# Modelowanie i analiza systemów informatycznych

dokumentacja projektu systemu ekspertowego do rozpoznawania cukrzycy wśród indian

Aleksander Kulpa, Mikołaj Macura, Paweł Habrzyk

10 grudnia 2023

# Część I

## Opis programu

Zaimplementuj system wspomagania lekarzy (poprzez użycie sieci neuronowej) poprzez automatyczną analizę danych medycznych. System posiada:

- dwa tryby – uczenie algorytmu/klasyfikacja nowej próbki
- dane medyczne szyfrowane i bezpieczne (o tym w następnej sekcji)
- dane lekarzy odpowiednio zabezpieczone (o tym w następnej sekcji)
- możliwość dodawania nowego pacjenta/lekarza
- wszystkie moduły zostały przetestowane
- sieć neuronowa została przeanalizowana pod względem ilości neuronów/warstw
- W zadaniu wykorzystaliśmy bazę danych: https://www.kaggle.com/uciml/pima-indians-diabetes-database

## Instrukcja obsługi

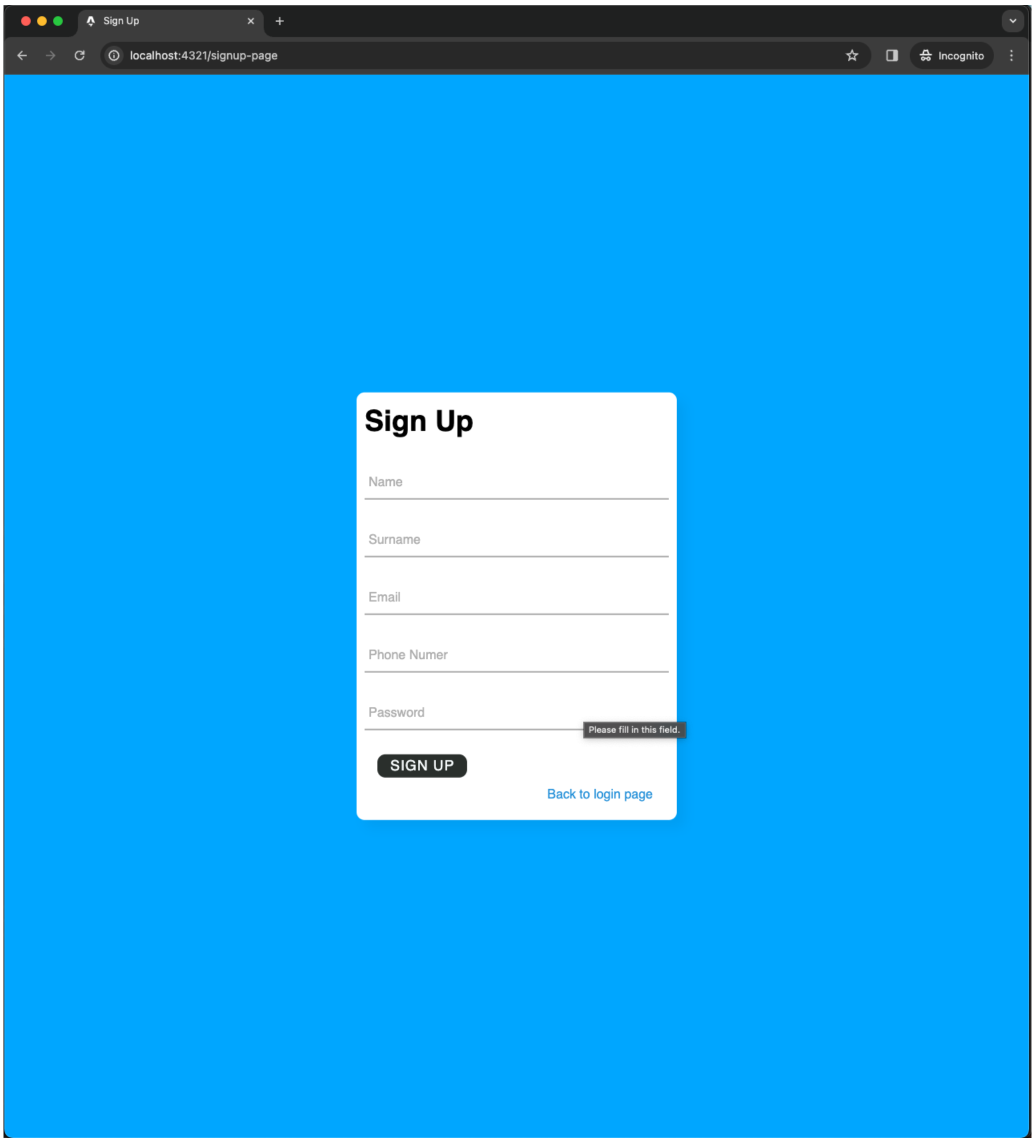
(Aby uruchomić aplikację potrzebny jest Docker: https://docs.docker.com/get-docker/)

Ściągamy kod źródłowy i wypakowujemy w dowolny miejscu na dysku. Jeżeli Docker jest zainstalowany uruchamiamy konsolę i z poziomu głównego folderu naszej aplikacji wykonujemy następujące komendy:
- 'docker compose build' - ściąga wszystkie pakiety dla naszych programów
- 'docker compose up' - uruchamia nasze aplikacje

Dostęp do interfejsu webowego naszej aplikacji znajduje się pod adresem "localhost:4321" a dostęp do endpointów RestAPI znajduje się pod adresem "localhost:8000/docs"

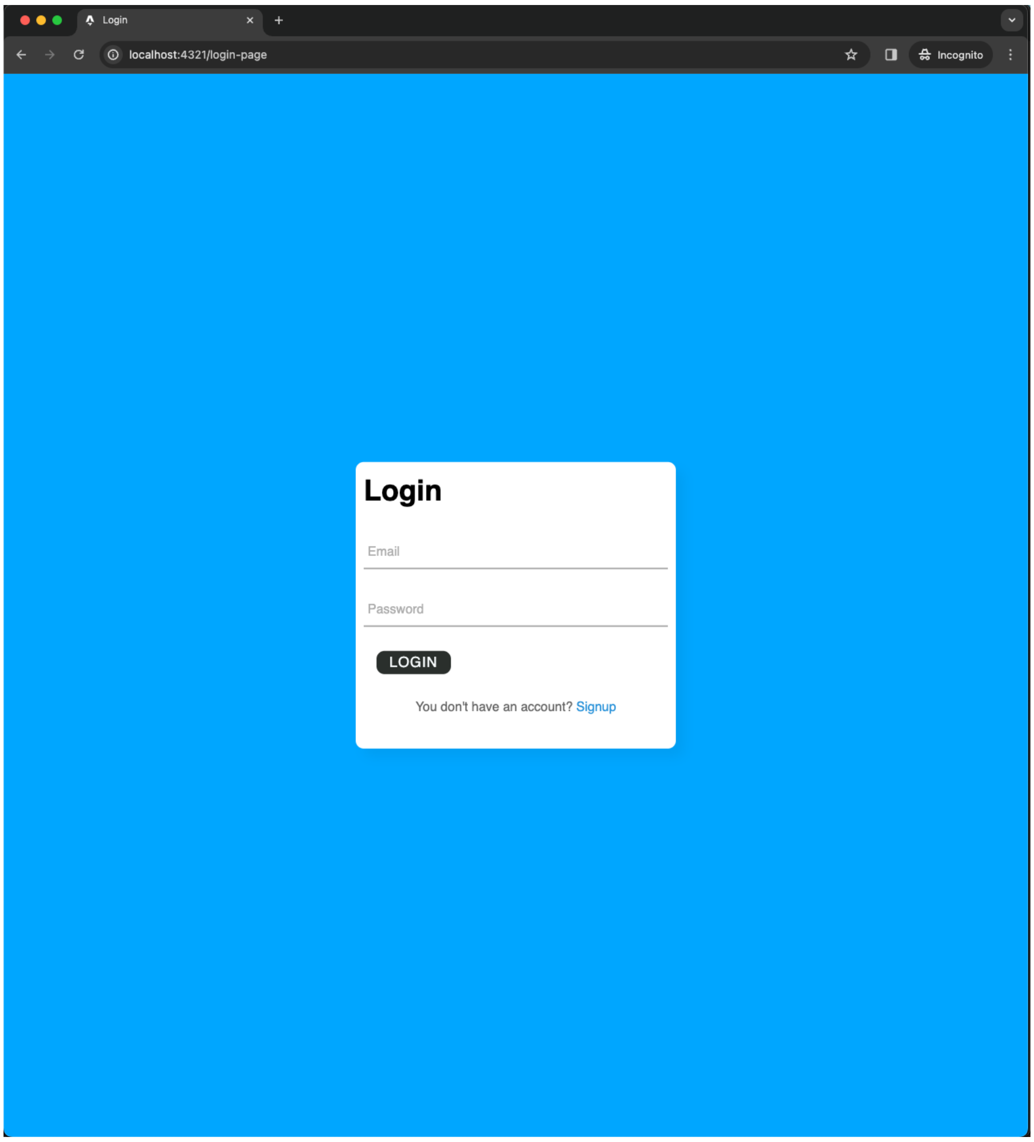
Aby zalogować się, do obu aplikacji domyślnym kontem adminastratora jest:
email: senior_registrar@hospital.com
hasło: passwd

Gdy obie aplikacje działają mamy dostępne możliwości jak poniżej.



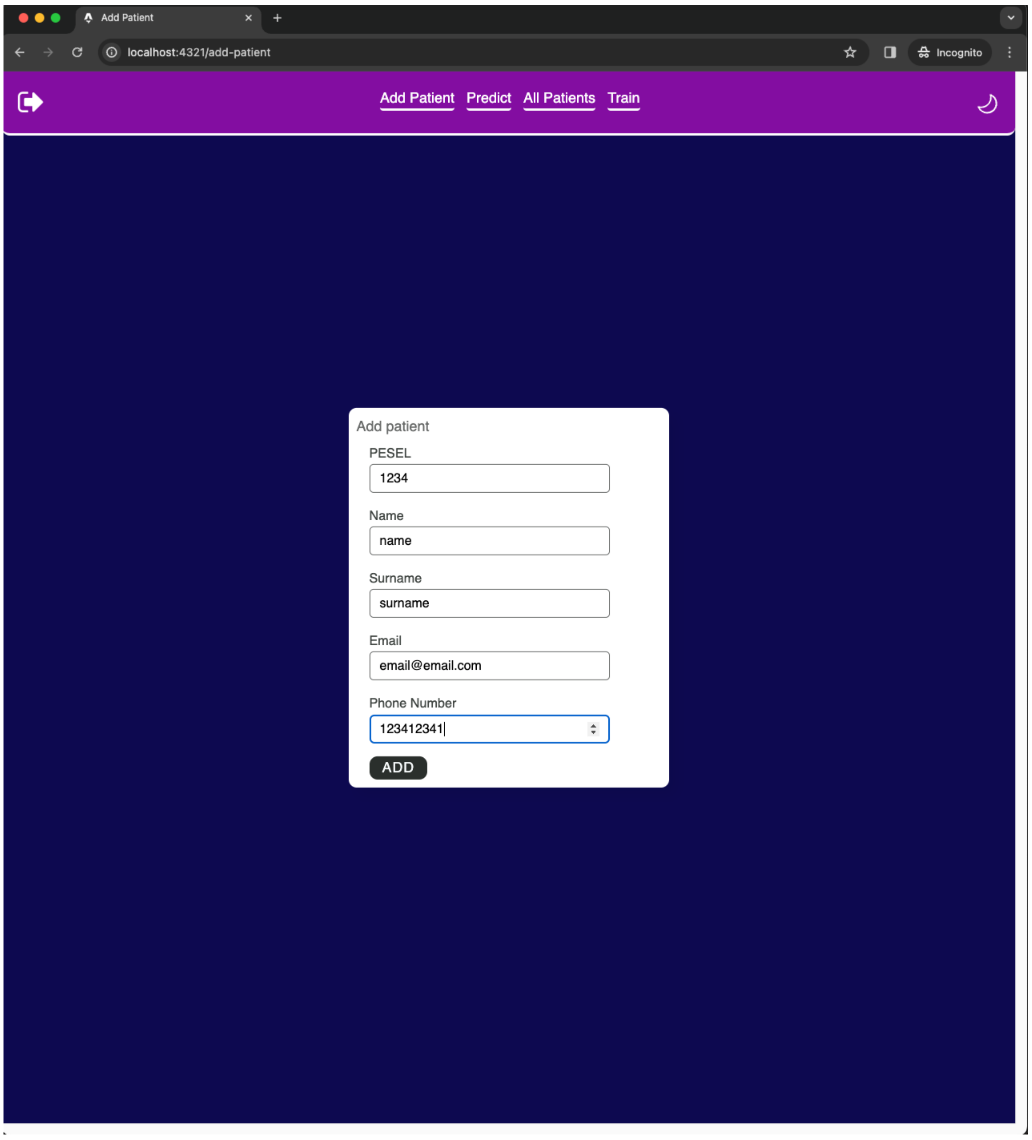
Rysunek 1: Możemy dodać lekarza

Pola Name, Surname są zabezpieczone przed wpisaniem znaków poza literami, Phone Number wymaga jedynie cyfr, których w sumie jest dziewięć. Pole email wymaga wpisania poprawnego emaila. Nie da się utworzyć konta bez któregoś z pól uzupełnionego.



Rysunek 2: Strona logowania się lekarzy posiadających konta stworzone w rysunku powyżej

Pole email wymaga wpisania poprawnego emaila. Nie da się utworzyć konta bez któregoś z pól uzupełnionego.

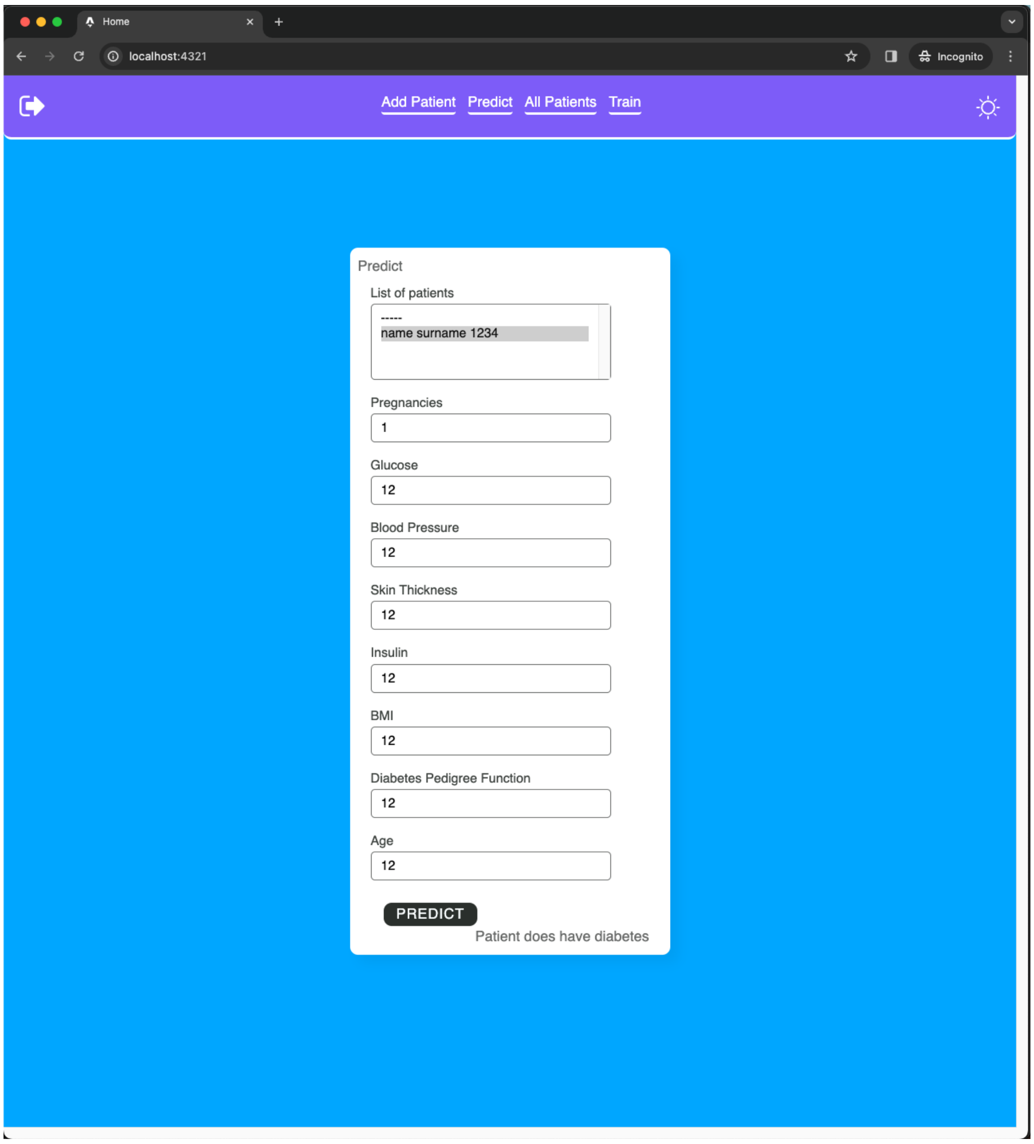


PESEL: 1234
Name: name
Surname: surname
Email: email@email.com
Phone Number: 123412341

Rysunek 3: Lekarz może dodać pacjenta, który poda mu swoje dane

Pole PESEL wymaga jedynie cyfr, minimum jednej. Pola Name, Surname są zabezpieczone przed wpisaniem znaków poza literami, Phone Number wymaga jedynie cyfr, których w sumie jest dziewięć. Pole email wymaga wpisania poprawnego emaila. Nie da się utworzyć pacjenta bez któregoś z pól uzupełnionego.

Również można zauważyć, iż zmieniło się tło na tryb nocny, w prawym górnym rogu steruje nim przycisk księżyca/słoneczka.



Predict

List of patients
-----
name surname 1234

Pregnancies
1

Glucose
12

Blood Pressure
12

Skin Thickness
12

Insulin
12

BMI
12

Diabetes Pedigree Function
12

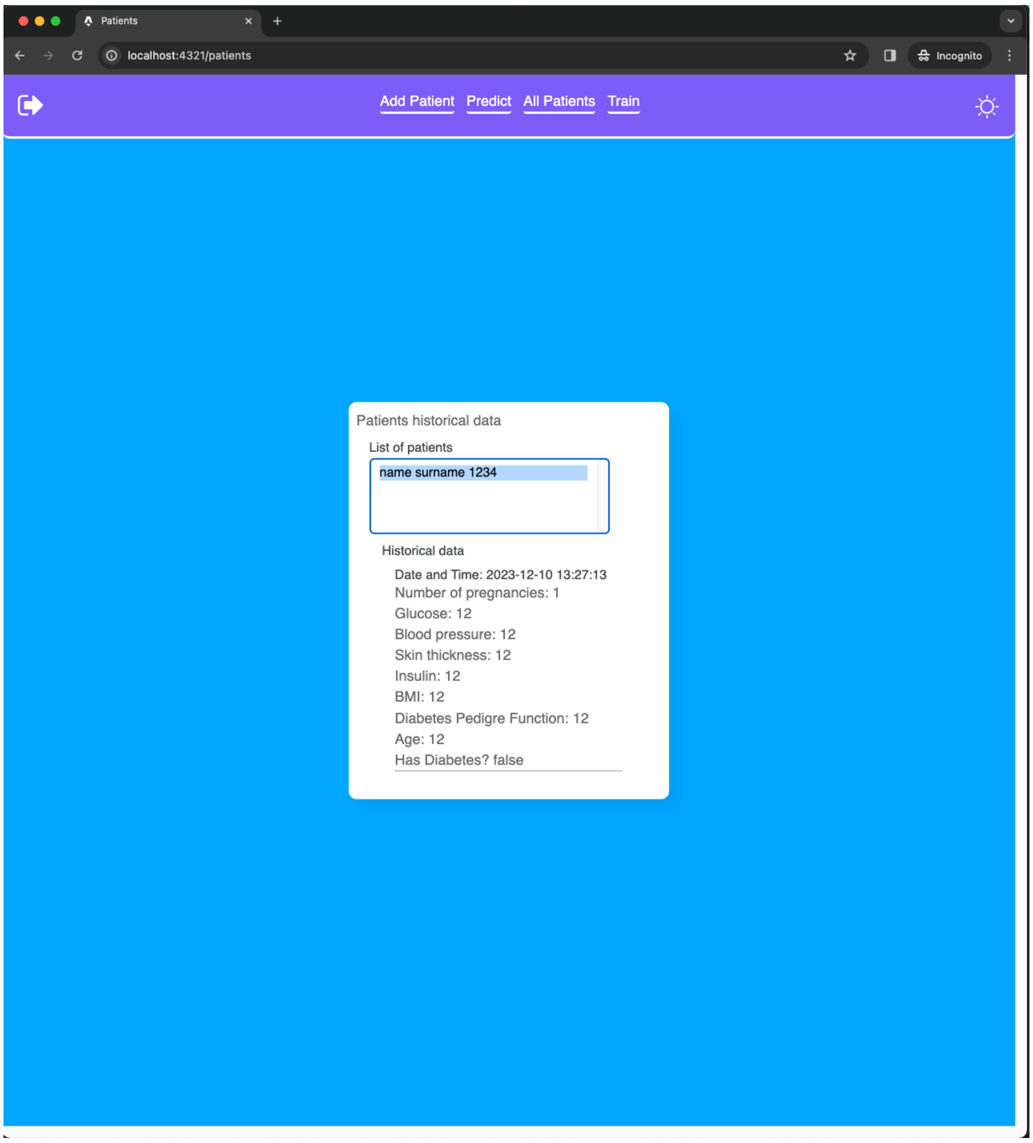
Age
12

PREDICT

Patient does have diabetes

Rysunek 4: Wybieramy pacjenta bądź pole puste, aby przetestować bez zapisywania danych do pacjenta

Nie ma możliwości nie wybrania pola z listy, wszystkie pola wymagają cyfr, nie ma możliwości wpisania innych znaków.



Patients historical data

List of patients

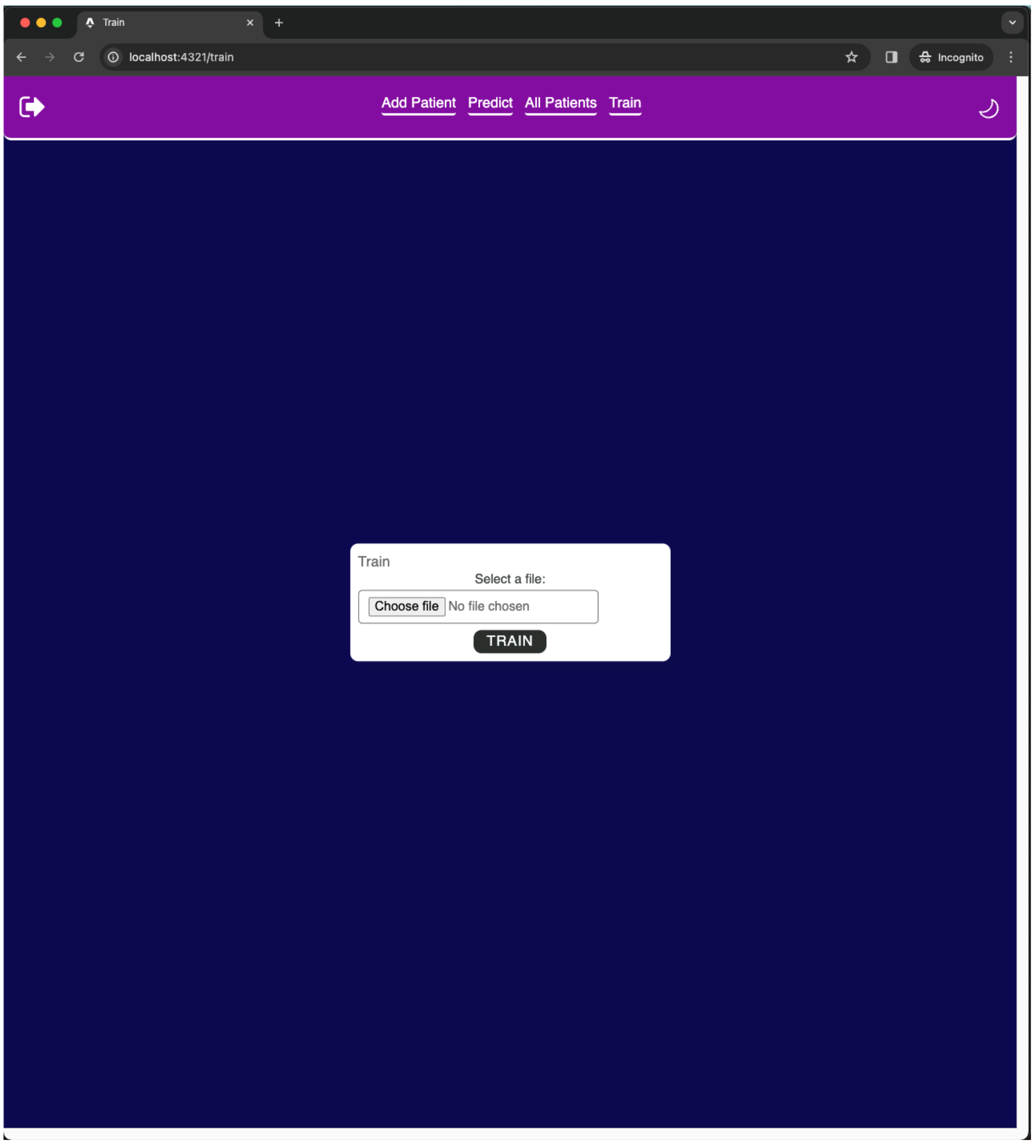
name surname 1234

Historical data

Date and Time: 2023-12-10 13:27:13
Number of pregnancies: 1
Glucose: 12
Blood pressure: 12
Skin thickness: 12
Insulin: 12
BMI: 12
Diabetes Pedigre Function: 12
Age: 12
Has Diabetes? false

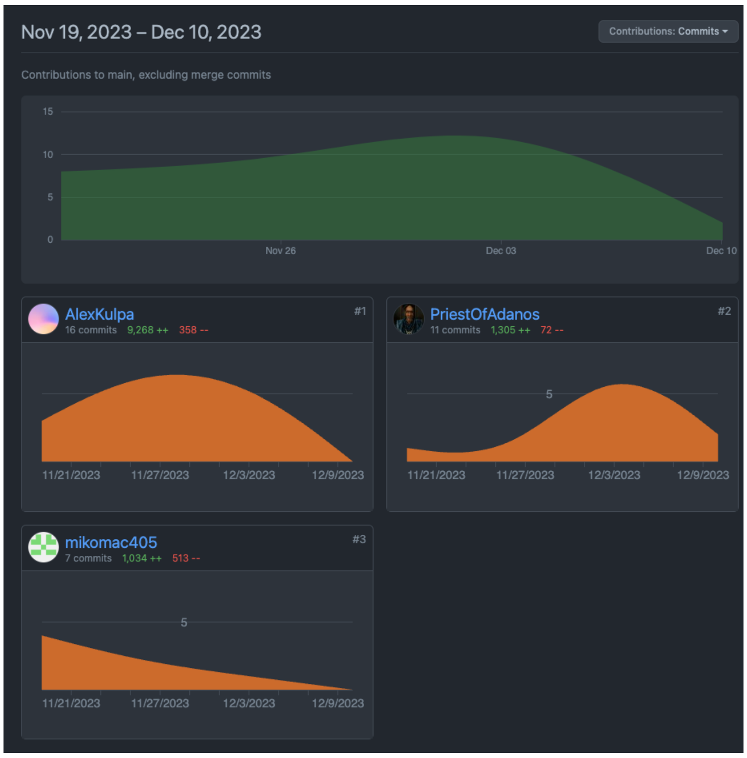
Rysunek 5: Wybierając z listy pacjenta możemy sprawdzić jego historyczne dane predykcji

Dane z predykcji są widoczne w postaci listy posortowanej chronologicznie.



Rysunek 6: Dodajemy plik csv z danymi do treningu

## 0.1 Instrukcja wdrożenia



Rysunek 7: Wybierając z listy pacjenta możemy sprawdzić jego historyczne dane predykcji

Podział pracy był następujący:

- Mikołaj Macura (mikomac405) - pisanie backend'u, baza danych i zabezpieczenia
- Paweł Habrzyk (PriestOfAdanos) - sieć neuronowa i jej dostrajanie + pomoc w backendzie
- Aleksander Kulpa (AlexKulpa) - pisanie frontend'u, odpowiedzialność za przesyłanie odpowiednich danych do backend'u

# Część II

## Opis działania

W tym skrypcie Pythona tworzona jest sieć neuronowa przy użyciu biblioteki `keras`. Sieć ta jest prosta, sekwencyjna, skonstruowana z kilku warstw.

Główna architektura sieci wygląda następująco:

1. Warstwa wejściowa: Warstwa `Dense` (gęstej) z 64 komórkami (neuronami) i funkcją aktywacji typu 'linear'. Ta warstwa przyjmuje dane wejściowe o określonym rozmiarze (rozmiarze cech wejściowych).

2. Druga warstwa: Warstwa robocza - `Dense` z 32 komórkami (neuronami) i funkcją aktywacji 'linear'.

3. Trzecia warstwa: Kolejna warstwa robocza - `Dense` z 16 komórkami (neuronami) i również z funkcją aktywacji 'linear'.

4. Warstwa wyjściowa: Końcowa warstwa - `Dense` z jednym neuronem i liniową funkcją aktywacji. Ta warstwa zwraca końcowy wynik prognozy.

Model sieci neuronowej jest trenowany przy użyciu optymalizatora 'adam' i funkcji straty jest 'mean_squared_error' (średni kwadrat błędu).

Natomiast w procesie tuningu hiperparametrów, stosowany jest algorytm `Random Search`. Wyszukuje on losowo kombinacje hiperparametrów, ocenia model dla każdej kombinacji i wybiera tę, która daje najmniejszą loss (stratę).

Podczas predykcji, model jest wczytywany z pliku, a następnie używany do prognozowania wartości wyjściowej na podstawie danych wejściowych. Prognoza jest następnie konwertowana do `int` i zwracana.

Wzory matematyczne:

- Adam: `https://arxiv.org/abs/1412.6980v8`

- Mean Squared Error:

$$MSE = \frac{1}{n}\sum(actual - prediction)^2$$

## Algorytmy

Reprezentacja matematyczna modelu

$$y = W_4 \cdot (W_3 \cdot (W_2 \cdot (W_1 \cdot x + b_1) + b_2) + b_3) + b_4 \tag{1}$$

### Biblioteki

- sklearn dla modelu regresji logistycznej

- keras dla modelu sekwencyjnego

- kerastuner do strojenia hiperparametrów.

**Modelowanie**

- Użycie Sequential z keras do tworzenia modelu sieci neuronowej.
- Konfiguracja warstw i neuronów w modelu (Dense layers).
- Kompilacja modelu z określonymi parametrami (np. optimizer='adam', loss='mean_squared_error'

**Przetwarzanie Danych**

- Użycie pandas do manipulacji i analizy danych.
- Podział danych na zestawy treningowe i testowe (train_test_split).

## Bazy danych

### Opis bazy i tabel

W projekcie użyta została baza danych SQLite Baza danych zawiera 3 tabele:

- diabetes - zawiera dane do szkolenia modelu
- patients - zawiera informacje o pacjencie oraz jego historię predykcji
- doctors - zawiara listę kont wraz z zaszyfrowanymi hasłami algorytmem bcrypt

Wszystkie table (łącznie z nazwą tabeli i kolumn) są zaszyfrowane algorytmemt md5 używając funkcji hashującej SHA512 dzięki bibliotece SQLCipher

### Operacje na bazie danych

- Wstawianie i pobieranie danych o cukrzycy.
- Rejestracja użytkowników i autentyfikacja.

## Implementacja systemu

### app.py

Tutaj uruchamiana jest aplikacja Flask. Prócz rozruchu w pliku zawarta jest definicja naszej sieci neuronowej oraz definijcje endpointów RestAPI.

### db_manager.py

W tym miejscu znajduje się klasa odpowiedzialna za inicjalizację oraz obsługę bazy danych, podczas działania programu.

Inicjalizacja bazy danych:

```
Otwórz i wczytaj bazę danych pod podaną ścieżką
if Podany plik nie istnieje lub nie jest bazą danych then
    Stwórz nową bazę danych pod podaną ścieżką i ją wczytaj
end
if Baza danych nie zawiera, którejś z tabel "diabetes", "doctors"oraz "patients" then
    Stwórz tabele "diabetes", "doctors"oraz "patients"zgodnie z ich definicjami oraz
      dodaj do tabeli "doctors"domyślnego administratora
end
```

```python
        def _setup_database_object(self) -> None:
          try:
              self.db = SqliteCipher(
                  dataBasePath=os.getenv("DB_PATH"),
                  checkSameThread=False,
                  password=os.getenv("DB_PASSWORD"),
              )
          except Exception as ex:
              logging.error(ex)

          for table_name in ["diabetes", "doctors", "patients"]:
              if not self.db.checkTableExist(table_name):
                  self._init_database()
                  break

    def _init_database(self) -> None:
        self.db.createTable(
            "diabetes",
            [
                ["pregnancies", "INT"],
                ["glucose", "REAL"],
                ["blood_pressure", "REAL"],
                ["skin_thickness", "REAL"],
                ["insulin", "REAL"],
                ["bmi", "REAL"],
                ["diabetes_pedigree_function", "REAL"],
                ["age", "INT"],
                ["outcome", "INT"],
            ],
            True,
            True,
        )

        self.db.createTable(
            "doctors",
            [
                ["first_name", "TEXT"],
                ["last_name", "TEXT"],
                ["email", "TEXT"],
                ["phone_number", "TEXT"],
                ["hashed_password", "TEXT"],
            ],
            True,
                True,
            )

            self.db.insertIntoTable(
                "doctors",
                [
                    "Senior",
                    "Registrar",
                    "senior_registrar@hospital.com",
                    "+48-111-222-333",
                    password_utils.get_hashed_password(os.getenv("
                       DEV_PASSWORD")),
                ],
                True,
            )

            self.db.createTable(
                "patients",
                [
                    ["PESEL", "TEXT"],
                    ["first_name", "TEXT"],
                    ["last_name", "TEXT"],
                    ["email", "TEXT"],
                    ["phone_number", "TEXT"],
                    ["historical_data", "BLOB"],
                ],
                True,
                True,
            )
```

### Elementy pomocnicze

Pliki **db_models.py**, **jwt_utils.py** oraz **password_utlis.py** zawierają modele oraz funkcje pomocnicze dla obsługi bazy danych oraz endpointów.

## Testy

## test_train_model_endpoint

- **Cel:** Testowanie endpointu odpowiedzialnego za trenowanie modelu uczenia maszynowego.
- **Proces:** Wysyłanie żądania POST na endpoint `/train` z plikiem `diabetes.csv`, zawierającym dane do treningu modelu. Używa tokena typu Bearer do autoryzacji.
- **Sprawdzenie:** Status odpowiedzi równy 200, co oznacza pomyślne przetworzenie żądania.
- **Wynik:** Wyświetla `"train 200"` po pomyślnym ukończeniu.

## get_login

- **Cel:** Uzyskanie tokenu dostępu do autoryzacji w innych testach.
- **Proces:** Wysyłanie żądania POST na endpoint `/login` z danymi użytkownika. Funkcja zwraca cały obiekt odpowiedzi.

## test_predict_diabetes_endpoint

- **Cel:** Testowanie endpointu do przewidywania cukrzycy.
- **Proces:** Wysyłanie żądania POST na endpoint `/predict` z danymi JSON zawierającymi informacje o pacjencie i jego parametrach zdrowotnych.
- **Sprawdzenie:** Status odpowiedzi równy 200.
- **Wynik:** Wyświetla `"predict: 200"` po pomyślnym ukończeniu.

## test_get_data_endpoint

- **Cel:** Testowanie endpointu odpowiedzialnego za pobieranie danych.
- **Proces:** Wysyłanie żądania GET na endpoint `/data`.
- **Sprawdzenie:** Status odpowiedzi równy 200.
- **Wynik:** Wyświetla `"data: 200"` po pomyślnym ukończeniu.

## test_signup_endpoint

- **Cel:** Testowanie endpointu do rejestracji użytkowników.
- **Proces:** Wysyłanie żądania POST na endpoint `/signup` z danymi nowego użytkownika.
- **Sprawdzenie:** Status odpowiedzi równy 200.
- **Wynik:** Wyświetla `"signup"`.

## Wykonanie Testów

- Skrypt uzyskuje token dostępu poprzez wywołanie `get_login()`.
- Następnie sekwencyjnie wykonuje `test_get_data_endpoint`, `test_predict_diabetes_endpoint` i `test_train_model_endpoint`, przekazując uzyskany token do autoryzacji.

Testy pozwalały na weryfikację czy zmiany wprowadzanie podczas rozwoju aplikacji były bespieczne z punktu widzenia api w sposób ciągły

# Pełen kod aplikacji

```
//test.py

import requests
port = 8000
def test_train_model_endpoint(token):
    headers = {
        'accept': 'application/json',
        'Authorization': f'Bearer {token}',
        # requests won't add a boundary if this header is set when you
           pass files=
        # 'Content-Type': 'multipart/form-data',
    }

    files = {
        'file': ('diabetes.csv', open('diabetes.csv', 'rb'), 'text/csv')
             ,
    }

    response = requests.post('http://localhost:8000/train', headers=
       headers, files=files)
    assert response.status_code == 200
    print("train 200")

def get_login():
    headers = {
    'accept': 'application/json',
    'Content-Type': 'application/x-www-form-urlencoded',
    }

    data = {
        'grant_type': '',
        'username': 'senior_registrar@hospital.com',
        'password': 'passwd',
        'scope': '',
        'client_id': '',
        'client_secret': '',
    }

    response = requests.post('http://localhost:8000/login', headers=
       headers, data=data)

    return response

def test_predict_diabetes_endpoint(token):
    headers = {
        'accept': 'application/json',
        'Authorization': f'Bearer {token}',
        'Content-Type': 'application/json',
    }

    json_data = {
        'patient_id': None,
        'input': {
            'pregnancies': 0,
            'glucose': 0,
            'blood_pressure': 0,
            'skin_thickness': 0,
            'insulin': 0,
            'bmi': 0,
            'diabetes_pedigree_function': 0,
            'age': 0,
        },
    }

    response = requests.post(f'http://localhost:{port}/predict', headers
        =headers, json=json_data)
    assert response.status_code == 200
    print("predict: 200")

def test_get_data_endpoint(token):
    headers = {
        'accept': 'text/html',
        'Authorization': f'Bearer {token}',
    }

    response = requests.get(f'http://localhost:{port}/data', headers=
        headers)
    assert response.status_code == 200
    print("data: 200")

def test_signup_endpoint(token):
    print("signup")
    url = f"http://localhost:{port}/signup"
    # Replace this with the appropriate user data for your endpoint
    user_data = {
        "email": "test@example.com",
        "password": "yourpassword"
    }
    response = requests.post(url, json=user_data)
    assert response.status_code == 200
token = get_login().json().get("access_token")
test_get_data_endpoint(token)
test_predict_diabetes_endpoint(token)
test_train_model_endpoint(token)

//app.py

import datetime
from typing import List
from fastapi import FastAPI, HTTPException, UploadFile, Depends
from fastapi.responses import HTMLResponse
from fastapi.middleware.cors import CORSMiddleware
import pandas as pd
from fastapi.security import OAuth2PasswordRequestForm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from dotenv import load_dotenv
```

```
import joblib
from kerastuner.tuners import RandomSearch
from keras_tuner import HyperModel
from fastapi import FastAPI, HTTPException, UploadFile, Depends
from fastapi.responses import HTMLResponse
from fastapi.middleware.cors import CORSMiddleware
import pandas as pd
from fastapi.security import OAuth2PasswordRequestForm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from dotenv import load_dotenv
import joblib
from keras.models import Sequential
from keras.layers import Dense
from db_manager import DatabaseManager
from keras.models import load_model

from jwt_utils import (
    create_access_token,
    create_refresh_token,
    get_current_user,
)
from db_models import (
    UserRegister,
    TokenSchema,
    User,
    Patient,
    PredictionInput,
    DiabetesHistoricalOutput,
)
from password_utils import verify_password

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

model_name = "model"

class MyHyperModel(HyperModel):
    def __init__(self, input_shape):
        self.input_shape = input_shape

    def build(self, hp):
        model = Sequential()
        model.add(Dense(units=hp.Int('units', min_value=32, max_value
            =512, step=32),
                                activation='relu', input_shape=self.input_shape)
                                    )
        model.add(Dense(1, activation='linear'))
        model.compile(optimizer='adam', loss='mean_squared_error')
        return model

@app.post("/train")
async def train_model(file: UploadFile, user: User = Depends(
    get_current_user)):
    db.insert_diabetes_data(file.file)

    df = db.select_all_diabetes_data()

    X = df.iloc[:, :-1]
    y = df.iloc[:, -1]
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.2, random_state=42
    )
    input_size = X_train.shape[1]  # Number of features

    model = Sequential()
    model.add(Dense(64, input_shape=(input_size,), activation='linear'))
    model.add(Dense(32, activation='linear'))
    model.add(Dense(16, activation='linear'))
    model.add(Dense(1, activation='linear'))

    model.compile(optimizer='adam', loss='mean_squared_error')

    input_shape = [X_train.shape[1]]  # Assuming X_train is your input
        data
    hypermodel = MyHyperModel(input_shape=input_shape)


    tuner = RandomSearch(
    hypermodel,
    objective='val_loss',
    max_trials=5,
    executions_per_trial=3,
    directory='my_dir',
    project_name='hparam_tuning'
    )

    tuner.search(X_train, y_train, epochs=10, validation_data=(X_test,
        y_test))
    best_model = tuner.get_best_models(num_models=1)[0]
    best_model.save(model_name)

    return {"message": "Model trained and saved successfully"}


@app.post("/predict")
async def predict_diabetes(
    input_data: PredictionInput, user: User = Depends(get_current_user)
):

    try:
        model = load_model(model_name)
```

```python
    except FileNotFoundError:
        raise HTTPException(
            status_code=500, detail="Model not found. Please train the
                model first."
        )
    i_data  = input_data.dict()['input']
    input_df = pd.DataFrame([i_data])
    prediction = model.predict(input_df)
    prediction_native_type = int(prediction[0])  # or float, as
        appropriate
    if input_data.patient_id is not None:
        db.add_historical_data_to_patient(
            patient_id=input_data.patient_id,
            output=DiabetesHistoricalOutput(
                pregnancies=input_data.input.pregnancies,
                glucose=input_data.input.glucose,
                blood_pressure=input_data.input.blood_pressure,
                skin_thickness=input_data.input.skin_thickness,
                insulin=input_data.input.insulin,
                bmi=input_data.input.bmi,
                diabetes_pedigree_function=input_data.input.
                    diabetes_pedigree_function,
                age=input_data.input.age,
                prediction=bool(prediction_native_type),
                created_at=datetime.datetime.today().timestamp(),
            ),
        )

    return {"prediction": prediction_native_type}




@app.get("/data", response_class=HTMLResponse)
async def get_data_from_database(user: User = Depends(get_current_user))
    :
    return db.select_all_diabetes_data().to_html(notebook=True)


@app.post("/signup", summary="Create new user")
async def create_user(data: UserRegister):
    return db.create_doctor(data)


@app.post(
    "/login",
    summary="Create access and refresh tokens for user",
    response_model=TokenSchema,
)
async def login(form_data: OAuth2PasswordRequestForm = Depends()):
    user = db.get_doctor(form_data.username)
    print(form_data.username)
    if user is None:
        raise HTTPException(status_code=400, detail="Incorrect email or
            password")

    hashed_pass = user.hashed_password
    print(hashed_pass)
    if not verify_password(form_data.password, hashed_pass):
        raise HTTPException(
            status_code=400,
            detail="Incorrect email or password",
        )

    return {
        "access_token": create_access_token(user.email),
        "refresh_token": create_refresh_token(user.email),
    }


@app.get("/me", summary="Get details of currently logged in user",
    response_model=User)
async def get_me(user: User = Depends(get_current_user)):
    return user


@app.post("/patient", summary="Adds new patient to database",
    response_model=Patient)
async def create_patient(patient_data: Patient, user: User = Depends(
    get_current_user)):
    return db.create_patient(patient_data)


@app.get(
    "/patient", summary="Get list of patients with data", response_model
        =List[Patient]
)
async def get_users(user: User = Depends(get_current_user)):
    return db.get_patients()

//db_manager.py

import datetime
from typing import Optional, BinaryIO

import pandas as pd
from fastapi import HTTPException
from fastapi.responses import Response
from pysqlitecipher.sqlitewrapper import SqliteCipher
import os
import logging

import password_utils
from db_models import UserOut, UserRegister, Patient,
    DiabetesHistoricalOutput


class SingletonMeta(type):
    _instances = {}

```

```python
    def __call__(cls, *args, **kwargs):
        if cls not in cls._instances:
            instance = super().__call__(*args, **kwargs)
            cls._instances[cls] = instance
        return cls._instances[cls]


class DatabaseManager(metaclass=SingletonMeta):
    def __init__(self) -> None:
        self.db: Optional[SqliteCipher] = None
        self._setup_database_object()

    def _setup_database_object(self) -> None:
        try:
            self.db = SqliteCipher(
                dataBasePath=os.getenv("DB_PATH"),
                checkSameThread=False,
                password=os.getenv("DB_PASSWORD"),
            )
        except Exception as ex:
            logging.error(ex)

        for table_name in ["diabetes", "doctors", "patients"]:
            if not self.db.checkTableExist(table_name):
                self._init_database()
                break

    def _init_database(self) -> None:
        self.db.createTable(
            "diabetes",
            [
                ["pregnancies", "INT"],
                ["glucose", "REAL"],
                ["blood_pressure", "REAL"],
                ["skin_thickness", "REAL"],
                ["insulin", "REAL"],
                ["bmi", "REAL"],
                ["diabetes_pedigree_function", "REAL"],
                ["age", "INT"],
                ["outcome", "INT"],
            ],
            True,
            True,
        )

        self.db.createTable(
            "doctors",
            [
                ["first_name", "TEXT"],
                ["last_name", "TEXT"],
                ["email", "TEXT"],
                ["phone_number", "TEXT"],
                ["hashed_password", "TEXT"],
            ],
            True,
                True,
        )

        self.db.insertIntoTable(
            "doctors",
            [
                "Senior",
                "Registrar",
                "senior_registrar@hospital.com",
                "+48-111-222-333",
                password_utils.get_hashed_password(os.getenv("
                    DEV_PASSWORD")),
            ],
            True,
        )

        self.db.createTable(
            "patients",
            [
                ["PESEL", "TEXT"],
                ["first_name", "TEXT"],
                ["last_name", "TEXT"],
                ["email", "TEXT"],
                ["phone_number", "TEXT"],
                ["historical_data", "BLOB"],
            ],
            True,
            True,
        )

    def select_all_diabetes_data(self) -> pd.DataFrame:
        columns, data = self.db.getDataFromTable("diabetes", True, True)
        return pd.DataFrame(data=data, columns=columns)

    def insert_diabetes_data(self, csv_file: BinaryIO) -> None:
        df = pd.read_csv(csv_file)
        df.columns = [
            "pregnancies",
            "glucose",
            "blood_pressure",
            "skin_thickness",
            "insulin",
            "bmi",
            "diabetes_pedigree_function",
            "age",
            "outcome",
        ]

        for i in df.index:
            self.db.insertIntoTable(
                "diabetes", [df[col_name][i] for col_name in df.columns
                    ], True
            )

    def dump_diabetes_data_to_csv(self) -> None:
```

```python
        pass

    def get_doctor(self, email: str) -> Optional[UserOut]:
        columns, data = self.db.getDataFromTable("doctors", True)
        for d in data:
            if d[3] == email:
                return UserOut(
                    first_name=d[1],
                    last_name=d[2],
                    email=d[3],
                    phone_number=d[4],
                    hashed_password=d[5],
                )
        return None

    def create_doctor(self, user: UserRegister):
        columns, data = self.db.getDataFromTable("doctors", True)
        for d in data:
            if d[3] == user.email or d[4] == user.phone_number:
                raise HTTPException(
                    status_code=404,
                    detail=f"Doctor with email {user.email} or phone number {user.phone_number} already exists.",
                )
        self.db.insertIntoTable(
            "doctors",
            [
                user.first_name,
                user.last_name,
                user.email,
                user.phone_number,
                password_utils.get_hashed_password(user.password),
            ],
            True,
        )
        return Response(
            f"Successfully created a doctor {user.first_name} {user.last_name}"
        )

    def create_patient(self, patient: Patient):
        columns, data = self.db.getDataFromTable("patients", True)
        for d in data:
            if (
                d[1] == patient.PESEL
                or d[4] == patient.email
                or d[5] == patient.phone_number
            ):
                raise HTTPException(
                    status_code=404,
                    detail=f"Patient with PESEL {patient.PESEL} or email {patient.email} or phone number {patient.phone_number} already exists.",
                )
        self.db.insertIntoTable(
            "patients",
            [
                patient.PESEL,
                patient.first_name,
                patient.last_name,
                patient.email,
                patient.phone_number,
                "".encode(),
            ],
            True,
        )

        return patient

    def add_historical_data_to_patient(
        self, patient_id: int, output: DiabetesHistoricalOutput
    ):
        columns, data = self.db.getDataFromTable("patients", True)
        exists = False
        patient_data = None
        print("hm")
        for d in data:
            if d[0] == patient_id:
                exists = True
                patient_data = d[6]
                break
        if not exists:
            raise HTTPException(
                status_code=404,
                detail=f"Patient with ID {patient_id} does not exists.",
            )
        print("got patient")
        patient_data = patient_data.decode()
        print("patient current data:", patient_data)
        data_str = (
            f"{output.pregnancies},{output.glucose},{output.
                blood_pressure},{output.skin_thickness},"
            f"{output.insulin},{output.bmi},{output.
                diabetes_pedigree_function},{output.age},"
            f"{output.prediction},{output.created_at.timestamp()};"
        )

        patient_data += data_str

        print("patient current data:", patient_data)
        self.db.updateInTable(
            "patients", patient_id, "historical_data", patient_data.
                encode(), True
        )

    def get_patients(self):
        columns, data = self.db.getDataFromTable("patients", True)
        patients = []
        for d in data:
            historical_data = []

            for data_point in d[6].decode().split(";")[:-1]:
                data_point = data_point.split(",")
                historical_data.append(
                    DiabetesHistoricalOutput(
                        pregnancies=data_point[0],
                        glucose=data_point[1],
                        blood_pressure=data_point[2],
                        skin_thickness=data_point[3],
                        insulin=data_point[4],
                        bmi=data_point[5],
                        diabetes_pedigree_function=data_point[6],
                        age=data_point[7],
                        prediction=data_point[8],
                        created_at=data_point[9],
                    )
                )

            patients.append(
                Patient(
                    id=d[0],
                    PESEL=d[1],
                    first_name=d[2],
                    last_name=d[3],
                    email=d[4],
                    phone_number=d[5],
                    historical_data=historical_data,
                )
            )
        return patients
//db_models.py

from datetime import datetime
from typing import List, Optional

from pydantic import BaseModel


class DiabetesPredictionInput(BaseModel):
    pregnancies: int
    glucose: float
    blood_pressure: float
    skin_thickness: float
    insulin: float
    bmi: float
    diabetes_pedigree_function: float
    age: int


class DiabetesHistoricalOutput(DiabetesPredictionInput):
    prediction: bool
    created_at: datetime


class Patient(BaseModel):
    id: Optional[int]
    PESEL: str
    first_name: str
    last_name: str
    email: str
    phone_number: str
    historical_data: Optional[List[DiabetesHistoricalOutput]]


class PredictionInput(BaseModel):
    patient_id: Optional[int]
    input: DiabetesPredictionInput


class User(BaseModel):
    first_name: str
    last_name: str
    email: str
    phone_number: str


class UserOut(User):
    hashed_password: str


class UserRegister(User):
    password: str


class TokenSchema(BaseModel):
    access_token: str
    refresh_token: str


class TokenPayload(BaseModel):
    sub: str = None
    exp: int = None

jwt_utils.py

from datetime import datetime, timedelta
from typing import Union, Any, Optional

from dotenv import load_dotenv
from fastapi import HTTPException, Depends
from fastapi.security import OAuth2PasswordBearer
from jose import jwt

import os

from pydantic import ValidationError

from db_manager import DatabaseManager
from db_models import TokenPayload, UserOut, User

```

```
load_dotenv()

ACCESS_TOKEN_EXPIRE_MINUTES = 120  # 120 minutes
REFRESH_TOKEN_EXPIRE_MINUTES = 60 * 24 * 7  # 7 days
ALGORITHM = "HS256"
JWT_SECRET_KEY = os.getenv("JWT_SECRET_KEY")  # should be kept secret
JWT_REFRESH_SECRET_KEY = os.getenv("JWT_REFRESH_SECRET_KEY")  # should
    be kept secret


db = DatabaseManager()

reuseable_oauth = OAuth2PasswordBearer(tokenUrl="/login", scheme_name="
    JWT")


def create_access_token(subject: Union[str, Any]) -> str:
    expires_delta = datetime.utcnow() + timedelta(minutes=
        ACCESS_TOKEN_EXPIRE_MINUTES)

    to_encode = {"exp": expires_delta, "sub": str(subject)}
    encoded_jwt = jwt.encode(to_encode, JWT_SECRET_KEY, ALGORITHM)
    return encoded_jwt


def create_refresh_token(subject: Union[str, Any]) -> str:
    expires_delta = datetime.utcnow() + timedelta(minutes=
        REFRESH_TOKEN_EXPIRE_MINUTES)

    to_encode = {"exp": expires_delta, "sub": str(subject)}
    encoded_jwt = jwt.encode(to_encode, JWT_REFRESH_SECRET_KEY,
        ALGORITHM)
    return encoded_jwt


async def get_current_user(token: str = Depends(reuseable_oauth)) ->
    User:
    try:
        payload = jwt.decode(token, JWT_SECRET_KEY, algorithms=[
            ALGORITHM])
        token_data = TokenPayload(**payload)

        if datetime.fromtimestamp(token_data.exp) < datetime.now():
            raise HTTPException(
                status_code=401,
                detail="Token expired",
                headers={"WWW-Authenticate": "Bearer"},
            )
    except (jwt.JWTError, ValidationError):
        raise HTTPException(
            status_code=403,
            detail="Could not validate credentials",
            headers={"WWW-Authenticate": "Bearer"},
        )

    user: Optional[UserOut] = db.get_doctor(token_data.sub)

    if user is None:
        raise HTTPException(
            status_code=404,
            detail="Could not find user",
        )

    return user

//password_utils.py

from passlib.context import CryptContext

password_context = CryptContext(schemes=["bcrypt"], deprecated="auto")


def get_hashed_password(password: str) -> str:
    return password_context.hash(password)


def verify_password(password: str, hashed_pass: str) -> bool:
    return password_context.verify(password, hashed_pass)

//requierments.txt

absl-py==2.0.0
annotated-types==0.6.0
anyio==3.7.1
astunparse==1.6.3
bcrypt==4.1.1
cachetools==5.3.2
certifi==2023.11.17
cffi==1.16.0
charset-normalizer==3.3.2
click==8.1.7
cryptography==41.0.7
dm-tree==0.1.8
ecdsa==0.18.0
fastapi==0.104.1
flatbuffers==23.5.26
gast==0.5.4
google-auth==2.24.0
google-auth-oauthlib==1.1.0
google-pasta==0.2.0
grpcio==1.59.3
h11==0.14.0
h5py==3.10.0
httpcore==1.0.2
httpx==0.25.2
idna==3.4
iniconfig==2.0.0
joblib==1.3.2
keras==2.15.0
keras-tuner==1.4.6
```

```
kt-legacy==1.0.5
libclang==16.0.6
Markdown==3.5.1
markdown-it-py==3.0.0
MarkupSafe==2.1.3
mdurl==0.1.2
ml-dtypes==0.2.0
namex==0.0.7
numpy==1.26.2
oauthlib==3.2.2
onetimepad==1.4
opt-einsum==3.3.0
packaging==23.2
pandas==2.1.3
passlib==1.7.4
pluggy==1.3.0
protobuf==4.23.4
pyasn1==0.5.1
pyasn1-modules==0.3.0
pycparser==2.21
pydantic==2.5.1
pydantic_core==2.14.3
Pygments==2.17.2
pysqlitecipher==0.22
pytest==7.4.3
python-dateutil==2.8.2
python-dotenv==1.0.0
python-jose==3.3.0
python-multipart==0.0.6
pytz==2023.3.post1
requests==2.31.0
requests-oauthlib==1.3.1
rich==13.7.0
rsa==4.9
scikit-learn==1.3.2
scipy==1.11.4
six==1.16.0
sniffio==1.3.0
starlette==0.27.0
tensorboard==2.15.1
tensorboard-data-server==0.7.2
tensorflow==2.15.0
tensorflow-estimator==2.15.0
tensorflow-io-gcs-filesystem==0.34.0
tensorflow-macos==2.15.0
termcolor==2.4.0
threadpoolctl==3.2.0
typing_extensions==4.8.0
tzdata==2023.3
urllib3==2.1.0
uvicorn==0.24.0.post1
Werkzeug==3.0.1
wrapt==1.14.1
pysqlitecipher==0.22
python-dotenv==1.0.0
```

```
python-multipart==0.0.6
passlib==1.7.4
python-jose==3.3.0

//global.css

*{
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  font-family: "Inter", sans-serif;
}

:root {
  --color-lightest: #f9fdfe;
  --color-gray-light: #cdcfcf;
  --color-gray-medium: #686a69;
  --color-gray-dark: #414643;
  --color-darkest: #2a2f2c;
}

    .input-group {
        margin-top: 0.25rem;
        padding: 0.5rem 1rem;
    }

    .contact-form label {
        display: block;
        color: var(--color-gray-dark);
        font-family: Lato, sans-serif;
        font-size: 90%;
        line-height: 1.125;
    }

    .group-label {
        display: inline-block;
        margin-right: 1rem;
        font-size: 90%;
    }

    .contact-form label.inline {
        display: inline-block;
        margin-left: 0.25rem;
    }

    .contact-form input,
    .contact-form select,
    .contact-form textarea{
        display: block;
        margin-top: 0.25rem;
        padding: 0.5rem 0.75rem;
        border-radius: 5px;
        border: 1px solid var(--color-gray-medium);
        width: 300px;
        font-family: "Open Sans", sans-serif;
```

```css
        font-size: 1rem;
        transition: 150ms border-color linear;
    }


    .contact-form input:focus,
    .contact-form input:active {
        border-color: var(--color-gray-medium);
    }

    button.glob{

        display: block;
        margin: 0.5rem 1rem 0;
        padding: 0 1rem 0.125rem;
        border-radius: 10px;
        background-color: var(--color-darkest);
        border: 0;
        color: var(--color-lightest);
        font-family: lato, sans-serif;
        font-size: 100%;
        letter-spacing: 0.05em;
        line-height: 1.5;
        text-transform: uppercase;
        transition: 150ms all linear;
    }

    .contact-form button:hover,
    .contact-form button:active,
    .contact-form button:focus {
        background: var(--color-darkest);
        cursor: pointer;
    }

    .result {
        margin: 10px;
    }

    .space {
        margin: 10px;
        height: 5px;
        background: gray;
        border-radius: 10px;
    }


a {
  text-decoration: none;
  color: white;
}
header {
  text-align: center;
  position: fixed;
  height: 80px;
  width: 100%;
  z-index: 100;
  padding: 20px 20px;
  border-style: none none solid none ;
  border-radius: 0px 0px 10px 10px;
  border-color: white;
  background-color: rgba(249, 18, 242, 0.5);
}

.UploadFile {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    width: 400px;
    background: white;
    border-radius: 10px;
    box-shadow: 10px 10px 15px rgba(0,0,0,0.05);
    padding: 10px;
    }


.hamburger {
  display: inline-block;
  width: fit-content;
  block-size: fit-content;
  padding-right: 20px;
  cursor: pointer;
}

.hamburger .line {
  display: block;
  width: 40px;
  height: 5px;
  margin-bottom: 10px;
  background-color: #ff9776;
}

body{
  margin: 0;
  padding: 0;
  height: auto;
  overflow: hidden;
  background: var(--color-lightest);
  color: var(--color-gray-medium);
  font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto,
      Helvetica,
  Arial, sans-serif, "Apple Color Emoji", "Segoe UI Emoji", "Segoe UI
      Symbol";
  font-size: 18px;
  line-height: 1.45;
}

html
{
    background-color: rgb(0, 166, 255);
```

```
}

html.dark {
  background-color: #0d0950;

}

//checkToken.js

let tokenToSend = `${localStorage.getItem("SavedToken")}`;

      fetch('http://127.0.0.1:8000/me', {
        credentials: "same-origin",
        method: 'GET',
        headers: {
          'accept': 'application/json',
          'Authorization': tokenToSend
        },

      }).then(response =>
       {
        if (response.status == 401) {
              response.json().then(data => {
                console.log(tokenToSend);
                window.location.href = "/login-page";
              });
        }});

//train.astro

---
import BaseLayout from '../layouts/BaseLayout.astro';
const pageTitle = "Train";
---

<script src="../scripts/checkToken.js"/>

<BaseLayout pageTitle={pageTitle}>
    <label >Train</label>
<div>
    <form id="trainForm">
    <section class="contact-form">
    <label for="csvToTrain">Select a file:</label>
    <input type="file" id="csvToTrain" name="myfile" accept="text/csv"/>
    <button class="glob">Train</button>
    <section/>
</form>
</div>
</BaseLayout>


<script>
    function sendDataToTrain(event) {
        event.preventDefault();

        let tokenToSend = `${localStorage.getItem("SavedToken")}`;

        let fileInput = (document.getElementById("csvToTrain") as
            HTMLInputElement);
    let file = fileInput.files[0];

    let formData = new FormData();
    formData.append('file', file, file.name);

      fetch('http://127.0.0.1:8000/train', {
        credentials: "same-origin",
        method: 'POST',
        headers: {
          'accept': 'application/json',
          'Authorization': tokenToSend
        },
        body: formData
      }).then(response =>
       {
        if (response.status == 200) {


        }
       });
    }

    var button = document.querySelector(".glob");
  button.addEventListener('click', sendDataToTrain);
</script>


<style>
    :root {
  --color-lightest: #f9fdfe;
  --color-gray-light: #cdcfcf;
  --color-gray-medium: #686a69;
  --color-gray-dark: #414643;
  --color-darkest: #2a2f2c;
}

div {
    text-align: center;
}

label {
    text-align: center;
}


button{
    display: inline-block;
}
</style>

//signup-page.astro
```

```

---
const pageTitle = "Sign Up";
import '../styles/global.css';
---
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" type="image/svg+xml" href="/favicon.svg" />
    <meta name="viewport" content="width=device-width" />
    <meta name="generator" content={Astro.generator} />
    <title>{pageTitle}</title>
  </head>

  <body>
<div class="UploadFile">

        <h1 style="color:black;">Sign Up</h1>
        <section class="contact-form">
        <form method="post">
          <div class="txt_field">
            <input class="inptClass2" id="first_name" name="first_name"
               type="text" required>
            <span></span>
            <label>Name</label>
          </div>
          <div class="txt_field">
            <input class="inptClass3" id="last_name" name="last_name"
               type="text" required>
            <span></span>
            <label>Surname</label>
          </div>
          <div class="txt_field">
            <input id="email" name="email" type="text" required>
            <span></span>
            <label>Email</label>
          </div>
          <div class="txt_field">
            <input class="inptClass5" id="phone_number" name="
               phone_number" type="number"  type="number" min="0" step="
               1" max="999999999" required>
            <span></span>
            <label>Phone Numer</label>
          </div>
          <div class="txt_field">
            <input id="password" name="password" type="text" required>
            <span></span>
            <label>Password</label>
          </div>
          <button class="glob" type="submit">Sign Up</button>
          <div class="signup_link">
             <a href="/login-page">Back to login page</a>
          </div>
        </form>
    </section>
```

```
      </div>

      <script>
        function handleFormSubmit(event) {
          event.preventDefault();

          const data = new FormData(event.target);
          const formJSON = {};

          data.forEach((value, key) => {
            formJSON[key] = /^\d+$/.test(value) ? String(value) : value;
          });
          console.log(JSON.stringify(formJSON, null, 2));

          fetch('http://127.0.0.1:8000/signup', {
            credentials: "same-origin",
            method: 'POST',
            headers: {
              'accept': 'application/json',
              'Content-Type': 'application/json'
            },
            body: JSON.stringify(formJSON, null, 2)
          })
           .then(response => response.json())
           .then(response => console.log(JSON.stringify(response)));
        }
        document.querySelector(".inptClass2").addEventListener("keypress", function (evt) {
    var charCode1 = (evt.which) ? evt.which : evt.keyCode;
    if ((charCode1 >= 65 && charCode1 <= 90) || (charCode1 >= 97 && charCode1 <= 122)) {
        // Allow the key press
    } else {
        // Prevent the key press if it doesn't meet the criteria
        evt.preventDefault();
    }
});
document.querySelector(".inptClass3").addEventListener("keypress", function (evt) {
    var charCode2 = (evt.which) ? evt.which : evt.keyCode;
    if ((charCode2 >= 65 && charCode2 <= 90) || (charCode2 >= 97 && charCode2 <= 122)) {
        // Allow the key press
    } else {
        // Prevent the key press if it doesn't meet the criteria
        evt.preventDefault();
    }
});

        document.querySelector(".inptClass5").addEventListener("keypress", function (evt) {
    if (evt.which != 8 && evt.which != 0 && evt.which < 48 || evt.which > 57)
    {
        evt.preventDefault();
    }
});
        const form = document.querySelector('.contact-form');
        form.addEventListener('submit', handleFormSubmit);
      </script>



</body>
</html>

<style>

    .center{
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  width: 400px;
  background: white;
  border-radius: 10px;
  box-shadow: 10px 10px 15px rgba(0,0,0,0.05);
}
.center h1{
  text-align: center;
  padding: 20px 0;
  border-bottom: 1px solid silver;

}
.center form{
  padding: 0 40px;
  box-sizing: border-box;
}
form .txt_field{
  position: relative;
  border-bottom: 2px solid #adadad;
  margin: 30px 0;
}
.txt_field input{
  width: 100%;
  padding: 0 5px;
  height: 40px;
  font-size: 16px;
  border: none;
  background: none;
  outline: none;
}
.txt_field label{
  position: absolute;
  top: 50%;
  left: 5px;
  color: #adadad;
  transform: translateY(-50%);
  font-size: 16px;
  pointer-events: none;
  transition: .5s;
}
.txt_field span::before{
  content: '';
  position: absolute;
  top: 40px;
  left: 0;
  width: 0%;
  height: 2px;
  background: #2691d9;
  transition: .5s;
}
.txt_field input:focus ~ label,
.txt_field input:valid ~ label{
  top: -5px;
  color: #2691d9;
}
.txt_field input:focus ~ span::before,
.txt_field input:valid ~ span::before{
  width: 100%;
}
.pass{
  margin: -5px 0 20px 5px;
  color: #a6a6a6;
  cursor: pointer;
}
.pass:hover{
  text-decoration: underline;
}
input[type="submit"]{
  width: 100%;
  height: 50px;
  border: 1px solid;
  background: #2691d9;
  border-radius: 25px;
  font-size: 18px;
  color: #e9f4fb;
  font-weight: 700;
  cursor: pointer;
  outline: none;
}
input[type="submit"]:hover{
  border-color: #2691d9;
  transition: .5s;
}
.signup_link{
    float: right;
  margin: 10px 20px;
  text-align: center;
  font-size: 16px;
  color: #666666;
}
.signup_link a{
  color: #2691d9;
  text-decoration: none;
```

```
}
.signup_link a:hover{
  text-decoration: underline;
}
</style>

//patients.astro

---
import BaseLayout from '../layouts/BaseLayout.astro';
const pageTitle = "Patients";
---

<script src="../scripts/checkToken.js"/>
<script>


</script>


<BaseLayout pageTitle={pageTitle}>
    <label >Patients historical data</label>
    <section class="contact-form" >

        <div class="input-group">
            <label for="pregnancies">List of patients</label>
        <select class="input-group" id="selectedValue"  size="4">

        </select>
        <div/>

        <div class="input-group">
            <label for="pregnancies">Historical data</label>
        <div id="histData" class="input-group">

        </div>
        <div/>

        <section/>
        <script>

    let dataGlob;
    let tokenToSend = `${localStorage.getItem("SavedToken")}`;

    fetch('http://127.0.0.1:8000/patient', {
      credentials: "same-origin",
      method: 'GET',
      headers: {
        'accept': 'application/json',
        'Authorization': tokenToSend
      },

    }).then(response =>
    {

```

```
   response.json().then(data => {
       const select = document.querySelector('select');
       console.log(data);
       dataGlob = data;
               console.log(dataGlob);
           for (var i = 0; i < data.length; i++){
               let newOption = new Option(`${data[i]["first_name"]} ${data[i]["last_name"]} ${data[i]["PESEL"]}`,`${data[i]["id"]}`);

               select.add(newOption,undefined)}
       });
});
           function valueChanges() {
           var e = document.getElementById("selectedValue");
           var value = parseInt(e.value);
           const historicData = document.getElementById("histData");
           historicData.innerHTML = '';
           var iDiv;
           var label;
           var dateAndTime;
           var pregn;
           var gluc;
           var blood;
           var skin;
           var insulin;
           var bmi;
           var dpf;
           var age;
           var pred;
           var hrTag;
           var dateAndTimeArray = {};
                       for (var i = 0; i < dataGlob[value]["historical_data"].length; i++){
                           pregn = document.createElement('div');
                           gluc = document.createElement('div');
                           blood = document.createElement('div');
                           skin = document.createElement('div');
                           insulin = document.createElement('div');
                           bmi = document.createElement('div');
                           dpf = document.createElement('div');
                           age = document.createElement('div');
                           pred = document.createElement('div');
                           label = document.createElement('label');
                           hrTag = document.createElement('hr');
                          console.log(dataGlob[value]["historical_data"][i]);
                          dateAndTime = dataGlob[value]["historical_data"][i]["created_at"];
                          dateAndTimeArray = dateAndTime.split("T");
                          console.log(dateAndTimeArray);
                          historicData.appendChild(label).innerText = `Date and Time: ${dateAndTimeArray[0]} ${dateAndTimeArray[1].substring(0,8)}`;
                          historicData.appendChild(pregn).innerText = `
```

```
                                Number of pregnancies: ${dataGlob[value]["
                                historical_data"][i]["pregnancies"]}`;
                            historicData.appendChild(gluc).innerText = `
                                Glucose: ${dataGlob[value]["
                                historical_data"][i]["glucose"]}`;
                            historicData.appendChild(blood).innerText = `
                                Blood pressure: ${dataGlob[value]["
                                historical_data"][i]["blood_pressure"]}`;
                            historicData.appendChild(skin).innerText = `
                                Skin thickness: ${dataGlob[value]["
                                historical_data"][i]["skin_thickness"]}`;
                            historicData.appendChild(insulin).innerText =
                                 `Insulin: ${dataGlob[value]["
                                historical_data"][i]["insulin"]}`;
                            historicData.appendChild(bmi).innerText = `
                                BMI: ${dataGlob[value]["historical_data"][
                                i]["bmi"]}`;
                            historicData.appendChild(dpf).innerText = `
                                Diabetes Pedigre Function: ${dataGlob[
                                value]["historical_data"][i]["
                                diabetes_pedigree_function"]}`;
                            historicData.appendChild(age).innerText = `
                                Age: ${dataGlob[value]["historical_data"][
                                i]["age"]}`;
                            historicData.appendChild(pred).innerText = `
                                Has Diabetes? ${dataGlob[value]["
                                historical_data"][i]["prediction"]}`;
                            historicData.appendChild(hrTag);
                        }
            }

            document.getElementById('selectedValue').addEventListener('
                change',valueChanges);
            valueChanges();
        </script>
</BaseLayout>

<style>
    #histData{
        max-height: 800px;
        overflow: auto;
        }
</style>

//login-page.astro

---
import Snack from '../components/Snack.astro';
const pageTitle = "Login";
import '../styles/global.css';
---
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" type="image/svg+xml" href="/favicon.svg" />
    <meta name="viewport" content="width=device-width" />
    <title>{pageTitle}</title>
  </head>

  <body>
<div class="UploadFile">

        <h1 style="color:black;">Login</h1>
        <section class="contact-form">
        <form method="post">
          <div class="txt_field">
            <input id="email" name="username" type="email" required>
            <span></span>
            <label>Email</label>
          </div>
          <div class="txt_field">
            <input  id="password" name="password"  type="password"
               required>
            <span></span>
            <label>Password</label>
          </div>
          <button class="glob" type="submit">Login</button>

        </form>
        </section>

        <Snack id="snackbar"></Snack>

<script>
  function handleFormSubmit(event) {
    event.preventDefault();

    const data = new FormData(event.target);

    let username_to_change = data.get("username");
    let final_username = username_to_change.replace('@','%40');

    let body_tosent = `grant_type=&username=${final_username}&password=$
       {data.get("password")}&scope=&client_id=&client_secret=`;

    fetch('http://127.0.0.1:8000/login', {
      credentials: "same-origin",
      method: 'POST',
      headers: {
        'accept': 'application/json',
        'Content-Type': 'application/x-www-form-urlencoded'
      },
      body: body_tosent
    }).then(response =>
     {
      if (response.status == 200) {
            response.json().then(data => {
                localStorage.setItem("SavedToken", "Bearer " + data["
                   access_token"]);
            });

            window.location.href = "/";
      }

      else {
        response.json().then(data => {
          var snack = document.getElementById("snackbar");
snack.innerHTML = "";
console.log(data["detail"]);
snack.innerHTML += data["detail"];
snack.className = "show";
setTimeout(function(){ snack.className = snack.className.replace("show",
    ""); }, 3000);
            });
      }
     });
  }

  const form = document.querySelector('.contact-form');
  form.addEventListener('submit', handleFormSubmit);
</script>

        <div class="signup_link">
            You don't have an account? <a href="/signup-page">Signup</a>
          </div>
      </div>

</body>
</html>

<style>
  .center{
position: absolute;
top: 50%;
left: 50%;
transform: translate(-50%, -50%);
width: 400px;
background: white;
border-radius: 10px;
box-shadow: 10px 10px 15px rgba(0,0,0,0.05);
}
.center h1{
text-align: center;
padding: 20px 0;
border-bottom: 1px solid silver;

}
.center form{
padding: 0 40px;
box-sizing: border-box;
}
form .txt_field{
position: relative;
border-bottom: 2px solid #adadad;
margin: 30px 0;
```

```
}
.txt_field input{
width: 100%;
padding: 0 5px;
height: 40px;
font-size: 16px;
border: none;
background: none;
outline: none;
}
.txt_field label{
position: absolute;
top: 50%;
left: 5px;
color: #adadad;
transform: translateY(-50%);
font-size: 16px;
pointer-events: none;
transition: .5s;
}
.txt_field span::before{
content: '';
position: absolute;
top: 40px;
left: 0;
width: 0%;
height: 2px;
background: #2691d9;
transition: .5s;
}
.txt_field input:focus ~ label,
.txt_field input:valid ~ label{
top: -5px;
color: #2691d9;
}
.txt_field input:focus ~ span::before,
.txt_field input:valid ~ span::before{
width: 100%;
}
.pass{
margin: -5px 0 20px 5px;
color: #a6a6a6;
cursor: pointer;
}
.pass:hover{
text-decoration: underline;
}

.signup_link{
margin: 30px 0;
text-align: center;
font-size: 16px;
color: #666666;
}
.signup_link a{
```

```
color: #2691d9;
text-decoration: none;
}
.signup_link a:hover{
text-decoration: underline;
}

</style>

/index.astro

---
import BaseLayout from '../layouts/BaseLayout.astro';
const pageTitle = "Home";
---

<script src="../scripts/checkToken.js"/>

<script>
    let tokenToSend = `${localStorage.getItem("SavedToken")}`;

        fetch('http://127.0.0.1:8000/patient', {
          credentials: "same-origin",
          method: 'GET',
          headers: {
            'accept': 'application/json',
            'Authorization': tokenToSend
          },

        }).then(response =>
         {

            response.json().then(data => {
                const select = document.querySelector('select');
                let defOption = new Option("-----",null);
                console.log(data[1]);
                select.add(defOption,undefined);
                    for (var i = 0; i < data.length; i++){
                        let newOption = new Option(`${data[i]["
                            first_name"]} ${data[i]["last_name"]} ${data[
                            i]["PESEL"]}`,`${data[i]["id"]}`);

                        select.add(newOption,undefined)}
                });
         });
    </script>

<BaseLayout pageTitle={pageTitle}>

    <label >Predict</label>
        <section class="contact-form" >
            <div class="input-group">
                <label for="pregnancies">List of patients</label>
            <select class="input-group" id="selectedValue"  size="4">

            </select>
        </div>
            <form>

                <div class="input-group">
                    <label for="pregnancies">Pregnancies</label>
                    <input class="inptClass" id="pregnancies" name="pregnancies" type="number" min="0" required/>
                </div>

                <div class="input-group">
                    <label for="Glucose">Glucose</label>
                    <input class="inptClass1" id="Glucose" name="glucose" type="number" min="0" required/>
                </div>

                <div class="input-group">
                    <label for="name">Blood Pressure</label>
                    <input class="inptClass2" id="name" name="blood_pressure" type="number" min="0" required/>
                </div>

                <div class="input-group">
                    <label for="SkinThicc">Skin Thickness</label>
                    <input class="inptClass3" id="SkinThicc" name="skin_thickness" type="number" min="0" required/>
                </div>

                <div class="input-group">
                    <label for="Insulin">Insulin</label>
                    <input class="inptClass4" id="Insulin" name="insulin" type="number" min="0" required/>
                </div>

                <div class="input-group">
                    <label for="BMI">BMI</label>
                    <input class="inptClass5" id="BMI" name="bmi" type="number" min="0" required/>
                </div>

                <div class="input-group">
                    <label for="dpf">Diabetes Pedigree Function</label>
                    <input class="inptClass6" id="dpf" name="diabetes_pedigree_function" type="number" min="0" required/>
                </div>

                <div class="input-group">
                    <label for="age">Age</label>
                    <input class="inptClass7" id="age" name="age" type="number" min="1" step="1" required/>
                </div>

                <div class="input-group">
                    <button class="glob" type="submit">Predict</button>
```

```
                        <div class="pred">  </div>
                    </div>

                </form>
            </section>
<!--
            <div class="results">
                <h3>Form Data</h3>
                <pre></pre>
            </div>
    -->


    <script>
        var predVal;
        function handleFormSubmit(event) {
          event.preventDefault();
          let tokenToSend = `${localStorage.getItem("SavedToken")}`;
          var e = document.getElementById("selectedValue");
          var value = parseInt(e.value);
          var predd = document.querySelector('.pred');
          const data = new FormData(event.target);
          const formJSON = {};
          const dataJSON = {};
          var y;
          var sth;
          var isIt;
          console.log(data);
          data.forEach((value, key) => {
            dataJSON[key] = /^\d+$/.test(value) ? parseFloat(value) :
                value;
          });
          formJSON["patient_id"] = value;
          formJSON["input"] = dataJSON;
          console.log(JSON.stringify(formJSON, null, 2));
          fetch('http://127.0.0.1:8000/predict', {
            credentials: "same-origin",
            method: 'POST',
            headers: {
              'accept': 'application/json',
              'Authorization': tokenToSend,
              'Content-Type': 'application/json'
            },
            body: JSON.stringify(formJSON, null, 2)
          })
           .then(response => {
            response.json().then(data => {
                console.log(data);
                predd.innerHTML = '';
                sth = parseInt(data.prediction);
                if (sth != 0 ) {
                    isIt = "Patient doesn't have diabetes";
                }
                else{
                    isIt = "Patient does have diabetes";
                }
           y = document.createTextNode(isIt);
           predd.appendChild(y);
           });
        });

        }
        document.querySelector(".inptClass").addEventListener("keypress", function (evt) {
    if (evt.which != 8 && evt.which != 8 && evt.which != 0 && evt.which < 48 || evt.which > 57)
    {
        evt.preventDefault();
    }
});

document.querySelector(".inptClass1").addEventListener("keypress", function (evt) {
    if (evt.which != 8 && evt.which != 0 && evt.which == 46 && evt.which < 48 || evt.which > 57)
    {
        evt.preventDefault();
    }
});
document.querySelector(".inptClass2").addEventListener("keypress", function (evt) {
    if (evt.which != 8 && evt.which != 0 && evt.which == 46 && evt.which < 48 || evt.which > 57)
    {
        evt.preventDefault();
    }
});
document.querySelector(".inptClass3").addEventListener("keypress", function (evt) {
    if (evt.which != 8 && evt.which != 0 && evt.which == 46 && evt.which < 48 || evt.which > 57)
    {
        evt.preventDefault();
    }
});
document.querySelector(".inptClass4").addEventListener("keypress", function (evt) {
    if (evt.which != 8 && evt.which != 0 && evt.which == 46 && evt.which < 48 || evt.which > 57)
    {
        evt.preventDefault();
    }
});
document.querySelector(".inptClass5").addEventListener("keypress", function (evt) {
    if (evt.which != 8 && evt.which != 0 && evt.which == 46 && evt.which < 48 || evt.which > 57)
    {
        evt.preventDefault();
    }
```

```
});

document.querySelector(".inptClass6").addEventListener("keypress",
    function (evt) {
     if (evt.which != 8 && evt.which != 0 && evt.which == 46 && evt.which
         < 48 || evt.which > 57)
     {
         evt.preventDefault();
     }
});

document.querySelector(".inptClass7").addEventListener("keypress",
    function (evt) {
     if (evt.which != 8 && evt.which != 0 && evt.which < 48 || evt.which
        > 57)
     {
         evt.preventDefault();
     }
});

         const form = document.querySelector('.contact-form');
         form.addEventListener('submit', handleFormSubmit);
       </script>
</BaseLayout>

<style>
    .pred{
        float: right;
    }
</style>

//add-patient.astro

---
import Snack from '../components/Snack.astro';
import BaseLayout from '../layouts/BaseLayout.astro';
const pageTitle = "Add Patient";
---
<script src="../scripts/checkToken.js"/>

<BaseLayout pageTitle={pageTitle}>
    <label >Add patient</label>
    <section class="contact-form">
        <form>

            <div class="input-group">
                <label for="PESEL">PESEL</label>
                <input class="inptClass1" id="PESEL" name="PESEL" type="
                    number" min="0" step="1" required/>
            </div>

            <div class="input-group">
                <label for="first_name">Name</label>
                <input class="inptClass2" id="first_name" name="
                    first_name" type="text" required/>
            </div>

            <div class="input-group">
                <label for="last_name">Surname</label>
                <input class="inptClass3" id="last_name" name="last_name
                    " type="text" required/>
            </div>

            <div class="input-group">
                <label for="email">Email</label>
                <input class="inptClass4" id="email" name="email" type="
                    email" required/>
            </div>

            <div class="input-group">
                <label for="phone_number">Phone Number</label>
                <input class="inptClass5" id="phone_number" name="
                    phone_number" type="number" min="0" step="1" max="
                    999999999" required/>
            </div>

            <button class="glob" type="submit">Add</button>
        </form>
    </section>
    <Snack id="snackbar"></Snack>
    <script>
        function handleFormSubmit(event) {
          event.preventDefault();

          const data = new FormData(event.target);
              const formJSON = {};
              formJSON["id"] = null;
              data.forEach((value, key) => {
                formJSON[key] = /^\d+$/.test(value) ? String(value) :
                    value;
              });
              formJSON["historical_data"] = null;
              console.log(JSON.stringify(formJSON, null, 2));

          let tokenToSend = `${localStorage.getItem("SavedToken")}`;

          fetch('http://127.0.0.1:8000/patient', {
            credentials: "same-origin",
            method: 'POST',
            headers: {
              'accept': 'application/json',
              'Authorization': tokenToSend,
              'Content-Type': 'application/json'
            },
            body: JSON.stringify(formJSON, null, 2)
          }).then(response =>
     {
      if (response.status == 404) {
        response.json().then(data => {
          var snack = document.getElementById("snackbar");
```

```
snack.innerHTML = "";
console.log(data["detail"]);
snack.innerHTML += data["detail"];
snack.className = "show";
setTimeout(function(){ snack.className = snack.className.replace("show",
    ""); }, 3000);
            });
      }
      if (response.status == 200) {
        response.json().then(data => {
          var snack = document.getElementById("snackbar");
snack.innerHTML = "";
snack.innerHTML += "Patient added correctly";
snack.className = "show";
setTimeout(function(){ snack.className = snack.className.replace("show",
    ""); }, 3000);
            });
      }
     });
    };
    document.querySelector(".inptClass1").addEventListener("keypress",
       function (evt) {
    if (evt.which != 8 && evt.which != 0 && evt.which < 48 || evt.which
       > 57)
    {
        evt.preventDefault();
    }
});
document.querySelector(".inptClass2").addEventListener("keypress",
   function (evt) {
    var charCode1 = (evt.which) ? evt.which : evt.keyCode;
    if ((charCode1 >= 65 && charCode1 <= 90) || (charCode1 >= 97 &&
       charCode1 <= 122)) {
        // Allow the key press
    } else {
        // Prevent the key press if it doesn't meet the criteria
        evt.preventDefault();
    }
});
document.querySelector(".inptClass3").addEventListener("keypress",
   function (evt) {
    var charCode2 = (evt.which) ? evt.which : evt.keyCode;
    if ((charCode2 >= 65 && charCode2 <= 90) || (charCode2 >= 97 &&
       charCode2 <= 122)) {
        // Allow the key press
    } else {
        // Prevent the key press if it doesn't meet the criteria
        evt.preventDefault();
    }
});

document.querySelector(".inptClass5").addEventListener("keypress",
   function (evt) {
    if (evt.which != 8 && evt.which != 0 && evt.which < 48 || evt.which
       > 57)
    {
        evt.preventDefault();
    }
});
        const form = document.querySelector('.contact-form');
        form.addEventListener('submit', handleFormSubmit);
      </script>

</BaseLayout>

//BaseLayout.astro

---
import Header from '../components/Header.astro';
import '../styles/global.css';
const { pageTitle } = Astro.props;
---
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" type="image/svg+xml" href="/favicon.svg" />
    <meta name="viewport" content="width=device-width" />
    <meta name="generator" content={Astro.generator} />
    <title>{pageTitle}</title>
  </head>

  <body>
    <Header/>
    <div class="UploadFile">
    <slot>
</div>

  </body>
</html>

<style>
body {
    overflow: scroll;
}
</style>

//ThemeIcon.astro

---
---
<button id="themeToggle">
    <svg width="30px" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24
        24">
      <path class="sun" fill-rule="evenodd" d="M12 17.5a5.5 5.5 0 1 0
         0-11 5.5 5.5 0 0 0 0 11zm0 1.5a7 7 0 1 0 0-14 7 7 0 0 0 0 14
         zm12-7a.8.8 0 0 1-.8.8h-2.4a.8.8 0 0 1 0-1.6h2.4a.8.8 0 0 1
         .8.8zM4 12a.8.8 0 0 1-.8.8H.8a.8.8 0 0 1 0-1.6h2.5a.8.8 0 0 1
         .8.8zm16.5-8.5a.8.8 0 0 1 0 1l-1.8 1.8a.8.8 0 0 1-1-1l1.7-1.8a
         .8.8 0 0 1 1 0zM6.3 17.7a.8.8 0 0 1 0 1l-1.7 1.8a.8.8 0 1 1-1-1
         l1.7-1.8a.8.8 0 0 1 1 0zM12 0a.8.8 0 0 1 .8.8v2.5a.8.8 0 0
         1-1.6 0V.8A.8.8 0 0 1 12 0zm0 20a.8.8 0 0 1 .8.8v2.4a.8.8 0 0
         1-1.6 0v-2.4a.8.8 0 0 1 .8-.8zM3.5 3.5a.8.8 0 0 1 1 0l1.8 1.8a
         .8.8 0 1 1-1 1L3.5 4.6a.8.8 0 0 1 0-1zm14.2 14.2a.8.8 0 0 1 1 0
         l1.8 1.7a.8.8 0 0 1-1 1l-1.8-1.7a.8.8 0 0 1 0-1z"/>
      <path class="moon" fill-rule="evenodd" d="M16.5 6A10.5 10.5 0 0 1
         4.7 16.4 8.5 8.5 0 1 0 16.4 4.7l.1 1.3zm-1.7-2a9 9 0 0 1 .2 2 9
          9 0 0 1-11 8.8 9.4 9.4 0 0 1-.8-.3c-.4 0-.8.3-.7.7a10 10 0 0 0
          .3.8 10 10 0 0 0 9.2 6 10 10 0 0 0 4-19.2 9.7 9.7 0 0 0-.9-.3c
         -.3-.1-.7.3-.6.7a9 9 0 0 1 .3.8z"/>
    </svg>
  </button>

  <style>
    #themeToggle {
      border: 0;
      background: none;
      cursor: pointer;
    }
    .sun { fill: white; }
    .moon { fill: transparent; }

    button {
        float: right;
        margin-top: 5px;
        display:inline-block
    }

    :global(.dark) .sun { fill: transparent; }
    :global(.dark) .moon { fill: white; }
  </style>

<script is:inline>
    const theme = (() => {
      if (typeof localStorage !== 'undefined' && localStorage.getItem('
         theme')) {
        return localStorage.getItem('theme');
      }
      if (window.matchMedia('(prefers-color-scheme: dark)').matches) {
        return 'dark';
      }
        return 'light';
    })();

    if (theme === 'light') {
      document.documentElement.classList.remove('dark');
    } else {
      document.documentElement.classList.add('dark');
    }

    window.localStorage.setItem('theme', theme);

    const handleToggleClick = () => {
      const element = document.documentElement;
      element.classList.toggle("dark");

      const isDark = element.classList.contains("dark");
      localStorage.setItem("theme", isDark ? "dark" : "light");
    }

    document.getElementById("themeToggle").addEventListener("click",
       handleToggleClick);
  </script>

  //Snack.astro
---
---

<div id="snackbar"></div>

<style>
    #snackbar {
  visibility: hidden;
  min-width: 250px;
  margin-left: -125px;
  background-color: #333;
  color: #fff;
  text-align: center;
  border-radius: 2px;
  padding: 16px;
  position: fixed;
  z-index: 1;
  left: 50%;
  bottom: 30px;
}

#snackbar.show {
  visibility: visible;
  -webkit-animation: fadein 0.5s, fadeout 0.5s 2.5s;
  animation: fadein 0.5s, fadeout 0.5s 2.5s;
}

@-webkit-keyframes fadein {
  from {bottom: 0; opacity: 0;}
  to {bottom: 30px; opacity: 1;}
}

@keyframes fadein {
  from {bottom: 0; opacity: 0;}
  to {bottom: 30px; opacity: 1;}
}

@-webkit-keyframes fadeout {
  from {bottom: 30px; opacity: 1;}
  to {bottom: 0; opacity: 0;}
}

@keyframes fadeout {
  from {bottom: 30px; opacity: 1;}
  to {bottom: 0; opacity: 0;}
}
```

```
</style>

//ScriptSnack.jsx
import { useState } from 'preact/hooks';

export default function Snack({messages}) {

  const randomMessage = () => messages[(Math.floor(Math.random() *
     messages.length))];

  const [greeting, setGreeting] = useState(messages[0]);

  return (
    <div>
      <h3>{greeting}! Thank you for visiting!</h3>
      <button onClick={() => setGreeting(randomMessage())}>
        New Greeting
      </button>
    </div>
  );
}


var snack = document.getElementById("snackbar");
snack.innerHTML = "";
console.log(data["detail"]);
snack.innerHTML += data["detail"];
snack.className = "show";
setTimeout(function(){ snack.className = snack.className.replace("show",
    ""); }, 3000);

//Header.astro

---
import ThemeIcon from './ThemeIcon.astro';
---
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font
   -awesome/4.7.0/css/font-awesome.min.css">
<header>
  <nav>

      <a class="log-out"><i class="fa fa-sign-out" style="font-size:36px
         ;color:white"></i></a>
      <div class="nav-container">
        <a class="link-to-pages" href="/add-patient">Add Patient</a>
        <a class="link-to-pages" href="/">Predict</a>
        <a class="link-to-pages" href="/patients">All Patients</a>
        <a class="link-to-pages" href="/train">Train</a>
      </div>
      <ThemeIcon />
  </nav>

</header>

<script>
```

```
  const button = document.querySelector(".log-out");

  button.addEventListener("click", (event) => {
    localStorage.removeItem("SavedToken");
    window.location.href = "/login-page";
  });

        </script>

<style>
  a.log-out{
    float: left;
  }
  .nav-container{
    display: inline-block;
  }
  a.link-to-pages{
   border-style: none none solid none;
   border-radius: 4px;
   margin: 0px 5px 0px 5px;
   display:inline-block;
  }
</style>
```