image.Image, bytes], prefix: str) -> None:

"""

Öffnet ein PIL-Bild oder JPEG-Bytes schnell in der Windows-Fotoanzeige.

"""

if isinstance(pil\_im, bytes):

data = pil\_im

else: # PIL-Objekt

buf = BytesIO()

pil\_im.save(buf, format="JPEG")

data = buf.getvalue()

tmp = tempfile.NamedTemporaryFile(delete=False,

suffix=".jpg",

prefix=prefix)

tmp.write(data)

tmp.close()

print("▶ Vorschau geöffnet:", tmp.name)

webbrowser.open(tmp.name)

def draw\_boxes(orig\_path: Union[str, Path],

boxes: List[dict],

keep\_ids: List[int]) -> Image.Image:

"""

Zeichnet nur die Boxen, deren IDs in keep\_ids enthalten sind.

"""

im = Image.open(orig\_path).convert("RGB")

draw = ImageDraw.Draw(im)

for b in boxes:

bid, x1, y1, x2, y2 = b["id"], b["x1"], b["y1"], b["x2"], b["y2"]

if bid not in keep\_ids:

continue

draw.rectangle([x1, y1, x2, y2], outline="lime", width=4)

draw.text((x1 + 4, y1 + 4), str(bid), fill="lime")

return im

# --------------------------------------------------------------

def main() -> None:

if len(sys.argv) != 2:

print("Aufruf: python onlynails\_cli.py <bilddatei.jpg>")

sys.exit(1)

img\_path = Path(sys.argv[1])

if not img\_path.exists():

print("❌ Datei nicht gefunden:", img\_path)

sys.exit(1)

# ---------- 1) /detect\_nails --------------------------------

with open(img\_path, "rb") as f:

resp = requests.post(ENDPOINT\_DETECT,

files={"image": f})

if resp.status\_code != 200:

print("Fehler bei /detect\_nails:", resp.text)

sys.exit(1)

det = resp.json()

boxes = det["bboxes"] # [{'id':…, 'x1':…}, …]

jpg\_b64 = det["annotated"].split(",", 1)[1]

open\_temp\_jpg(base64.b64decode(jpg\_b64), "first\_")

# ---------- 2) IDs wählen -----------------------------------

print("\nGefundene Box-IDs:", [b["id"] for b in boxes])

ids\_str = input("IDs zum IGNORIEREN (Komma getrennt, leer = alle behalten): ").strip()

if ids\_str:

drop\_ids = [int(x) for x in ids\_str.split(",")]

keep\_ids = [b["id"] for b in boxes if b["id"] not in drop\_ids]

else:

keep\_ids = [b["id"] for b in boxes]

# Zweite Vorschau

im2 = draw\_boxes(img\_path, boxes, keep\_ids)

open\_temp\_jpg(im2, "second\_")

# ---------- 3) /predict\_hb\_custom ---------------------------

files = {"image": open(img\_path, "rb")}

data = {"keep\_ids": json.dumps(keep\_ids)}

resp2 = requests.post(ENDPOINT\_PRED, files=files, data=data)

if resp2.status\_code != 200:

print("Fehler bei /predict\_hb\_custom:", resp2.text)

sys.exit(1)

hb = resp2.json()["hb"]

print(f"\n💉 Geschätzter Hb-Wert: {hb:.1f} g/L")

if \_\_name\_\_ == "\_\_main\_\_":

main()

# predict\_hb.py

# --------------------------------------------------------------

# Nutzt detect\_nails + Random-Forest-Regressor,

# um aus allen (oder gefilterten) Box-Crops den Hb-Wert zu schätzen

# --------------------------------------------------------------

from pathlib import Path

import numpy as np

import joblib

from detect\_nails import detect\_and\_annotate

ROOT = Path(\_\_file\_\_).resolve().parent

RF\_PICKLE = ROOT / "model\_rf\_rgb\_avg.pkl" # <-- dein Pickle-Dateiname

\_regressor = joblib.load(RF\_PICKLE)

def predict\_hb(img\_bytes: bytes,

keep\_ids=None,

drop\_ids=None) -> float:

"""

img\_bytes : Original-JPEG-Bytes der Hand

keep\_ids : Liste Box-IDs → nur diese Boxen werden genutzt

drop\_ids : Liste Box-IDs → diese Boxen werden ignoriert

------------------------------------------------------------

Rückgabe : Hb-Schätzung in g/L (float)

"""

boxes, \_, avg\_rgbs = detect\_and\_annotate(img\_bytes)

# Boxen filtern ------------------------------------------------

if keep\_ids is not None:

mask = [i in keep\_ids for i in range(len(boxes))]

elif drop\_ids is not None:

mask = [i not in drop\_ids for i in range(len(boxes))]

else:

mask = [True]\*len(boxes)

if not any(mask):

raise ValueError("No boxes left after filtering.")

# Mittelwert über die gewählten Boxen -------------------------

sel\_rgbs = np.array([avg\_rgbs[i] for i, m in enumerate(mask) if m])

r\_avg, g\_avg, b\_avg = sel\_rgbs.mean(axis=0)

# Regressor-Eingabe 2D

X = np.array([[r\_avg, g\_avg, b\_avg]])

hb\_pred = float(\_regressor.predict(X)[0])

return hb\_pred

**# README zum Backup**

**## Installation**

1. [Python herunterladen](https://www.python.org/downloads/) und installieren

2. Ordner in einer IDE (VSCode oder PyCharm) öffnen oder über den Terminal aufrufen (Rechtsklick -> Dienste -> Neues Terminal beim Ordner)

3. In der Konsole folgendes ausführen

```bash

pip install flask flask-cors pymongo certifi scikit-learn joblib ultralytics gridfs bcrypt pyjwt

```

4. Über die Konsole das Backend mit folgendem Befehl starten:

```bash

python3 app.py

```

JavaScript für Funktionalitäten

Kamera:

let stream = null;

export function capture() {

document.getElementById("noFingernailsDetected").style.display = "none"

document.getElementById("canvas").style.display = "none";

document.getElementById("video").style.display = "block"

document.getElementById("confirmBoxes").style.display = "none"

document.getElementById("selectBoxes").style.display = "none"

document.getElementById("analyze").style.display = "block"

if (stream) {

stream.getTracks().forEach(track => track.stop());

stream = null;

}

navigator.mediaDevices.getUserMedia({ video: true })

.then(videoStream => {

stream = videoStream;

document.getElementById("video").srcObject = stream;

document.getElementById("takePhoto").style.display = "block";

document.getElementById("takePhotoAgain").style.display = "none";

document.getElementById("openCam").style.display = "none";

document.getElementById("analyze").style.display = "none";

})

.catch(err => {

console.error("Kamera-Zugriff verweigert: ", err);

});

}

export function takePhoto() {

let canvas = document.getElementById("canvas")

let video = document.getElementById("video")

canvas.width = video.videoWidth;

canvas.height = video.videoHeight;

canvas.getContext('2d').drawImage(video, 0, 0, canvas.width, canvas.height);

canvas.style.display = "block";

video.style.display = "none";

document.getElementById("takePhoto").style.display = "none";

document.getElementById("takePhotoAgain").style.display = "block";

document.getElementById("analyze").style.display = "block";

if (stream) {

stream.getTracks().forEach(track => track.stop());

}

let imageUrl = canvas.toDataURL("image/png")

sessionStorage.setItem("picture", imageUrl)

}

window.capture = capture

window.takePhoto = takePhoto

Daily:

let BACKEND\_URL = ""

fetch('./config.json')

.then(res => res.json())

.then(config => {

BACKEND\_URL = config.BACKEND\_URL

});

export function saveMood() {

if(sessionStorage.getItem("mood") !== null){

document.getElementById("symptoms-section").style.display = "block"

document.getElementById("saveMood").style.display = "none"

document.getElementById("moodError").style.display = "none"

} else {

document.getElementById("moodError").style.display = "block"

}

}

export function saveSymptoms() {

if(sessionStorage.getItem("symptoms") !== null && sessionStorage.getItem("symptoms") !== "[]"){

document.getElementById("tracking-section").style.display = "block"

document.getElementById("saveSymptoms").style.display = "none"

document.getElementById("symptomsError").style.display = "none"

} else {

document.getElementById("symptomsError").style.display = "block"

}

}

export function savePeriod() {

if(sessionStorage.getItem("period") !== null){

document.getElementById("periodError").style.display = "none"

document.getElementById("sports-section").style.display = "block"

document.getElementById("savePeriod").style.display = "none"

} else {

document.getElementById("periodError").style.display = "block"

}

}

export async function saveSport() {

if(sessionStorage.getItem("sport\_type") !== null && sessionStorage.getItem("sport\_intensity") !== null && sessionStorage.getItem("sport\_duration") !== null){

let body = {

uid: localStorage.getItem("uid"),

mood: sessionStorage.getItem("mood"),

symptoms: JSON.parse(sessionStorage.getItem("symptoms")),

period: sessionStorage.getItem("period"),

sport: {

type: sessionStorage.getItem("sport\_type"),

intensity: sessionStorage.getItem("sport\_intensity"),

duration: sessionStorage.getItem("sport\_duration"),

comment: sessionStorage.getItem("sport\_comment")

},

date: Date.now()

}

await axios.post(BACKEND\_URL + "/daily", body, {

headers: {

'Content-Type': 'application/json',

'Authorization': `Bearer ${localStorage.getItem('token')}`

}

}).then(resp => {

if(resp.status === 200){

document.getElementById("successMessage").style.display = "block"

document.getElementById("mood-section").style.display = "none"

document.getElementById("symptoms-section").style.display = "none"

document.getElementById("tracking-section").style.display = "none"

document.getElementById("sports-section").style.display = "none"

sessionStorage.setItem("uploadedDaily", "true")

}

})

}

}

// Stimmungs-Funktion

export function selectMood(element) {

document.getElementById("moodError").style.display = "none"

document.querySelectorAll('.mood-option').forEach(option => {

option.classList.remove('selected');

});

element.classList.add('selected');

const mood = element.getAttribute('data-mood');

const responseElement = document.getElementById('moodResponse');

if (!responseElement) return;

let response = '';

let responseColor = '';

switch(mood) {

case 'glücklich':

response = 'Wie schön, dass du heute glücklich bist! 🌟';

responseColor = '#4CAF50';

sessionStorage.setItem("mood", "glücklich")

break;

case 'entspannt':

response = 'Toll, dass du dich entspannt fühlst! 🌸';

responseColor = '#2196F3';

sessionStorage.setItem("mood", "entspannt")

break;

case 'gestresst':

response = 'Stress kann sehr belastend sein. Denk daran, kleine Pausen einzulegen! 🍃';

responseColor = '#FFC107';

sessionStorage.setItem("mood", "gestresst")

break;

case 'traurig':

response = 'Schade, dass du heute traurig bist. Morgen wird bestimmt ein besserer Tag! 💝';

responseColor = '#E91E63';

sessionStorage.setItem("mood", "traurig")

break;

}

responseElement.style.backgroundColor = responseColor + '20';

responseElement.style.color = responseColor;

responseElement.innerHTML = response;

responseElement.classList.add('show');

}

// Menstruations-Funktion

export function selectMenstruation(element) {

document.getElementById("periodError").style.display = "none"

document.querySelectorAll('.menstruation-card').forEach(card => {

card.classList.remove('selected');

});

element.classList.add('selected');

const intensity = element.getAttribute('data-value');

sessionStorage.setItem("period", intensity)

const warningElement = document.getElementById('menstruationWarning');

if (!warningElement) return;

if (intensity === 'stark') {

warningElement.innerHTML = `

<div>

<h4>⚠️ Wichtiger Hinweis zur starken Menstruation</h4>

<p>Starke Monatsblutungen können zu Eisenmangel führen. Wir empfehlen:</p>

<ul>

<li>Dokumentiere die Dauer der starken Blutungen</li>

<li>Achte auf zusätzliche Anämie-Symptome</li>

<li>Lass nach Ende der Menstruation deine Eisenwerte überprüfen</li>

<li>Sprich mit deinem Arzt über präventive Maßnahmen</li>

</ul>

<p><strong>Tipp:</strong> Eine eisenreiche Ernährung kann unterstützend wirken.</p>

</div>

`;

warningElement.style.display = 'block';

} else {

warningElement.style.display = 'none';

}

}

// Symptom-Check Funktion

export function checkSymptoms() {

document.getElementById("symptomsError").style.display = "none"

console.log("Symptom-Check wird ausgeführt");

const symptoms = [

'fatigue', 'dizziness', 'shortness\_of\_breath', 'headache',

'pale\_skin', 'weakness', 'cold\_hands', 'concentration'

];

let checkedSymptoms = symptoms.filter(symptom => {

const checkbox = document.getElementById(symptom);

return checkbox && checkbox.checked;

});

sessionStorage.setItem("symptoms", JSON.stringify(checkedSymptoms))

const warningElement = document.getElementById('symptomWarning');

if (!warningElement) {

console.error("symptomWarning Element nicht gefunden");

return;

}

if (checkedSymptoms.length >= 3) {

warningElement.innerHTML = `

<div class="warning-box warning-severe">

<h4>⚠️ Wichtiger Hinweis</h4>

<p>Du hast ${checkedSymptoms.length} Symptome angegeben, die auf eine Anämie hinweisen könnten.

Wir empfehlen dir, dies mit deinem Arzt zu besprechen und gegebenenfalls

deine Blutwerte überprüfen zu lassen.</p>

<p><strong>Tipp:</strong> Führe ein Symptom-Tagebuch und zeige es bei deinem nächsten Arztbesuch.</p>

</div>

`;

warningElement.style.display = 'block';

document.getElementById('symptomWarning').style.backgroundColor = "var(--danger-light)"

} else if (checkedSymptoms.length > 0) {

document.getElementById('symptomWarning').style.backgroundColor = "var(--warning-light)"

warningElement.innerHTML = `

<div class="warning-box warning-mild">

<h4>ℹ️ Information</h4>

<p>Die von dir angegebenen Symptome können verschiedene Ursachen haben.

Beobachte sie weiter und sprich mit deinem Arzt, wenn sie länger anhalten.</p>

</div>

`;

warningElement.style.display = 'block';

} else {

warningElement.style.display = 'none';

}

}

// Sport-Intensität Auswahl

export function selectIntensity(button) {

console.log("Intensität wird ausgewählt");

document.querySelectorAll('.intensity-btn').forEach(btn => {

btn.classList.remove('selected');

});

button.classList.add('selected');

updateSportWarning();

}

// Sport-Warning Update Funktion

export function updateSportWarning() {

console.log("Sport-Warning wird aktualisiert");

const selectedSport = document.querySelector('input[name="sport-type"]:checked');

const selectedIntensity = document.querySelector('.intensity-btn.selected');

const durationElement = document.getElementById('duration');

const duration = durationElement ? parseInt(durationElement.value) || 0 : 0;

const warningElement = document.getElementById('sportWarning');

if (!selectedSport || !selectedIntensity) {

warningElement.style.display = 'none';

return;

}

const sportType = selectedSport.id;

const intensity = selectedIntensity.getAttribute('data-intensity');

const comment = document.getElementById("activities").value

sessionStorage.setItem("sport\_type", sportType)

sessionStorage.setItem("sport\_intensity", intensity)

sessionStorage.setItem("sport\_duration", duration.toString())

sessionStorage.setItem("sport\_comment", comment)

let warningMessage = '';

if (sportType === 'endurance' && (intensity === 'intense' || (intensity === 'moderate' && duration > 60))) {

document.getElementById("sportWarning").style.borderLeft = "4px solid var(--warning-dark)"

document.getElementById("sportWarning").style.backgroundColor = "var(--warning-light)"

warningMessage = `

<div>

<h4>⚠️ Hinweis zu intensivem Ausdauertraining</h4>

<p>Intensives Ausdauertraining kann zu einer temporären Sportanämie führen:</p>

<ul>

<li>Erhöhte Zerstörung roter Blutkörperchen durch mechanische Belastung</li>

<li>Verdünnung des Blutes durch erhöhtes Plasmavolumen</li>

</ul>

<p><strong>Empfehlungen:</strong></p>

<ul>

<li>Achte auf ausreichende Regeneration</li>

<li>Beobachte Anämie-Symptome</li>

<li>Stelle eine ausgewogene Ernährung sicher</li>

</ul>

</div>

`;

} else if (intensity === 'moderate') {

document.getElementById("sportWarning").style.borderLeft = "4px solid var(--primary-green)"

document.getElementById("sportWarning").style.backgroundColor = "var(--primary-light)"

warningMessage = `

<div>

<h4>ℹ️ Information zu moderater Bewegung</h4>

<p>Moderate Bewegung kann positive Effekte auf deine Blutwerte haben:</p>

<ul>

<li>Verbesserte Durchblutung</li>

<li>Anregung der Blutbildung</li>

<li>Stärkung des Immunsystems</li>

</ul>

</div>

`;

}

if (warningMessage) {

warningElement.innerHTML = warningMessage;

warningElement.style.display = 'block';

} else {

warningElement.style.display = 'none';

}

}

window.saveMood = saveMood

window.saveSymptoms = saveSymptoms

window.savePeriod = savePeriod

window.saveSport = saveSport

window.selectMood = selectMood;

window.selectMenstruation = selectMenstruation;

window.checkSymptoms = checkSymptoms;

window.selectIntensity = selectIntensity;

window.updateSportWarning = updateSportWarning;

funkitionen:

import {checkSymptoms, updateSportWarning} from "./daily.js";

let BACKEND\_URL = ""

fetch('./config.json')

.then(res => res.json())

.then(config => {

BACKEND\_URL = config.BACKEND\_URL

});

let historyChart = null;

function stampToDate(timestamp){

const date = new Date(timestamp)

const day = String(date.getDate()).padStart(2, '0')

const month = String(date.getMonth() + 1).padStart(2, '0')

const year = date.getFullYear()

return `${day}.${month}.${year}`

}

// Initialize history chart when the DOM is loaded

export function initializeHistoryChart() {

const chartCanvas = document.getElementById('historyChart');

if (chartCanvas) {

historyChart = new Chart(chartCanvas, {

type: 'line',

data: {

labels: [],

datasets: [{

label: 'Hämoglobinwert Verlauf',

data: [],

borderColor: '#2196F3',

tension: 0.1

}]

},

options: {

responsive: true,

scales: {

y: {

beginAtZero: false,

min: 100,

max: 180

}

}

}

});

}

}

export async function updateHistoryChart() {

if (historyChart) {

await axios.get(BACKEND\_URL + "/hb?uid=" + localStorage.getItem("uid"), {

headers: {

'Authorization': `Bearer ${localStorage.getItem('token')}`,

"Content-Type": "multipart/form-data"

}

}).then(res => {

historyChart.data.labels = res.data.map(d => d.date);

historyChart.data.datasets[0].data = res.data.map(d => d.value);

historyChart.update()

})

axios.get(BACKEND\_URL + "/last-hb?uid=" + localStorage.getItem("uid"), {

headers: {

'Authorization': `Bearer ${localStorage.getItem('token')}`,

"Content-Type": "multipart/form-data"

}

}).then(res => {

console.log("Resp", res.data);

let difference = "";

let oldValue = parseFloat(sessionStorage.getItem("hb"));

let newValue = parseFloat(res.data.value);

if (!isNaN(oldValue) && !isNaN(newValue)) {

let diff = Math.floor(oldValue - newValue);

difference = (diff > 0 ? "+" : "") + diff;

}

sessionStorage.setItem("lasthb\_date", res.data.date);

sessionStorage.setItem("lasthb\_difference", difference);

sessionStorage.setItem("hb", newValue); // Wichtig: neuen Wert speichern

document.getElementById("lastTest").textContent = res.data.date;

document.getElementById("change").textContent = difference;

})

}

}

// convert images/files into "blob" to be readable for the backend

export function convertDataUrlToBlob(dataUrl){

let arr = dataUrl.split(',');

let mime = arr[0].match(/:(.\*?);/)[1];

let bstr = atob(arr[1]);

let n = bstr.length;

let u8arr = new Uint8Array(n);

while(n--) {

u8arr[n] = bstr.charCodeAt(n);

}

return new Blob([u8arr], { type: mime });

}

export function detectNails() {

// Convert picture to blob

let imageBlob = convertDataUrlToBlob(sessionStorage.getItem("picture"));

let formData = new FormData()

formData.append("file", imageBlob, "image.png")

// Send picture to Backend

axios.post(BACKEND\_URL + "/nails/detect", formData, {

headers: {

'Authorization': `Bearer ${localStorage.getItem('token')}`,

"Content-Type": "multipart/form-data"

}

}).then(res => {

// Process backend response and do something with the detection boxes (?)

if(res.data.bboxes.length > 0){

let canvas = document.getElementById("canvas")

const ctx = canvas.getContext('2d')

const img = new Image();

img.onload = function () {

canvas.width = img.width;

canvas.height = img.height;

ctx.drawImage(img, 0, 0);

ctx.strokeStyle = "red";

ctx.lineWidth = 2;

ctx.font = "16px Arial";

ctx.textBaseline = "top";

res.data.bboxes.forEach((box, index) => {

const width = box.x2 - box.x1;

const height = box.y2 - box.y1;

ctx.strokeRect(box.x1, box.y1, width, height);

ctx.fillStyle = "rgba(255, 0, 0, 0.7)";

const text = (index + 1).toString();

const textWidth = ctx.measureText(text).width;

const padding = 4;

ctx.fillRect(box.x1, box.y1, textWidth + padding \* 2, 20);

ctx.fillStyle = "white";

ctx.fillText(text, box.x1 + padding, box.y1 + 2);

});

};

// Show select and confirm boxes for nail detection boxes -> not functional somehow

img.src = res.data.annotated;

document.getElementById("selectBoxes").style.display = "block"

document.getElementById("confirmBoxes").style.display = "block"

} else {

// Show error message if no nails were found

document.getElementById("noFingernailsDetected").style.display = "block"

}

// hide "Analysieren" button

document.getElementById("analyze").style.display = "none"

})

}

// --- Not functional ---

export function changeBoxes(){

}

// Get the hb prediction from the backend based on the detected nails

export function predictHb(){

// show results and hide edit-buttons

document.getElementById("results").style.display = "block"

document.getElementById("confirmBoxes").style.display = "none"

document.getElementById("selectBoxes").style.display = "none"

// send analyzed picture to backend

let imageBlob = convertDataUrlToBlob(sessionStorage.getItem("picture"));

let formData = new FormData()

const date = stampToDate(Date.now())

formData.append("file", imageBlob, "image.png")

formData.append("uid", localStorage.getItem("uid"))

formData.append("date", date)

axios.post(BACKEND\_URL + "/hb/predict", formData, {

headers: {

'Authorization': `Bearer ${localStorage.getItem('token')}`,

"Content-Type": "multipart/form-data"

}

}).then(res => {

// save hb value locally (session) and updateHbValue() to show it to the user

sessionStorage.setItem("hb", res.data.hb)

updateHbValue()

updateHistoryChart()

})

}

export function updateHbValue() {

const hemoglobinValue = document.getElementById("hemoglobinValue")

const hemoglobinBar = document.getElementById('hemoglobinBar');

const recommendationBox = document.getElementById("recommendation")

let value = sessionStorage.getItem("hb")

if (hemoglobinBar) {

// calculate status of the hb bar

const percentage = ((value - 100) / (180 - 100)) \* 100;

hemoglobinBar.style.width = `${percentage}%`;

}

hemoglobinValue.innerText = value + " g/L"

let color;

let recommendation;

// showing recommendations and matching colors depending on the hb value

if (value < 120) {

color = "var(--danger-dark)";

hemoglobinBar.style.background = "linear-gradient(90deg, var(--danger-light), var(--danger-dark))"

recommendation = 'Niedriger Hämoglobinwert<br />Ärztliche Untersuchung empfohlen';

recommendationBox.style.background = "var(--danger-light)"

recommendationBox.style.borderLeft = "4px solid var(--danger-dark)"

} else if (value > 160) {

color = "var(--danger-dark)";

hemoglobinBar.style.background = "linear-gradient(90deg, var(--danger-light), var(--danger-dark))"

recommendation = 'Erhöhter Hämoglobinwert<br />Ärztliche Kontrolle empfohlen';

recommendationBox.style.background = "var(--danger-light)"

recommendationBox.style.borderLeft = "4px solid var(--danger-dark)"

} else {

color = "var(--primary-dark)";

recommendation = 'Normaler Hämoglobinwert<br />Keine Maßnahmen erforderlich';

}

hemoglobinValue.style.color = color;

const recommendationElement = document.getElementById('recommendation');

if (recommendationElement) {

recommendationElement.innerHTML = `

<p>${recommendation}</p>

`;

}

}

// --- NOT FUNCTIONAL ---

export function exportPDF() {

alert('PDF-Export-Funktion wird implementiert');

}

// --- NOT FUNCTIONAL / NOT TESTED ---

export function shareResults() {

if (navigator.share) {

navigator.share({

title: 'Meine Anämie-Testergebnisse',

text: `Hämoglobinwert: ${document.getElementById('hemoglobinValue').textContent}`

}).catch(console.error);

} else {

alert('Teilen-Funktion nicht verfügbar');

}

}

// Globaler Event-Listener für Klicks außerhalb der Modals

window.onclick = function(event) {

if (event.target.classList.contains('modal')) {

event.target.style.display = 'none';

}

};

// Initialize everything on DOM content loaded

document.addEventListener('DOMContentLoaded', function() {

// Initialize Chart if it exists

initializeHistoryChart();

// Set up symptom checkboxes

document.querySelectorAll('.symptom-checkbox').forEach(checkbox => {

checkbox.addEventListener('change', checkSymptoms);

});

// Set up sport type radio buttons

document.querySelectorAll('input[name="sport-type"]').forEach(radio => {

radio.addEventListener('change', updateSportWarning);

});

// Set up duration input

const durationInput = document.getElementById('duration');

if (durationInput) {

durationInput.addEventListener('input', updateSportWarning);

}

const commentInput = document.getElementById('activities')

if(commentInput) {

commentInput.addEventListener('input', updateSportWarning)

}

// Set up intensity buttons

document.querySelectorAll('.intensity-btn').forEach(btn => {

btn.addEventListener('click', function() {

selectIntensity(this);

});

});

});

window.detectNails = detectNails

window.changeBoxes = changeBoxes

window.predictHb = predictHb

window.updateHbValue = updateHbValue

window.exportPDF = exportPDF

window.shareResults = shareResults

window.initializeHistoryChart = initializeHistoryChart

window.updateHistoryChart = updateHistoryChart

Profil:

let BACKEND\_URL = ""

fetch('./config.json')

.then(res => res.json())

.then(config => {

BACKEND\_URL = config.BACKEND\_URL

});

// Array mit Anämie-relevanten Erkrankungen

const anemiaRelatedDiseases = [

"systemischer lupus erythematodes",

"rheumatoide arthritis",

"rheuma",

"autoimmunhämolytische anämie",

"leukämie",

"lymphome",

"aplastische anämie",

"myelodysplastisches syndrom",

"sichelzellanämie",

"thalassämie",

"sphärozytose",

"niereninsuffizienz",

"lebererkrankungen",

"chronische entzündungen",

"hiv",

"aids",

"vitamin b12-mangel",

"perniziöse anämie",

"folsäuremangel",

"zöliakie",

"morbus crohn",

"colitis ulcerosa",

"magengeschwüre"

];

// Array mit Anämie-relevanten Medikamenten

const anemiaRelatedMedications = [

"ibuprofen",

"diclofenac",

"aspirin",

"penicillin",

"cephalosporin",

"sulfonamid",

"platinverbindung",

"methotrexat",

"cyclophosphamid",

"azathioprin",

"mycophenolat",

"phenytoin",

"carbamazepin",

"protonenpumpenhemmer",

"ace-hemmer",

"thiazid",

"colchicin",

"methyldopa"

];

export function prefillData(){

if(sessionStorage.getItem("picture") !== null){

if(sessionStorage.getItem("hb") !== null){

let canvas = document.getElementById("canvas")

let ctx = canvas.getContext("2d");

let img = new Image()

img.onload = function () {

canvas.width = img.width

canvas.height = img.height

ctx.drawImage(img, 0, 0, canvas.width, canvas.height);

};

img.src = sessionStorage.getItem("picture")

canvas.style.display = "block"

document.getElementById("takePhoto").style.display = "none"

document.getElementById("takePhotoAgain").style.display = "block"

document.getElementById("video").style.display = "none"

document.getElementById("confirmBoxes").style.display = "none"

document.getElementById("selectBoxes").style.display = "none"

document.getElementById("openCam").style.display = "none"

document.getElementById("results").style.display = "block"

updateHbValue()

} else {

document.getElementById("canvas").style.display = "block"

document.getElementById("video").style.display = "none"

document.getElementById("takePhoto").style.display = "none"

document.getElementById("takePhotoAgain").style.display = "block"

document.getElementById("analyze").style.display = "block"

}

}

}

// Funktion zum Laden der gespeicherten Daten

export function loadProfilData() {

const nameElement = document.getElementById('name');

if (nameElement && sessionStorage.getItem("name") !== "undefined"){

nameElement.value = sessionStorage.getItem("name");

} else {

nameElement.value = ''

}

const geburtsdatumElement = document.getElementById('geburtsdatum');

if (geburtsdatumElement && sessionStorage.getItem("birthday") !== "undefined"){

geburtsdatumElement.value = sessionStorage.getItem("birthday");

} else {

geburtsdatumElement.value = ''

}

const geschlechtElement = document.getElementById('geschlecht');

if (geschlechtElement && sessionStorage.getItem("gender") !== "undefined"){

geschlechtElement.value = sessionStorage.getItem("gender");

} else {

geschlechtElement.value = ''

}

if (sessionStorage.getItem("vorerkrankung")) {

document.getElementById("vorerkrankungenListe").innerHTML = ''

const vorerkrankungen = JSON.parse(sessionStorage.getItem("vorerkrankung"))

vorerkrankungen.forEach(item => {

addItemToList('vorerkrankungenListe', item);

});

}

if (sessionStorage.getItem("medikamente")) {

document.getElementById("medikamenteListe").innerHTML = ''

const medikamente = JSON.parse(sessionStorage.getItem("medikamente"))

medikamente.forEach(item => {

addItemToList('medikamenteListe', item);

});

}

}

// Einheitliche Funktion zum Hinzufügen von Items zu einer Liste

export function addItemToList(listId, text) {

const list = document.getElementById(listId);

if (!list) return;

const div = document.createElement('div');

div.className = 'list-tag';

div.id = encodeURIComponent(text)

div.innerHTML = `

<span class="tag-text">${text}</span>

<button type="button" class="delete-btn" onclick="deleteItemFromList('${listId}', '${text}')">×</button>

`;

list.appendChild(div);

if(listId === "vorerkrankungenListe"){

if(sessionStorage.getItem("vorerkrankung") !== "undefined"){

let vorerkrankungen = JSON.parse(sessionStorage.getItem("vorerkrankung"))

vorerkrankungen.push(text)

sessionStorage.setItem("vorerkrankung", JSON.stringify(vorerkrankungen))

} else {

sessionStorage.setItem("vorerkrankung", "[]")

}

} else if(listId === "medikamenteListe"){

if(sessionStorage.getItem("medikamente") !== "undefined") {

let meds = JSON.parse(sessionStorage.getItem("medikamente"))

meds.push(text)

sessionStorage.setItem("medikamente", JSON.stringify(meds))

} else {

sessionStorage.setItem("medikamente", "[]")

}

}

}

export function deleteItemFromList(listId, text){

document.getElementById(encodeURIComponent(text)).remove()

if(listId === "vorerkrankungenListe"){

let krankliste = JSON.parse(sessionStorage.getItem("vorerkrankung"))

let neueListe = krankliste.filter(item => item !== text)

sessionStorage.setItem("vorerkrankung", JSON.stringify(neueListe))

} else if(listId === "medikamenteListe"){

let medListe = JSON.parse(sessionStorage.getItem("medikamente"))

let neueListe = medListe.filter(item => item !== text)

sessionStorage.setItem("medikamente", JSON.stringify(neueListe))

}

}

// Generische Funktion zum Anzeigen von Modals

export function showWarningModal(type) {

const modalId = `${type}WarningModal`;

let modal = document.getElementById(modalId);

if (!modal) {

modal = document.createElement('div');

modal.id = modalId;

modal.className = 'modal';

const content = type === 'anemia' ?

`<div class="modal-content">

<span class="close-button">&times;</span>

<h2>⚠️ Wichtiger Hinweis zur Anämie</h2>

<p>Die von Ihnen eingetragene Erkrankung steht in direktem Zusammenhang mit Anämie (Blutarmut) und kann:</p>

<ul>

<li>eine bestehende Anämie verstärken.</li>

<li>eine Anämie auslösen.</li>

<li>die Behandlung einer Anämie erschweren.</li>

</ul>

<p>Bitte besprechen Sie dies mit Ihrem behandelnden Arzt und lassen Sie Ihre Blutwerte regelmäßig kontrollieren.</p>

</div>` :

`<div class="modal-content">

<span class="close-button">&times;</span>

<h2>⚠️ Wichtiger Hinweis zu Ihrem Medikament</h2>

<div class="warning-content">

<p>Das von Ihnen eingetragene Medikament kann Auswirkungen auf Ihre Blutwerte haben:</p>

<ul>

<li>Es kann eine bestehende Anämie verstärken</li>

<li>Es kann die Bildung roter Blutkörperchen beeinträchtigen</li>

<li>Es kann die Eisenaufnahme oder -verwertung stören</li>

<li>Es kann die Behandlung einer Anämie erschweren</li>

</ul>

<div class="important-notice">

<p><strong>Wichtige Hinweise:</strong></p>

<ul>

<li>Setzen Sie das Medikament NICHT eigenständig ab</li>

<li>Besprechen Sie mögliche Wechselwirkungen mit Ihrem Arzt</li>

<li>Lassen Sie Ihre Blutwerte regelmäßig kontrollieren</li>

<li>Achten Sie auf Symptome wie verstärkte Müdigkeit oder Schwäche</li>

</ul>

</div>

</div>

</div>`;

modal.innerHTML = content;

document.body.appendChild(modal);

// Event-Listener für das Schließen

const closeButton = modal.querySelector('.close-button');

closeButton.onclick = () => modal.style.display = 'none';

}

modal.style.display = 'block';

}

// Funktion zum Hinzufügen von Vorerkrankungen

export function addVorerkrankung() {

const input = document.getElementById('neueVorerkrankung');

if (!input) return;

const krankheit = input.value

if (krankheit) {

addItemToList('vorerkrankungenListe', input.value);

if (anemiaRelatedDiseases.some(disease => krankheit.includes(disease))) {

showWarningModal('anemia');

}

input.value = '';

}

}

// Funktion zum Hinzufügen von Medikamenten

export function addMedikament() {

const input = document.getElementById('neuesMedikament');

if (!input) return;

const medikament = input.value.trim().toLowerCase();

if (medikament) {

addItemToList('medikamenteListe', input.value.trim());

if (anemiaRelatedMedications.some(med => medikament.includes(med))) {

showWarningModal('medication');

}

input.value = '';

}

}

// Hilfsfunktionen

export function collectSymptoms() {

return {

fatigue: document.getElementById('fatigue')?.checked || false,

dizziness: document.getElementById('dizziness')?.checked || false,

shortness\_of\_breath: document.getElementById('shortness\_of\_breath')?.checked || false,

headache: document.getElementById('headache')?.checked || false,

pale\_skin: document.getElementById('pale\_skin')?.checked || false,

weakness: document.getElementById('weakness')?.checked || false,

cold\_hands: document.getElementById('cold\_hands')?.checked || false,

concentration: document.getElementById('concentration')?.checked || false

};

}

export function resetForm() {

document.querySelectorAll('.mood-option, .menstruation-card').forEach(option => {

option.classList.remove('selected');

});

document.querySelectorAll('.symptom-checkbox').forEach(checkbox => {

checkbox.checked = false;

});

document.querySelectorAll('.intensity-btn').forEach(btn => {

btn.classList.remove('selected');

});

const activitiesField = document.getElementById('activities');

if (activitiesField) activitiesField.value = '';

['symptomWarning', 'sportWarning', 'menstruationWarning', 'moodResponse'].forEach(id => {

const element = document.getElementById(id);

if (element) element.style.display = 'none';

});

}

window.prefillData = prefillData;

window.loadProfilData = loadProfilData;

window.addItemToList = addItemToList;

window.deleteItemFromList = deleteItemFromList;

window.showWarningModal = showWarningModal;

window.addVorerkrankung = addVorerkrankung;

window.addMedikament = addMedikament;

window.collectSymptoms = collectSymptoms;

window.resetForm = resetForm;

User:

let BACKEND\_URL = ""

await fetch('./config.json')

.then(res => res.json())

.then(config => {

BACKEND\_URL = config.BACKEND\_URL

});

// call backend to receive userdata such as name, birthday etc. and write into sessionStorage

export async function getUserdata(){

let uid = localStorage.getItem("uid")

if(uid !== null || uid !== ""){

await axios.get(BACKEND\_URL + "/userdata?uid=" + uid, {

headers: {

'Authorization': `Bearer ${localStorage.getItem('token')}`

}

}).then(response => {

const resData = response.data

if(resData.name !== "" || resData.name !== null){

sessionStorage.setItem("name", resData.name)

}

if(resData.birthday !== "" || resData.birthday !== null){

sessionStorage.setItem("birthday", resData.birthday)

}

if(resData.gender !== "" || resData.gender !== null){

sessionStorage.setItem("gender", resData.gender)

}

if(resData.vorerkrankung !== "" || resData.vorerkrankung !== null){

sessionStorage.setItem("vorerkrankung", resData.vorerkrankung)

}

if(resData.medikamente !== "" || resData.medikamente !== null){

sessionStorage.setItem("medikamente", resData.medikamente)

}

})

}

}

// send new userdata to backend to save it in the db

export function saveUserdata(){

sessionStorage.setItem("name", document.getElementById("name").value)

sessionStorage.setItem("birthday", document.getElementById("geburtsdatum").value)

sessionStorage.setItem("gender", document.getElementById("geschlecht").value)

let body = {

uid: localStorage.getItem("uid"),

name: sessionStorage.getItem("name"),

birthday: sessionStorage.getItem("birthday"),

gender: sessionStorage.getItem("gender"),

vorerkrankung: sessionStorage.getItem("vorerkrankung"),

medikamente: sessionStorage.getItem("medikamente")

}

axios.post(BACKEND\_URL + "/userdata", body, {

headers: {

'Content-Type': 'application/json',

'Authorization': `Bearer ${localStorage.getItem('token')}`

}

})

}

// clear the sessionStorage (so the user gets redirected to login everytime trying to view a page) and redirect to login page

export function logout() {

localStorage.clear()

sessionStorage.clear()

window.location.href = "login.html"

}

// send username, password to backend, check if passwords are matching and check if user already exists

export async function register() {

let username = document.getElementById("username").value

let password = document.getElementById("password").value

let password2 = document.getElementById("password2").value

document.getElementById("errormessage").style.display = "none"

if(username !== ("" || null) && password !== ("" || null)){

if(password === password2){

let body = {

username: username,

password: password

}

try {

const response = await axios.post(BACKEND\_URL + "/register", body, {

headers: {

'Content-Type': 'application/json'

}

}).then(response => {

if(response.status === 200){

console.log("Response", response.data)

localStorage.setItem("loggedIn", "true");

localStorage.setItem("username", username);

localStorage.setItem("password", password);

localStorage.setItem("uid", response.data.id)

console.log("Erfolgreich registriert und eingeloggt");

window.location.href = "index.html";

} else {

document.getElementById("errormessage").style.display = "block"

document.getElementById("errormessage").innerText = "Registrierung fehlgeschlagen."

}

})

} catch (error) {

document.getElementById("errormessage").style.display = "block"

document.getElementById("errormessage").innerText = "Registrierung fehlgeschlagen."

console.error("Login fehlgeschlagen:", error);

}

} else {

document.getElementById("errormessage").style.display = "block"

document.getElementById("errormessage").innerText = "Die Passwörter stimmen\nnicht überein."

}

} else {

document.getElementById("errormessage").style.display = "block"

document.getElementById("errormessage").innerText = "Bitte Passwort und\nBenutzernamen angeben."

}

}

// Verifying if the user credentials in localStorage are valid and match the ones in the db - else redirect to login page

export async function verify(param) {

let username = localStorage.getItem("username")

let password = localStorage.getItem("password")

let body = {

username: username,

password: password

}

const forwarding = (param === "standard")

if(localStorage.getItem("username") !== null && localStorage.getItem("password") !== null){

try {

console.log("starting call to: " + BACKEND\_URL + "/login")

const response = await axios.post(BACKEND\_URL + "/login", body, {

headers: {

'Authorization': `Bearer ${localStorage.getItem('token')}`,

'Content-Type': 'application/json'

}

})

if(response.status === 200){

console.log("Logindaten erfolgreich verifiziert");

localStorage.setItem("loggedIn", "true");

localStorage.setItem("uid", response.data.id)

return true

} else {

if(forwarding) window.location.href = "login.html";

return false

}

} catch (error) {

if(forwarding) window.location.href = "login.html";

console.error("Login fehlgeschlagen:", error);

return false

}

} else {

if(forwarding) window.location.href = "login.html"

return false

}

}

// send login data to backend to check, if successful write credentials into localStorage

export async function login() {

let username, password;

document.getElementById("loginfailure").style.display = "none"

if(document.getElementById("username").value && document.getElementById("password").value) {

username = document.getElementById("username").value;

password = document.getElementById("password").value;

} else {

username = localStorage.getItem("username");

password = localStorage.getItem("password");

if (!username || !password) {

console.error("Keine Anmeldedaten verfügbar");

return;

}

}

const body = {

username: username,

password: password

};

try {

const response = await axios.post(BACKEND\_URL + "/login", body, {

headers: {

'Content-Type': 'application/json'

}

});

if(response.status === 200){

localStorage.setItem("username", username);

localStorage.setItem("password", password);

localStorage.setItem("uid", response.data.id)

localStorage.setItem("token", response.data.token)

console.log("Erfolgreich eingeloggt");

window.location.href = "index.html";

} else {

document.getElementById("loginfailure").style.display = "block"

}

} catch (error) {

document.getElementById("loginfailure").style.display = "block"

console.error("Login fehlgeschlagen:", error);

}

}

window.getUserdata = getUserdata;

window.saveUserdata = saveUserdata;

window.logout = logout;

window.register = register;

window.verify = verify;

window.login = login;

HTML Skripte:

Anämie Page

<!DOCTYPE html>

<html lang="de">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Anämie-Früherkennung | Anämie</title>

<link rel="stylesheet" href="./style.css">

<script type="module" src="./js/functions.js"></script>

<script type="module" src="./js/user.js"></script>

<script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>

<script>

//verify("standard")

</script>

</head>

<body>

<div class="navbar">

<a href="./index.html">

<span class="icon">🔬</span>

<div>Analyse</div>

</a>

<a href="./daily.html">

<span class="icon">📅</span>

<div>Daily Stuff</div>

</a>

<a href="./profil.html">

<span class="icon">👤</span>

<div>Profil</div>

</a>

<a href="#" style="background-color: #575757">

<span class="icon">💉</span>

<div>Anämie</div>

</a>

<a href="./ueber-uns.html">

<span class="icon">ℹ️</span>

<div>Über uns</div>

</a>

</div>

<div class="main-content">

<div id="anaemie" class="container">

<h1 class="title">Anämie (Blutarmut)</h1>

<div class="content">

<p class="description">

Anämie, auch als Blutarmut bekannt, ist ein Zustand, bei dem der Körper nicht genügend gesunde rote Blutkörperchen oder zu wenig Hämoglobin hat. Das ist problematisch, weil dadurch nicht ausreichend Sauerstoff im Körper transportiert werden kann.

</p>

<div class="symptoms">

<h2>Häufige Symptome:</h2>

<ul>

<li>Müdigkeit und Erschöpfung</li>

<li>Blässe</li>

<li>Schwäche</li>

<li>Kurzatmigkeit</li>

<li>Schwindel</li>

</ul>

</div>

<div class="risk-groups">

<h2>Besonders gefährdete Gruppen:</h2>

<ul>

<li>Frauen mit starker Menstruation</li>

<li>Schwangere</li>

<li>Vegetarier/Veganer</li>

<li>Ältere Menschen</li>

<li>Chronisch Kranke</li>

</ul>

</div>

<div class="treatment">

<h2>Behandlung:</h2>

<p>

Die gute Nachricht ist: Anämie ist in den meisten Fällen gut behandelbar. Die Therapie richtet sich nach der Grundursache und kann zum Beispiel durch Eisenpräparate, Vitamine oder eine Ernährungsumstellung erfolgen.

</p>

</div>

</div>

</div>

</div>

</body>

Daily Page:

<!DOCTYPE html>

<html lang="de">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Anämie-Früherkennung | Daily Stuff</title>

<link rel="stylesheet" href="./style.css">

<script type="module" src="./js/functions.js"></script>

<script type="module" src="./js/user.js"></script>

<script type="module" src="./js/daily.js"></script>

<script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>

<script type="module">

import {verify} from "./js/user.js";

verify("standard")

if(sessionStorage.getItem("uploadedDaily") === "true"){

document.getElementById("successMessage").style.display = "block"

document.getElementById("mood-section").style.display = "none"

document.getElementById("symptoms-section").style.display = "none"

document.getElementById("tracking-section").style.display = "none"

document.getElementById("sports-section").style.display = "none"

} else {

let mood = sessionStorage.getItem("mood")

if (mood !== null) {

switch (mood) {

case "glücklich":

document.getElementById("glücklich").classList.add("selected")

break

case "entspannt":

document.getElementById("entspannt").classList.add("selected")

break

case "gestresst":

document.getElementById("gestresst").classList.add("selected")

break

case "traurig":

document.getElementById("traurig").classList.add("selected")

break

}

}

let symptoms = sessionStorage.getItem("symptoms")

if (symptoms !== null && symptoms !== "[]") {

let sympArray = JSON.parse(symptoms)

for (let i = 0; i < sympArray.length; i++) {

switch (sympArray[i]) {

case "fatigue":

document.getElementById("fatigue").checked = true

break

case "dizziness":

document.getElementById("dizziness").checked = true

break

case "shortness\_of\_breath":

document.getElementById("shortness\_of\_breath").checked = true

break

case "headache":

document.getElementById("headache").checked = true

break

case "pale\_skin":

document.getElementById("pale\_skin").checked = true

break

case "weakness":

document.getElementById("weakness").checked = true

break

case "cold\_hands":

document.getElementById("cold\_hands").checked = true

break

case "concentration":

document.getElementById("concentration").checked = true

break

}

}

}

let period = sessionStorage.getItem("period")

if (period !== null) {

switch (period) {

case "nicht":

document.getElementById("period-nicht").classList.add("selected")

break

case "leicht":

document.getElementById("period-leicht").classList.add("selected")

break

case "mittel":

document.getElementById("period-mittel").classList.add("selected")

break

case "stark":

document.getElementById("period-stark").classList.add("selected")

break

}

}

let sport\_type = sessionStorage.getItem("sport\_type")

if (sport\_type !== null) {

switch (sport\_type) {

case "endurance":

document.getElementById("endurance").checked = true

break

case "strength":

document.getElementById("strength").checked = true

break

case "flexibility":

document.getElementById("flexibility").checked = true

break

}

}

let sport\_intensity = sessionStorage.getItem("sport\_intensity")

if (sport\_intensity != null) {

switch (sport\_intensity) {

case "light":

document.getElementById("intensity\_light").classList.add("selected")

break

case "moderate":

document.getElementById("intensity\_moderate").classList.add("selected")

break

case "intense":

document.getElementById("intensity\_intense").classList.add("selected")

break

}

}

let sport\_duration = sessionStorage.getItem("sport\_duration")

if (sport\_duration != null) {

document.getElementById("duration").value = sport\_duration.valueOf()

}

let sport\_comment = sessionStorage.getItem("sport\_comment")

if (sport\_comment != null) {

document.getElementById("activities").value = sport\_comment

}

}

</script>

</head>

<body>

<div class="navbar">

<a href="./index.html">

<span class="icon">🔬</span>

<div>Analyse</div>

</a>

<a href="#" style="background-color: #575757">

<span class="icon">📅</span>

<div>Daily Stuff</div>

</a>

<a href="./profil.html">

<span class="icon">👤</span>

<div>Profil</div>

</a>

<a href="./anaemie.html">

<span class="icon">💉</span>

<div>Anämie</div>

</a>

<a href="./ueber-uns.html">

<span class="icon">ℹ️</span>

<div>Über uns</div>

</a>

</div>

<!-- Inhalt für den "Daily Stuff"-Tab -->

<div class="main-content">

<div id="daily" class="container">

<div class="daily-header">

<h1>Tägliches Befinden</h1>

<span id="successMessage" style="display: none">Vielen Dank, dass du dein heutiges Befinden hinterlegt hast!</span>

</div>

<div class="mood-section" id="mood-section">

<h2 class="mood-question">Wie ist deine Stimmung heute?</h2>

<div class="mood-selector">

<div class="mood-option" id="glücklich" data-mood="glücklich" onclick="selectMood(this)">

<span class="emoji">😄</span>

<span class="mood-label">Glücklich</span>

</div>

<div class="mood-option"id="entspannt" data-mood="entspannt" onclick="selectMood(this)">

<span class="emoji">😌</span>

<span class="mood-label">Entspannt</span>

</div>

<div class="mood-option" id="gestresst" data-mood="gestresst" onclick="selectMood(this)">

<span class="emoji">😣</span>

<span class="mood-label">Gestresst</span>

</div>

<div class="mood-option" id="traurig" data-mood="traurig" onclick="selectMood(this)">

<span class="emoji">😢</span>

<span class="mood-label">Traurig</span>

</div>

</div>

<div id="moodResponse" class="mood-response"></div>

<span id="moodError" style="display: none" class="errormessage">Bitte wähle eine Stimmung aus</span>

<div class="button" id="saveMood">

Weiter

</div>

</div>

<div id="symptoms-section" style="display: none">

<h2>Symptom-Check</h2>

<p class="symptoms-intro">Hast du heute eines dieser Symptome bemerkt?</p>

<div class="symptoms-grid">

<div class="symptom-card">

<input type="checkbox" id="fatigue" class="symptom-checkbox">

<label for="fatigue">

<span class="symptom-icon">😴</span>

<span class="symptom-name">Müdigkeit</span>

</label>

</div>

<div class="symptom-card">

<input type="checkbox" id="dizziness" class="symptom-checkbox">

<label for="dizziness">

<span class="symptom-icon">💫</span>

<span class="symptom-name">Schwindel</span>

</label>

</div>

<div class="symptom-card">

<input type="checkbox" id="shortness\_of\_breath" class="symptom-checkbox">

<label for="shortness\_of\_breath">

<span class="symptom-icon">🫁</span>

<span class="symptom-name">Kurzatmigkeit</span>

</label>

</div>

<div class="symptom-card">

<input type="checkbox" id="headache" class="symptom-checkbox">

<label for="headache">

<span class="symptom-icon">🤕</span>

<span class="symptom-name">Kopfschmerzen</span>

</label>

</div>

<div class="symptom-card">

<input type="checkbox" id="pale\_skin" class="symptom-checkbox">

<label for="pale\_skin">

<span class="symptom-icon">👤</span>

<span class="symptom-name">Blasse Haut</span>

</label>

</div>

<div class="symptom-card">

<input type="checkbox" id="weakness" class="symptom-checkbox">

<label for="weakness">

<span class="symptom-icon">💪</span>

<span class="symptom-name">Schwäche</span>

</label>

</div>

<div class="symptom-card">

<input type="checkbox" id="cold\_hands" class="symptom-checkbox">

<label for="cold\_hands">

<span class="symptom-icon">🧊</span>

<span class="symptom-name">Kalte Hände/Füße</span>

</label>

</div>

<div class="symptom-card">

<input type="checkbox" id="concentration" class="symptom-checkbox">

<label for="concentration">

<span class="symptom-icon">🤔</span>

<span class="symptom-name">Konzentrationsprobleme</span>

</label>

</div>

</div>

<div id="symptomWarning" class="symptom-warning" style="display: none;">

<!-- Hier erscheint die Warnung -->

</div>

<div style="text-align: center">

<span id="symptomsError" style="display: none" class="errormessage">Bitte wähle mindestens ein Symptom aus</span>

</div>

<div class="button" style="text-align: center" id="saveSymptoms">

Weiter

</div>

</div>

<div id="tracking-section" style="display: none">

<h2>Menstruations-Tracking</h2>

<p class="tracking-intro">Wie stark ist deine Menstruation heute?</p>

<div class="menstruation-options">

<div class="menstruation-card" id="period-nicht" data-value="nicht" onclick="selectMenstruation(this)">

<span class="menstruation-icon">🚫</span>

<span class="menstruation-label">Keine</span>

</div>

<div class="menstruation-card" id="period-leicht" data-value="leicht" onclick="selectMenstruation(this)">

<span class="menstruation-icon">💧</span>

<span class="menstruation-label">Leicht</span>

</div>

<div class="menstruation-card" id="period-mittel" data-value="mittel" onclick="selectMenstruation(this)">

<span class="menstruation-icon">💧💧</span>

<span class="menstruation-label">Mittel</span>

</div>

<div class="menstruation-card" id="period-stark" data-value="stark" onclick="selectMenstruation(this)">

<span class="menstruation-icon">💧💧💧</span>

<span class="menstruation-label">Stark</span>

</div>

</div>

<div id="menstruationWarning" class="warning-yellow"></div>

<div style="text-align: center">

<span class="errormessage" style="display: none" id="periodError">Bitte wähle die Intensität der Menstruationsblutung aus</span>

</div>

<div class="button" style="text-align: center" id="savePeriod">

Weiter

</div>

</div>

<div id="sports-section" style="display: none">

<h2>Sportliche Aktivitäten</h2>

<p class="sports-intro">Welche sportlichen Aktivitäten hast du heute ausgeübt?</p>

<div class="sports-input-container">

<!-- Sportart Auswahl -->

<div class="sport-type-selector">

<label>Art der Aktivität:</label>

<div class="sport-categories">

<div class="sport-category" data-intensity="high">

<input type="radio" name="sport-type" id="endurance">

<label for="endurance">

<span class="sport-icon">🏃‍♀️</span>

<span>Ausdauersport</span>

<small>(Laufen, Radfahren, Schwimmen)</small>

</label>

</div>

<div class="sport-category" data-intensity="medium">

<input type="radio" name="sport-type" id="strength">

<label for="strength">

<span class="sport-icon">💪</span>

<span>Krafttraining</span>

<small>(Gewichte, Eigengewicht)</small>

</label>

</div>

<div class="sport-category" data-intensity="low">

<input type="radio" name="sport-type" id="flexibility">

<label for="flexibility">

<span class="sport-icon">🧘‍♀️</span>

<span>Beweglichkeit</span>

<small>(Yoga, Dehnen, Pilates)</small>

</label>

</div>

</div>

</div>

<!-- Intensität und Dauer -->

<div class="intensity-options">

<button type="button" class="intensity-btn" id="intensity\_light" data-intensity="light" onclick="selectIntensity(this)">Leicht</button>

<button type="button" class="intensity-btn" id="intensity\_moderate" data-intensity="moderate" onclick="selectIntensity(this)">Moderat</button>

<button type="button" class="intensity-btn" id="intensity\_intense" data-intensity="intense" onclick="selectIntensity(this)">Intensiv</button>

</div>

<div class="duration-input">

<label for="duration">Dauer (Minuten):</label>

<input type="number" id="duration" min="1" max="300">

</div>

</div>

<!-- Zusätzliche Notizen -->

<div class="notes-section">

<label for="activities">Zusätzliche Notizen:</label>

<textarea id="activities" rows="2" placeholder="Wie hast du dich während des Sports gefühlt?"></textarea>

</div>

<div class="button" style="margin-top: 20px; text-align: center" id="saveSport">

Abschließen

</div>

<!-- Dynamische Warnung/Info Box -->

<div id="sportWarning" class="sport-warning"></div>

</div>

</div>

</div>

<script type="module">

import {saveMood, saveSymptoms, savePeriod, saveSport} from "./js/daily.js"

document.addEventListener("DOMContentLoaded", function() {

document.getElementById("saveMood").addEventListener("click", saveMood)

document.getElementById("saveSymptoms").addEventListener("click", saveSymptoms)

document.getElementById("savePeriod").addEventListener("click", savePeriod)

document.getElementById("saveSport").addEventListener("click", saveSport)

});

</script>

</body>

Index Page

<!DOCTYPE html>

<html lang="de">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Anämie-Früherkennung</title>

<link rel="stylesheet" href="./style.css">

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

<script type="module" src="./js/functions.js"></script>

<script type="module" src="./js/user.js"></script>

<script type="module" src="./js/camera.js"></script>

<script type="module" src="./js/profile.js"></script>

<script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>

<script type="module">

import {verify} from "./js/user.js";

import {updateHistoryChart} from "./js/functions.js";

await verify("standard")

updateHistoryChart()

</script>

<script>

document.addEventListener("DOMContentLoaded", function() {

prefillData()

document.getElementById("takePhoto").addEventListener("click", takePhoto)

document.getElementById("openCam").addEventListener("click", capture)

document.getElementById("takePhotoAgain").addEventListener("click", capture)

document.getElementById("analyze").addEventListener("click", detectNails)

document.getElementById("selectBoxes").addEventListener("click", changeBoxes)

document.getElementById("confirmBoxes").addEventListener("click", predictHb)

});

</script>

</head>

<body>

<div class="navbar">

<a href="#" style="background-color: #575757">

<span class="icon">🔬</span>

<div>Analyse</div>

</a>

<a href="./daily.html">

<span class="icon">📅</span>

<div>Daily Stuff</div>

</a>

<a href="./profil.html">

<span class="icon">👤</span>

<div>Profil</div>

</a>

<a href="./anaemie.html">

<span class="icon">💉</span>

<div>Anämie</div>

</a>

<a href="./ueber-uns.html">

<span class="icon">ℹ️</span>

<div>Über uns</div>

</a>

</div>

<div class="main-content">

<div id="analyse" class="container active">

<h1>Anämie-Früherkennung</h1>

<div class="instructions">

<h2>Anleitung:</h2>

<ol>

<li>Positionieren Sie Ihren Finger auf einer sauberen Glasfläche.</li>

<li>Stellen Sie sicher, dass ausreichend Licht vorhanden ist</li>

<li>Klicken Sie auf "Kamera öffnen"</li>

<li>Nehmen Sie das Bild auf und klicken Sie auf "Analysieren"</li>

<li>Warten Sie auf die Analyse</li>

</ol>

</div>

<div class="camera-container">

<video id="video" autoplay></video>

<div>

<div class="cam-container">

<button class="button cam-button" id="openCam">

<span class="icon">📷</span>

Kamera öffnen

</button>

<button class="button cam-button" id="takePhoto">

<span class="icon">📷</span>

Foto aufnehmen

</button>

<canvas id="canvas"></canvas>

<button class="button cam-button" id="takePhotoAgain">

<span class="icon">📷</span>

Neues Foto aufnehmen

</button>

<button class="button" id="analyze" style="display: none">

<span class="icon">🔎</span>

Analysieren

</button>

<button class="button" id="selectBoxes" style="display: none">

Erkannte Fingernägel ändern

</button>

<button class="button" id="confirmBoxes" style="display: none">

<span class="icon">☑️</span>

Korrekte Erkennung bestätigen

</button>

</div>

<div id="noFingernailsDetected" style="display: none">

<span>Keine Fingernägel erkannt. Bitte nimm ein neues Foto auf.</span>

</div>

</div>

</div>

<div class="results" id="results" style="display: none">

<h2>Analyseergebnisse</h2>

<div class="result-card">

<div class="result-header">

<h3>Hämoglobinwert</h3>

</div>

<div class="result-value" id="hemoglobinValue">-- g/L</div>

<div class="meter-container">

<div class="meter">

<div class="meter-bar" id="hemoglobinBar"></div>

</div>

<div class="meter-labels">

<span>Niedrig (<120 g/L) </span>

<span>Normal (120-160 g/L)</span>

<span>Hoch (>160 g/L)</span>

</div>

</div>

<div class="recommendation-box" id="recommendation"></div>

</div>

</div>

<div class="history-chart">

<canvas id="historyChart"></canvas>

</div>

<div class="details-grid">

<div class="detail-item">

<span class="label">Letzter Test:</span>

<span class="value" id="lastTest"></span>

</div>

<div class="detail-item">

<span class="label">Veränderung in g/L:</span>

<span class="value" id="change"></span>

</div>

</div>

<div class="export-options">

<button class="button" onclick="exportPDF()">Als PDF exportieren</button>

<button class="button" onclick="shareResults()">Ergebnisse teilen</button>

</div>

</div>

</div>

</body>

</html>

Login Page

<!DOCTYPE html>

<html lang="de">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Anämie-Früherkennung | Login</title>

<script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0/css/all.min.css">

<link rel="stylesheet" href="./style.css">

<script type="module" src="./js/user.js"></script>

<script type="module">

import {verify} from "./js/user.js";

(async() => {

await verify("loginPage").then(instLogin => {

console.log("instLogin", instLogin)

if (instLogin) {

window.location.href = "/index.html"

}

}

)

})()

</script>

<script>

document.addEventListener("DOMContentLoaded", async function () {

document.getElementById("loginform").addEventListener("submit", function (event) {

event.preventDefault();

login();

});

});

</script>

<style>

.login-error {

color: red;

font-size: smaller;

text-align: center;

display: none;

}

.oderdiv {

text-align: center;

font-size: small;

}

</style>

</head>

<body>

<div class="main-content">

<div class="login-container">

<h1>Login</h1>

<form id="loginform">

<div class="input-container">

<label for="username">Benutzername</label>

<input id="username" type="text" name="username" placeholder="Benutzername" required>

</div>

<div class="input-container">

<label for="password">Passwort</label>

<input id="password" type="password" name="password" placeholder="Passwort" required>

</div>

<div style="text-align: center">

<span id="loginfailure" class="login-error">Login war nicht erfolgreich.</span>

</div>

<div class="login-button-container">

<button class="button" type="submit">Login</button>

</div>

</form>

<div class="oderdiv">

<div class="separator"><span>oder</span></div>

<a href="./register.html">

<button class="button">Registrieren</button>

</a>

</div>

</div>

</div>

</body>

</html>

Profil Page:

<!DOCTYPE html>

<html lang="de">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Anämie-Früherkennung | Profil</title>

<link rel="stylesheet" href="./style.css">

<script type="module" src="./js/functions.js"></script>

<script type="module" src="./js/user.js"></script>

<script type="module" src="./js/profile.js"></script>

<script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>

<script type="module">

import {verify, getUserdata} from "./js/user.js";

import {loadProfilData} from "./js/profile.js";

verify("standard")

getUserdata();

loadProfilData();

let neueVorerkrankungen = JSON.parse(localStorage.getItem("vorerkrankung") || "[]")

let neueMedikamente = JSON.parse(localStorage.getItem("medikamente") || "[]")

for(let i = 0; i < neueVorerkrankungen.length; i++){

addItemToList("vorerkrankungenListe", neueVorerkrankungen[i])

}

for(let j = 0; j < neueMedikamente.length; j++){

addItemToList("medikamenteListe", neueMedikamente[j])

}

</script>

</head>

<body>

<div class="navbar">

<a href="./index.html">

<span class="icon">🔬</span>

<div>Analyse</div>

</a>

<a href="./daily.html">

<span class="icon">📅</span>

<div>Daily Stuff</div>

</a>

<a href="#" style="background-color: #575757">

<span class="icon">👤</span>

<div>Profil</div>

</a>

<a href="./anaemie.html">

<span class="icon">💉</span>

<div>Anämie</div>

</a>

<a href="./ueber-uns.html">

<span class="icon">ℹ️</span>

<div>Über uns</div>

</a>

</div>

<div class="main-content">

<div id="profil" class="container">

<div class="profil-header">

<h1>Mein Gesundheitsportal</h1>

<button class="button" onclick="logout()">Logout</button>

</div>

<form id="profilForm">

<div class="section">

<h2>Persönliche Daten</h2>

<div class="form-group">

<label for="name">Name:</label>

<input type="text" id="name" name="name" class="input-box mb-10px" required>

</div>

<div class="form-group">

<label for="geburtsdatum">Geburtsdatum:</label>

<input type="date" id="geburtsdatum" name="geburtsdatum" class="input-box mb-10px" required>

</div>

<div class="form-group">

<label for="geschlecht">Geschlecht:</label>

<select id="geschlecht" name="geschlecht" class="input-box mb-10px" required>

<option value="">Bitte wählen</option>

<option value="weiblich">Weiblich</option>

<option value="männlich">Männlich</option>

<option value="divers">Divers</option>

</select>

</div>

</div>

<div class="section">

<h2>Vorerkrankungen</h2>

<div class="form-group dyn-list">

<div class="tag-list" id="vorerkrankungenListe">

<!-- Hier werden die Vorerkrankungen eingefügt -->

</div>

<input type="text" id="neueVorerkrankung" class="input-box" style="width: 75%" placeholder="Neue Vorerkrankung eingeben">

<button type="button" class="button" onclick="addVorerkrankung()">Hinzufügen</button>

</div>

</div>

<div class="section">

<h2>Medikamente</h2>

<div class="form-group dyn-list">

<div class="tag-list" id="medikamenteListe">

<!-- Hier werden die Medikamente eingefügt -->

</div>

<input type="text" id="neuesMedikament" class="input-box" style="width: 75%" placeholder="Neues Medikament eingeben">

<button type="button" class="button" onclick="addMedikament()">Hinzufügen</button>

</div>

</div>

<button type="submit" class="button" style="margin-top: 20px;" onclick="saveUserdata()">Speichern</button>

</form>

</div>

</div>

</body>

</html>

Register Page:

<!DOCTYPE html>

<html lang="de">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Anämie-Früherkennung | Registrierung</title>

<script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0/css/all.min.css">

<link rel="stylesheet" href="./style.css">

<script type="module" src="./js/user.js"></script>

<script>

document.addEventListener("DOMContentLoaded", function() {

document.getElementById("registerform").addEventListener("submit", function(event) {

event.preventDefault();

register();

});

});

</script>

<style>

.error {

color: red;

font-size: smaller;

text-align: center;

display: none;

}

#registerform {

justify-content: center;

}

.oderdiv {

text-align: center;

font-size: small;

}

</style>

</head>

<body>

<div class="main-content">

<div class="login-container">

<h1>Registrierung</h1>

<form id="registerform">

<div class="input-container">

<label for="username">Benutzername</label>

<input id="username" type="text" name="username" placeholder="Benutzername" required>

</div>

<div class="input-container">

<label for="password">Passwort</label>

<input id="password" type="password" name="password" placeholder="Passwort" required>

</div>

<div class="input-container">

<input id="password2" type="password" name="password2" placeholder="Passwort wiederholen" required>

</div>

<div style="text-align: center">

<span id="errormessage" class="error"></span>

</div>

<div class="login-button-container">

<button class="button" type="submit">Registrieren</button>

</div>

</form>

<div class="oderdiv">

<div class="separator"><span>oder</span></div>

<a href="./login.html">

<button class="button">Login</button>

</a>

</div>

</div>

</div>

</body>

</html>

CSS Styles:

:root {

--primary-green: #8ac79f;

--primary-light: #c1e0cb;

--primary-dark: #5b9e6f;

--accent-green: #a8dabc;

--light-bg: #f0f7f3;

--dark-text: #2c3e33;

--light-text: #4a6354;

--very-light-text: #6e8577;

--shadow: rgba(0, 0, 0, 0.1);

--warning-light: #fff3cd;

--warning-dark: #ffc107;

--danger-light: #f8d7da;

--danger-dark: #dc3545;

}

html, body {

overflow-x: hidden;

margin: 0;

}

body {

font-family: 'Roboto', sans-serif;

background-color: var(--light-bg);

color: var(--dark-text);

}

h2 {

color: var(--primary-dark);

margin-top: 20px;

}

input[type="date"]{

font-family: inherit;

}

textarea {

border: 1px solid var(--primary-light);

border-radius: 5px;

padding: 8px;

font-family: inherit;

}

.separator {

display: flex;

align-items: center;

text-align: center;

margin: 20px 0;

}

.separator::before,

.separator::after {

content: "";

flex: 1;

border-bottom: 1px solid var(--light-bg);

}

.separator:not(:empty)::before {

margin-right: 0.75em;

}

.separator:not(:empty)::after {

margin-left: 0.75em;

}

.separator span {

white-space: nowrap;

color: #666;

font-size: 0.9em;

}

.main-content {

max-width: 800px;

margin: 0 auto;

padding: 20px 20px 120px 20px;

}

.container {

background-color: white;

padding: 20px;

border-radius: 10px;

box-shadow: 0 4px 6px var(--shadow);

}

.container.active {

display: block;

}

.cam-container {

display: flex;

justify-content: center;

flex-wrap: wrap;

}

.cam-button {

width: fit-content;

}

.cam-button .icon {

padding-right: 5px;

}

.login-container {

display: flex;

justify-content: center;

align-items: center;

padding: 20px;

flex-wrap: wrap;

flex-direction: column;

}

.login-container input {

width: 150px;

padding: 10px;

margin-bottom: 10px;

border: 1px solid var(--primary-light);

border-radius: 4px;

}

.input-container label {

font-size: smaller;

display: block;

margin-bottom: 5px;

color: var(--light-text);

}

.login-button-container {

display: flex;

justify-content: center;

}

.profil-header {

display: flex;

justify-content: space-between;

align-items: center;

width: 100%;

}

.tabs {

overflow: hidden;

border-bottom: 1px solid var(--primary-light);

background-color: var(--light-bg);

border-radius: 10px 10px 0 0;

}

.tablinks {

background-color: var(--light-bg);

float: left;

border: none;

outline: none;

cursor: pointer;

padding: 14px 16px;

transition: 0.3s;

font-size: 17px;

color: var(--light-text);

}

.tablinks:hover {

background-color: var(--primary-light);

color: var(--dark-text);

}

.tablinks.active {

background-color: var(--primary-green);

color: white;

}

.tabcontent {

display: none;

padding: 20px;

border: 1px solid var(--primary-light);

border-top: none;

border-radius: 0 0 10px 10px;

background-color: white;

}

#video {

width: 100%;

border-radius: 10px;

margin-bottom: 10px;

display: none;

}

#canvas {

display: none;

width: 100%;

border-radius: 10px;

margin-bottom: 10px;

}

#takePhoto {

display: none;

}

#takePhotoAgain {

display: none;

}

.button {

background-color: var(--primary-green);

color: white;

padding: 10px 20px;

border: none;

border-radius: 5px;

cursor: pointer;

margin: 5px;

transition: background-color 0.3s, transform 0.3s;

box-shadow: 0 4px 6px var(--shadow);

}

.button:hover {

background-color: var(--primary-dark);

transform: translateY(-2px);

}

.results {

margin-top: 20px;

padding: 15px;

border: 1px solid var(--primary-light);

border-radius: 5px;

background-color: white;

}

.instructions {

margin: 20px 0;

padding: 15px;

background-color: var(--primary-light);

border-radius: 5px;

}

.result-card {

background: white;

border-radius: 10px;

padding: 20px;

box-shadow: 0 4px 6px var(--shadow);

margin-bottom: 20px;

border-left: 4px solid var(--primary-green);

}

.result-header {

display: flex;

align-items: center;

gap: 10px;

margin-bottom: 15px;

color: var(--primary-dark);

}

.result-value {

font-size: 36px;

font-weight: bold;

color: var(--primary-dark);

text-align: center;

margin: 20px 0;

}

.meter-container {

margin: 20px 0;

}

.meter {

width: 100%;

height: 20px;

background-color: var(--light-bg);

border-radius: 10px;

overflow: hidden;

}

.meter-bar {

height: 100%;

width: 0%;

background: linear-gradient(90deg, var(--primary-light), var(--primary-green), var(--primary-dark));

transition: width 0.5s ease-in-out;

}

.meter-labels {

display: flex;

justify-content: space-between;

margin-top: 5px;

color: var(--very-light-text);

}

.history-chart {

margin: 20px 0;

height: 200px;

display: flex;

justify-content: center;

}

#historyChart {

width: 100%;

}

.details-grid {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));

gap: 20px;

margin: 20px 0;

}

.detail-item {

background: white;

padding: 15px;

border-radius: 8px;

box-shadow: 0 2px 5px var(--shadow);

border-top: 3px solid var(--primary-green);

}

.export-options {

display: flex;

gap: 10px;

margin-top: 20px;

}

.recommendation-box {

background-color: var(--primary-light);

border-left: 4px solid var(--primary-dark);

padding: 15px;

margin-top: 20px;

border-radius: 4px;

}

.content {

flex: 1;

padding: 20px;

text-align: center;

}

.navbar {

height: 80px;

display: flex;

justify-content: space-around;

background: linear-gradient(90deg, var(--primary-green), var(--primary-dark));

position: fixed;

bottom: 0;

left: 0;

right: 0;

width: 100vw;

z-index: 1000;

margin: 0;

box-shadow: 0 -2px 5px var(--shadow);

}

.navbar a {

width: 20%;

padding: 10px;

color: white;

text-align: center;

text-decoration: none;

display: flex;

justify-content: center;

align-items: center;

flex-direction: column;

transition: background-color 0.3s, transform 0.3s;

}

.navbar a:hover {

background-color: var(--primary-dark);

transform: translateY(-5px);

box-shadow: 0 4px 8px var(--shadow);

}

.navbar .icon {

font-size: 24px;

margin-bottom: 5px;

}

.navbar a span {

font-size: 14px;

}

.symptoms, .risk-groups, .treatment {

background-color: white;

padding: 15px;

margin: 15px 0;

border-radius: 5px;

box-shadow: 0 1px 3px var(--shadow);

}

ul {

list-style-type: none;

padding-left: 20px;

}

li {

margin: 8px 0;

position: relative;

}

.description, .treatment p {

line-height: 1.6;

color: var(--dark-text);

}

.delete-btn {

background-color: var(--danger-dark);

padding: 4px 8px;

color: white;

border: none;

border-radius: 4px;

cursor: pointer;

}

.delete-btn:hover {

background-color: var(--danger-light);

}

.section {

background-color: white;

padding: 20px;

margin-bottom: 20px;

border-radius: 8px;

box-shadow: 0 2px 4px var(--shadow);

border-top: 3px solid var(--primary-green);

}

.form-group {

margin-bottom: 15px;

width: 100%;

}

.dyn-list {

display: block;

vertical-align: center

}

.tag-list {

display: flex;

flex-wrap: wrap;

gap: 8px;

margin-bottom: 10px;

}

.list-tag {

background-color: #f8f9fa;

border: 1px solid #d1d1d1;

border-radius: 20px;

padding: 5px 10px;

display: flex;

align-items: center;

font-size: 14px;

color: #333;

}

.list-tag .delete-btn {

background-color: #dc3545;

color: white;

border: none;

border-radius: 50%;

width: 18px;

height: 18px;

font-size: 12px;

margin-left: 8px;

cursor: pointer;

display: flex;

align-items: center;

justify-content: center;

}

label {

display: block;

margin-bottom: 5px;

font-weight: bold;

color: var(--light-text);

}

.input-box {

padding: 10px;

border: 1px solid var(--primary-light);

border-radius: 5px;

color: var(--dark-text);

}

.input-box:focus {

outline: none;

border-color: var(--primary-green);

box-shadow: 0 0 0 2px var(--primary-light);

}

.mb-10px {

margin-bottom: 10px

}

/\*

.modal {

display: none;

position: fixed;

z-index: 1000;

left: 0;

top: 0;

width: 100%;

height: 100%;

background-color: var(--shadow);

}

.modal-content {

background-color: white;

margin: 15% auto;

padding: 20px;

border: 1px solid var(--primary-light);

width: 80%;

max-width: 600px;

border-radius: 8px;

position: relative;

animation: slideIn 0.3s ease-out;

box-shadow: 0 5px 15px var(--shadow);

}

@keyframes slideIn {

from {

transform: translateY(-100px);

opacity: 0;

}

to {

transform: translateY(0);

opacity: 1;

}

}

.modal-content h2 {

color: var(--primary-dark);

margin-top: 0;

}

.modal-content ul {

margin: 15px 0;

padding-left: 20px;

}

.modal-content li {

margin: 8px 0;

color: var(--dark-text);

}\*/

.close-button {

position: absolute;

right: 10px;

top: 10px;

color: var(--very-light-text);

font-size: 28px;

font-weight: bold;

cursor: pointer;

}

.close-button:hover {

color: var(--light-text);

}

/\* Warning Animation \*/

/\*

@keyframes shake {

0%, 100% { transform: translateX(0); }

10%, 30%, 50%, 70%, 90% { transform: translateX(-5px); }

20%, 40%, 60%, 80% { transform: translateX(5px); }

}

.warning-shake {

animation: shake 0.5s ease-in-out;

}

.warning-content {

padding: 15px;

background-color: var(--warning-light);

border-left: 4px solid var(--warning-dark);

margin: 10px 0;

border-radius: 4px;

}

.important-notice {

margin-top: 20px;

padding: 15px;

background-color: var(--danger-light);

border-left: 4px solid var(--danger-dark);

border-radius: 4px;

}

.important-notice strong {

color: var(--danger-dark);

}\*/

/\* Daily Stuff \*/

/\* Stimmungsbarometer \*/

.daily-header {

text-align: center;

margin-bottom: 30px;

}

.greeting {

font-size: 1.2em;

color: var(--light-text);

margin: 15px 0;

}

.mood-section {

text-align: center;

margin-bottom: 40px;

}

.mood-question {

font-size: 1.4em;

color: var(--dark-text);

margin-bottom: 20px;

}

.mood-selector {

display: flex;

justify-content: center;

gap: 20px;

margin-bottom: 30px;

}

.mood-option {

display: flex;

flex-direction: column;

align-items: center;

padding: 15px;

border-radius: 12px;

cursor: pointer;

transition: all 0.3s ease;

}

.mood-option:hover {

transform: translateY(-5px);

background-color: var(--light-bg);

}

.mood-option.selected {

background-color: var(--primary-light);

border: 2px solid var(--primary-green);

}

.emoji {

font-size: 2.5em;

margin-bottom: 8px;

}

.mood-label {

font-size: 0.9em;

color: var(--light-text);

}

.mood-response {

margin: 20px 0;

padding: 15px;

border-radius: 8px;

font-size: 1.1em;

display: none;

background-color: var(--primary-light);

}

#tracking-section {

background-color: white;

padding: 25px;

border-radius: 12px;

margin-top: 30px;

box-shadow: 0 2px 10px var(--shadow);

border-top: 3px solid var(--primary-green);

}

/\* Animation für Mood Response \*/

@keyframes fadeIn {

from { opacity: 0; transform: translateY(-10px); }

to { opacity: 1; transform: translateY(0); }

}

.mood-response.show {

display: block;

animation: fadeIn 0.5s ease-out;

}

/\* Symptome \*/

#symptoms-section {

background-color: white;

padding: 30px;

border-radius: 12px;

margin: 30px 0;

box-shadow: 0 2px 10px var(--shadow);

border-top: 3px solid var(--primary-green);

}

.symptoms-intro {

color: var(--light-text);

margin-bottom: 20px;

}

.symptoms-grid {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));

gap: 15px;

margin-bottom: 20px;

}

.symptom-card {

position: relative;

background-color: var(--light-bg);

border-radius: 15px;

transition: all 0.3s ease;

}

.symptom-card:hover {

transform: translateY(-2px);

box-shadow: 0 2px 8px var(--shadow);

}

.symptom-checkbox {

display: none;

}

.symptom-card label {

display: flex;

flex-direction: column;

align-items: center;

padding: 15px;

cursor: pointer;

height: 100%;

border: 2px solid transparent;

cursor: pointer;

height: 100%;

border-radius: 10px;

}

.symptom-checkbox:checked + label {

background-color: var(--primary-light);

border-color: var(--primary-green);

padding: 2px

}

.symptom-icon {

font-size: 2em;

margin-bottom: 15px;

color: var(--primary-dark);

}

.symptom-name {

text-align: center;

font-size: 0.9em;

color: var(--dark-text);

}

.warning-severe {

background-color: var(--danger-light);

}

.warning-mild {

background-color: var(--warning-light);

}

.symptom-warning {

margin-top: 20px;

padding: 15px;

border-radius: 8px;

background-color: var(--warning-light);

border-left: 4px solid var(--warning-dark);

display: none;

animation: slideIn 0.3s ease-out;

}

/\* Menstruation \*/

.menstruation-options {

display: flex;

justify-content: center;

gap: 20px;

margin: 20px 0;

flex-wrap: wrap;

}

.menstruation-card {

display: flex;

flex-direction: column;

align-items: center;

padding: 20px;

background-color: var(--light-bg);

border-radius: 12px;

cursor: pointer;

transition: all 0.3s ease;

min-width: 120px;

border: 2px solid transparent;

}

.menstruation-card:hover {

transform: translateY(-3px);

box-shadow: 0 4px 8px var(--shadow);

}

.menstruation-card.selected {

background-color: var(--primary-light);

border-color: var(--primary-green);

}

.menstruation-icon {

font-size: 1.8em;

margin-bottom: 10px;

color: var(--primary-dark);

}

.menstruation-label {

font-size: 1em;

color: var(--dark-text);

font-weight: 500;

}

.warning-yellow {

margin: 20px 0 20px 0;

padding: 15px;

border-radius: 8px;

display: none;

animation: slideIn 0.3s ease-out;

background-color: var(--warning-light);

border-left: 4px solid var(--warning-dark);

}

.warning-strong {

background-color: var(--warning-light);

border-left: 4px solid var(--warning-dark);

}

/\* Sport \*/

#sports-section {

background-color: white;

padding: 25px;

border-radius: 12px;

margin: 30px 0;

box-shadow: 0 2px 10px var(--shadow);

border-top: 3px solid var(--primary-green);

}

.sports-intro {

color: var(--light-text);

margin-bottom: 20px;

}

.sports-input-container {

display: flex;

flex-direction: column;

gap: 20px;

}

.sport-categories {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));

gap: 15px;

margin-top: 10px;

}

.sport-category {

background-color: var(--light-bg);

border-radius: 10px;

padding: 15px;

cursor: pointer;

transition: all 0.3s ease;

text-align: center;

}

.sport-category:hover {

transform: translateY(-2px);

box-shadow: 0 4px 8px var(--shadow);

}

.sport-category input[type="radio"] {

display: none;

}

.sport-category label {

display: flex;

flex-direction: column;

align-items: center;

gap: 5px;

cursor: pointer;

}

.sport-icon {

font-size: 2em;

color: var(--primary-dark);

}

.sport-category small {

color: var(--very-light-text);

font-size: 0.8em;

}

.sport-category:has(input[type="radio"]:checked) {

border: 2px solid var(--primary-dark);

border-radius: 0.5rem;

background-color: var(--primary-light)

}

.intensity-options {

display: flex;

gap: 10px;

margin-top: 10px;

}

.intensity-btn {

padding: 8px 15px;

border: none;

border-radius: 20px;

cursor: pointer;

transition: all 0.3s ease;

background-color: var(--light-bg);

color: var(--dark-text);

}

.intensity-btn:hover {

background-color: var(--primary-light);

}

.intensity-btn.selected {

background-color: var(--primary-green);

color: white;

}

.duration-input input {

width: 100px;

padding: 8px;

border: 1px solid var(--primary-light);

border-radius: 5px;

}

.notes-section {

margin-top: 10px;

}

.notes-section textarea {

width: calc(100% - 25px);

}

.sport-warning {

margin-top: 20px;

padding: 15px;

border-radius: 8px;

display: none;

animation: slideIn 0.3s ease-out;

}

.save-section {

text-align: center;

margin-top: 30px;

}

/\* Responsive Design \*/

@media (max-width: 600px) {

body {

max-width: 100%;

padding: 0;

}

.symptoms-grid,

.sport-categories,

.menstruation-options {

grid-template-columns: repeat(2, 1fr);

}

.menstruation-card {

min-width: calc(50% - 10px);

}

.modal-content {

margin: 10% auto;

width: 95%;

padding: 15px;

}

.warning-content,

.important-notice {

padding: 10px;

}

.mood-selector {

gap: 10px;

}

}

#noFingernailsDetected {

text-align:center;

width:100%;

color:indianred;

margin-top:10px

}

.errormessage {

color: red;

font-size: 14px;

margin-bottom: 1rem;

}

READ ME

**# Starten des Frontends**

**## Beim ersten Mal**

- [NodeJS herunterladen](https://nodejs.org/en/download) und installieren

-> unten "prebuilt", wenn M1/2/3/4-Chip ARM64, sonst x64 für MacOS

**## Starten**

```bash

npm start

```

**## Frontend aufrufen**

http://localhost:8080 im Browser

Wenn man etwas am Frontend-Code ändert, wird das erkannt und direkt aktualisiert (ggf. dafür einmal in die Konsole klicken)

{

"devDependencies": {

"http-server": "^14.1.1",

"live-server": "^1.2.2"

},

"name": "projekt-anaemie",

"version": "1.0.0",

"main": "index.js",

"dependencies": {

"ansi-styles": "^4.3.0",

"async": "^3.2.6",

"basic-auth": "^2.0.1",

"call-bind-apply-helpers": "^1.0.2",

"call-bound": "^1.0.4",

"chalk": "^4.1.2",

"color-convert": "^2.0.1",

"color-name": "^1.1.4",

"corser": "^2.0.1",

"debug": "^4.4.1",

"dunder-proto": "^1.0.1",

"es-define-property": "^1.0.1",

"es-errors": "^1.3.0",

"es-object-atoms": "^1.1.1",

"eventemitter3": "^4.0.7",

"follow-redirects": "^1.15.9",

"function-bind": "^1.1.2",

"get-intrinsic": "^1.3.0",

"get-proto": "^1.0.1",

"gopd": "^1.2.0",

"has-flag": "^4.0.0",

"has-symbols": "^1.1.0",

"hasown": "^2.0.2",

"he": "^1.2.0",

"html-encoding-sniffer": "^3.0.0",

"http-proxy": "^1.18.1",

"iconv-lite": "^0.6.3",

"math-intrinsics": "^1.1.0",

"mime": "^1.6.0",

"minimist": "^1.2.8",

"ms": "^2.1.3",

"object-inspect": "^1.13.4",

"opener": "^1.5.2",

"portfinder": "^1.0.37",

"qs": "^6.14.0",

"requires-port": "^1.0.0",

"safe-buffer": "^5.1.2",

"safer-buffer": "^2.1.2",

"secure-compare": "^3.0.1",

"side-channel": "^1.1.0",

"side-channel-list": "^1.0.0",

"side-channel-map": "^1.0.1",

"side-channel-weakmap": "^1.0.2",

"supports-color": "^7.2.0",

"union": "^0.5.0",

"url-join": "^4.0.1",

"whatwg-encoding": "^2.0.0"

},

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1",

"start": "live-server --port=8080 --no-browser"

},

"keywords": [],

"author": "",

"license": "ISC",

"description": ""

}