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

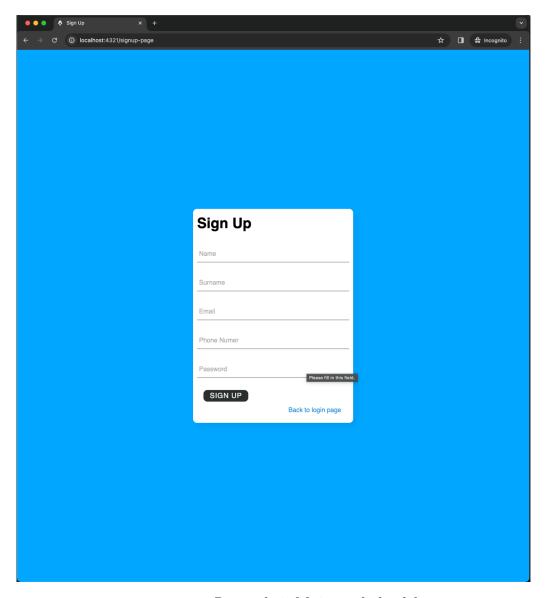
Dostęp do interfejsu webowego naszej aplikacji znajduje się pod adresem "localhost:4321" a dostęp do endpointów RestAPI znajduje się pod adresem "localhost:8000/docs"

Aby zalogować się, do obu aplikacji domyślnym kontem adminastratora jest:

email: senior registrar@hospital.com

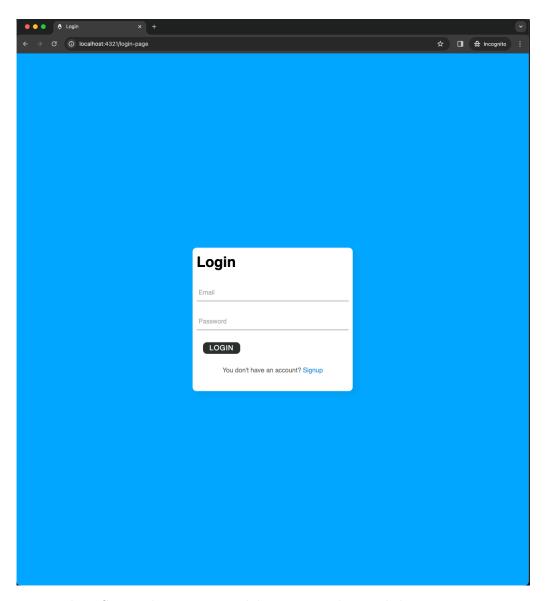
hasło: passwd

Gdy obie aplikacje działają mamy dostępne możliwości jak poniżej.



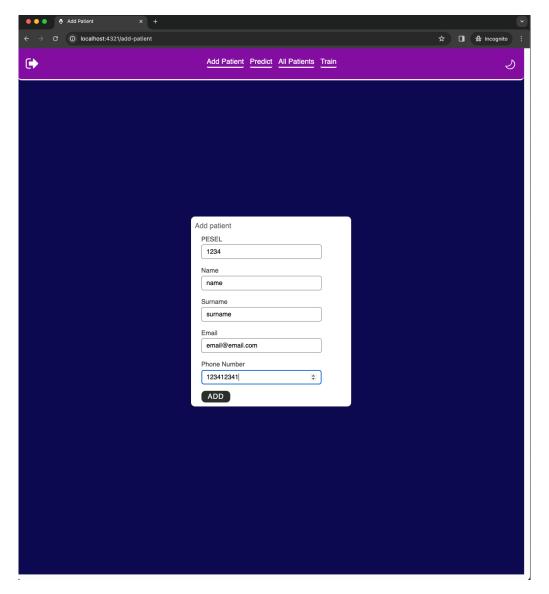
Rysunek 1: Możemy dodać lekarza

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

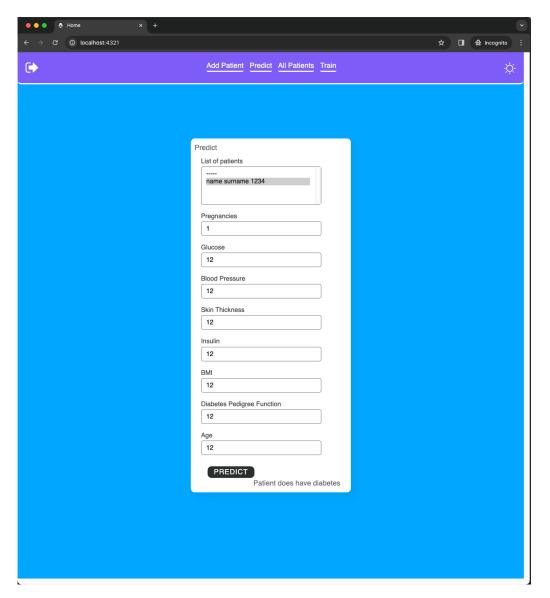
Pole email wymaga wpisania poprawnego emaila. Nie da się utworzyć konta bez któregoś z pól uzupełnionego.



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

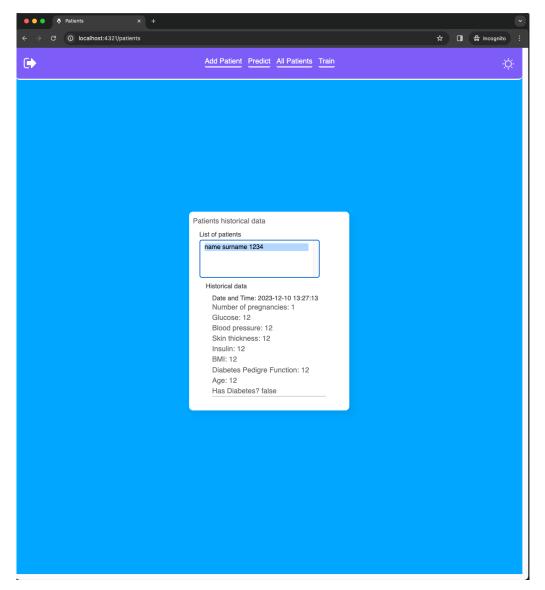
Pole PESEL wymaga jedynie cyfr, minimum jednej. Pola Name, Surname są zabezpieczone przed wpisaniem znaków poza literami, Phone Number wymaga jedynie cyfr, których w sumie jest dziewięć. Pole email wymaga wpisania poprawnego emaila. Nie da się utworzyć pacjenta bez któregoś z pól uzupełnionego.

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

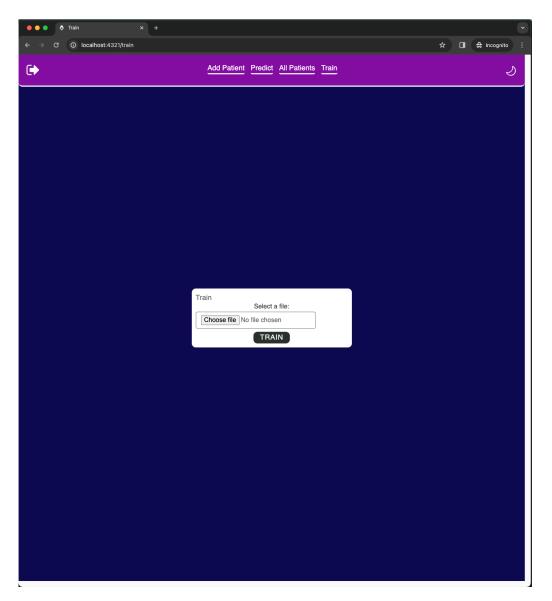


Rysunek 4: Wybieramy pacjenta bądź pole puste, aby przetestować bez zapisywania danych do pacjenta

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



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 odpowiedznich 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
```

```
def _setup_database_object(self) -> None:
           try:
               self.db = SqliteCipher(
3
                   dataBasePath=os.getenv("DB_PATH"),
4
                    checkSameThread=False,
                    password=os.getenv("DB_PASSWORD"),
6
               )
           except Exception as ex:
               logging.error(ex)
10
           for table_name in ["diabetes", "doctors", "patients"]:
11
               if not self.db.checkTableExist(table_name):
                    self._init_database()
13
                   break
14
15
      def _init_database(self) -> None:
           self.db.createTable(
17
               "diabetes",
18
               19
                    ["pregnancies", "INT"],
20
                    ["glucose", "REAL"],
21
                    ["blood_pressure", "REAL"],
22
                    ["skin_thickness", "REAL"],
23
                    ["insulin", "REAL"],
                    ["bmi", "REAL"],
25
                    ["diabetes_pedigree_function", "REAL"],
26
                    ["age", "INT"],
27
                    ["outcome", "INT"],
               ],
29
               True,
30
               True,
31
           )
33
           self.db.createTable(
34
               "doctors",
35
               36
                    ["first_name", "TEXT"],
37
                    ["last_name", "TEXT"],
                    ["email", "TEXT"],
                    ["phone_number", "TEXT"],
40
                    ["hashed_password", "TEXT"],
41
               ],
42
43
               True,
```

```
True,
44
           )
45
46
           self.db.insertIntoTable(
47
                "doctors",
                Г
49
                    "Senior",
50
                    "Registrar",
                    "senior_registrar@hospital.com",
                    "+48-111-222-333",
53
                    password_utils.get_hashed_password(os.getenv("
54
                        DEV_PASSWORD")),
                ],
55
                True,
56
57
           self.db.createTable(
59
                "patients",
60
                61
                    ["PESEL", "TEXT"],
                    ["first_name", "TEXT"],
63
                    ["last_name", "TEXT"],
                    ["email", "TEXT"],
                    ["phone_number", "TEXT"],
                    ["historical_data", "BLOB"],
67
                ],
68
                True,
69
                True,
70
           )
71
```

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

```
1 //test.py
3 import requests
_{4} port = 8000
5 def test_train_model_endpoint(token):
      headers = {
           'accept': 'application/json',
           'Authorization': f'Bearer {token}',
8
          # requests won't add a boundary if this header is set when you
9
              pass files=
          # 'Content-Type': 'multipart/form-data',
10
11
12
      files = {
13
           'file': ('diabetes.csv', open('diabetes.csv', 'rb'), 'text/csv')
14
      }
15
16
      response = requests.post('http://localhost:8000/train', headers=
17
         headers, files=files)
      assert response.status_code == 200
18
      print("train 200")
19
20
21 def get_login():
      headers = {
22
      'accept': 'application/json',
23
      'Content-Type': 'application/x-www-form-urlencoded',
24
      }
25
26
      data = {
           'grant_type': '',
28
           'username': 'senior_registrar@hospital.com',
29
          'password': 'passwd',
          'scope': '',
31
           'client_id': '',
32
           'client_secret': '',
33
      }
34
35
      response = requests.post('http://localhost:8000/login', headers=
36
         headers, data=data)
37
      return response
38
40 def test_predict_diabetes_endpoint(token):
      headers = {
41
           'accept': 'application/json',
42
           'Authorization': f'Bearer {token}',
43
           'Content-Type': 'application/json',
44
45
      }
46
      json_data = {
47
          'patient_id': None,
48
```

```
'input': {
49
               'pregnancies': 0,
50
               'glucose': 0,
51
               'blood_pressure': 0,
52
               'skin_thickness': 0,
               'insulin': 0,
54
               'bmi': 0,
55
               'diabetes_pedigree_function': 0,
56
               'age': 0,
           },
58
      }
59
60
       response = requests.post(f'http://localhost:{port}/predict', headers
61
          =headers, json=json_data)
       assert response.status_code == 200
62
       print("predict: 200")
63
64
65 def test_get_data_endpoint(token):
      headers = {
66
           'accept': 'text/html',
67
           'Authorization': f'Bearer {token}',
68
69
70
      response = requests.get(f'http://localhost:{port}/data', headers=
71
          headers)
       assert response.status_code == 200
72
       print("data: 200")
73
74
75 def test_signup_endpoint(token):
      print("signup")
76
      url = f"http://localhost:{port}/signup"
77
78
      # Replace this with the appropriate user data for your endpoint
      user_data = {
79
           "email": "test@example.com",
80
           "password": "yourpassword"
81
      response = requests.post(url, json=user_data)
83
       assert response.status_code == 200
85 token = get_login().json().get("access_token")
86 test_get_data_endpoint(token)
87 test_predict_diabetes_endpoint(token)
88 test_train_model_endpoint(token)
89
90 //app.py
91
92 import datetime
93 from typing import List
94 from fastapi import FastAPI, HTTPException, UploadFile, Depends
95 from fastapi.responses import HTMLResponse
96 from fastapi.middleware.cors import CORSMiddleware
97 import pandas as pd
98 from fastapi.security import OAuth2PasswordRequestForm
99 from sklearn.model_selection import train_test_split
100 from sklearn.linear_model import LogisticRegression
101 from dotenv import load_dotenv
```

```
102 import joblib
103 from kerastuner.tuners import RandomSearch
104 from keras_tuner import HyperModel
{\scriptstyle 105} from fastapi import FastAPI, HTTPException, UploadFile, Depends
106 from fastapi.responses import HTMLResponse
107 from fastapi.middleware.cors import CORSMiddleware
108 import pandas as pd
109 from fastapi.security import OAuth2PasswordRequestForm
110 from sklearn.model_selection import train_test_split
111 from sklearn.linear_model import LogisticRegression
112 from dotenv import load_dotenv
113 import joblib
114 from keras.models import Sequential
115 from keras.layers import Dense
116 from db_manager import DatabaseManager
117 from keras.models import load_model
119 from jwt_utils import (
       create_access_token,
120
       create_refresh_token,
121
122
       get_current_user,
123 )
124 from db_models import (
      UserRegister,
125
126
       TokenSchema,
      User,
127
      Patient,
128
      PredictionInput,
129
       DiabetesHistoricalOutput,
130
132 from password_utils import verify_password
134 load_dotenv()
135 db = DatabaseManager()
136 app = FastAPI()
137 app.add_middleware(
       CORSMiddleware,
138
       allow_origins=["*"],
139
       allow_credentials=True,
140
       allow_methods = ["*"],
141
       allow_headers = ["*"],
142
143
144
145 model_name = "model"
147 class MyHyperModel(HyperModel):
       def __init__(self, input_shape):
           self.input_shape = input_shape
149
150
       def build(self, hp):
151
           model = Sequential()
           model.add(Dense(units=hp.Int('units', min_value=32, max_value
153
               =512, step=32),
                             activation='relu', input_shape=self.input_shape)
154
                                )
```

```
model.add(Dense(1, activation='linear'))
           model.compile(optimizer='adam', loss='mean_squared_error')
156
           return model
157
158
159 @app.post("/train")
160 async def train_model(file: UploadFile, user: User = Depends(
      get_current_user)):
       db.insert_diabetes_data(file.file)
161
162
       df = db.select_all_diabetes_data()
163
164
       X = df.iloc[:, :-1]
165
       y = df.iloc[:, -1]
166
       X_train, X_test, y_train, y_test = train_test_split(
167
           X, y, test_size=0.2, random_state=42
168
169
170
       input_size = X_train.shape[1]
                                         # Number of features
171
       model = Sequential()
172
       model.add(Dense(64, input_shape=(input_size,), activation='linear'))
173
       model.add(Dense(32, activation='linear'))
174
       model.add(Dense(16, activation='linear'))
175
       model.add(Dense(1, activation='linear'))
176
       model.compile(optimizer='adam', loss='mean_squared_error')
179
       input_shape = [X_train.shape[1]] # Assuming X_train is your input
180
       hypermodel = MyHyperModel(input_shape=input_shape)
181
182
183
       tuner = RandomSearch(
184
       hypermodel,
185
       objective='val_loss',
186
187
       max_trials=5,
       executions_per_trial=3,
188
       directory='my_dir',
189
       project_name='hparam_tuning'
190
191
192
       tuner.search(X_train, y_train, epochs=10, validation_data=(X_test,
193
          y_test))
       best_model = tuner.get_best_models(num_models=1)[0]
194
       best_model.save(model_name)
195
196
       return {"message": "Model trained and saved successfully"}
197
198
199
200 @app.post("/predict")
201 async def predict_diabetes(
       input_data: PredictionInput, user: User = Depends(get_current_user)
202
<sub>203</sub> ):
204
205
       try:
           model = load_model(model_name)
```

```
except FileNotFoundError:
207
           raise HTTPException(
208
                status_code=500, detail="Model not found. Please train the
209
                   model first."
           )
210
       i_data = input_data.dict()['input']
211
       input_df = pd.DataFrame([i_data])
212
       prediction = model.predict(input_df)
213
       prediction_native_type = int(prediction[0])
                                                      # or float, as
          appropriate
       if input_data.patient_id is not None:
215
           db.add_historical_data_to_patient(
216
                patient_id=input_data.patient_id,
217
                output=DiabetesHistoricalOutput(
218
                    pregnancies=input_data.input.pregnancies,
219
                    glucose=input_data.input.glucose,
220
                    blood_pressure=input_data.input.blood_pressure,
                    skin_thickness=input_data.input.skin_thickness,
222
                    insulin=input_data.input.insulin,
223
                    bmi=input_data.input.bmi,
224
                    diabetes_pedigree_function=input_data.input.
225
                       diabetes_pedigree_function,
                    age=input_data.input.age,
226
                    prediction=bool(prediction_native_type),
227
                    created_at=datetime.datetime.today().timestamp(),
228
               ),
229
           )
230
231
       return {"prediction": prediction_native_type}
232
233
234
236
237 @app.get("/data", response_class=HTMLResponse)
238 async def get_data_from_database(user: User = Depends(get_current_user))
       return db.select_all_diabetes_data().to_html(notebook=True)
239
240
242 Capp.post("/signup", summary="Create new user")
  async def create_user(data: UserRegister):
243
       return db.create_doctor(data)
244
245
246
  @app.post(
247
       "/login",
248
       summary="Create access and refresh tokens for user",
249
       response_model=TokenSchema,
250
251 )
252 async def login(form_data: OAuth2PasswordRequestForm = Depends()):
       user = db.get_doctor(form_data.username)
       print(form_data.username)
254
       if user is None:
255
           raise HTTPException(status_code=400, detail="Incorrect email or
256
              password")
```

```
257
       hashed_pass = user.hashed_password
258
       print(hashed_pass)
259
       if not verify_password(form_data.password, hashed_pass):
260
           raise HTTPException(
261
                status_code=400,
262
                detail="Incorrect email or password",
263
           )
264
       return {
266
           "access_token": create_access_token(user.email),
267
           "refresh_token": create_refresh_token(user.email),
268
       }
269
270
271
272 Capp.get("/me", summary="Get details of currently logged in user",
      response_model=User)
273 async def get_me(user: User = Depends(get_current_user)):
       return user
274
275
276
277 @app.post("/patient", summary="Adds new patient to database",
      response_model=Patient)
278 async def create_patient(patient_data: Patient, user: User = Depends(
      get_current_user)):
       return db.create_patient(patient_data)
279
280
282 @app.get(
       "/patient", summary="Get list of patients with data", response_model
283
          =List[Patient]
285 async def get_users(user: User = Depends(get_current_user)):
       return db.get_patients()
286
287
288 //db_manager.py
289
290 import datetime
291 from typing import Optional, BinaryIO
293 import pandas as pd
294 from fastapi import HTTPException
295 from fastapi.responses import Response
296 from pysqlitecipher.sqlitewrapper import SqliteCipher
297 import os
298 import logging
300 import password_utils
301 from db_models import UserOut, UserRegister, Patient,
      DiabetesHistoricalOutput
302
303
304 class SingletonMeta(type):
       _instances = {}
305
306
```

```
def __call__(cls, *args, **kwargs):
307
            if cls not in cls._instances:
308
                 instance = super().__call__(*args, **kwargs)
309
                 cls._instances[cls] = instance
310
            return cls._instances[cls]
311
312
313
314 class DatabaseManager(metaclass=SingletonMeta):
        def __init__(self) -> None:
            self.db: Optional[SqliteCipher] = None
316
            self._setup_database_object()
317
318
        def _setup_database_object(self) -> None:
319
            try:
320
                 self.db = SqliteCipher(
321
                      dataBasePath=os.getenv("DB_PATH"),
322
                      checkSameThread=False,
323
                      password=os.getenv("DB_PASSWORD"),
324
                 )
325
            except Exception as ex:
326
                 logging.error(ex)
327
328
            for table_name in ["diabetes", "doctors", "patients"]:
329
                 if not self.db.checkTableExist(table_name):
330
331
                      self._init_database()
                      break
332
333
        def _init_database(self) -> None:
334
            self.db.createTable(
335
                 "diabetes",
336
                 Г
337
                      ["pregnancies", "INT"],
338
                      ["glucose", "REAL"],
339
                      ["blood_pressure", "REAL"], ["skin_thickness", "REAL"],
340
341
                      ["insulin", "REAL"],
342
                      ["bmi", "REAL"],
343
                      ["diabetes_pedigree_function", "REAL"],
344
                      ["age", "INT"],
345
                      ["outcome", "INT"],
346
                 ],
347
                 True,
348
349
                 True,
            )
350
351
            self.db.createTable(
352
                 "doctors",
353
                 Г
354
                      ["first_name", "TEXT"],
["last_name", "TEXT"],
355
356
                      ["email", "TEXT"],
357
                      ["phone_number", "TEXT"],
358
                      ["hashed_password", "TEXT"],
359
                 ],
360
                 True,
361
```

```
362
                 True,
            )
363
364
            self.db.insertIntoTable(
365
                "doctors",
366
                 Г
367
                     "Senior",
368
                     "Registrar",
369
                     "senior_registrar@hospital.com",
                     "+48-111-222-333",
371
                     password_utils.get_hashed_password(os.getenv("
372
                         DEV_PASSWORD")),
                ],
373
                True,
374
375
376
377
            self.db.createTable(
                 "patients",
378
                 379
                     ["PESEL", "TEXT"],
380
                     ["first_name", "TEXT"],
381
                     ["last_name", "TEXT"],
382
                     ["email", "TEXT"],
383
                     ["phone_number", "TEXT"],
384
                     ["historical_data", "BLOB"],
385
                ],
386
                True,
387
                True,
388
            )
389
390
       def select_all_diabetes_data(self) -> pd.DataFrame:
391
            columns, data = self.db.getDataFromTable("diabetes", True, True)
392
            return pd.DataFrame(data=data, columns=columns)
393
394
       def insert_diabetes_data(self, csv_file: BinaryIO) -> None:
395
            df = pd.read_csv(csv_file)
396
            df.columns = [
397
                 "pregnancies",
398
                 "glucose",
399
                 "blood_pressure",
400
                 "skin_thickness",
401
                 "insulin",
402
                 "bmi",
403
                 "diabetes_pedigree_function",
404
                 "age",
405
                 "outcome",
406
            ]
407
408
            for i in df.index:
409
                self.db.insertIntoTable(
410
                     "diabetes", [df[col_name][i] for col_name in df.columns
411
                         ], True
                )
412
413
       def dump_diabetes_data_to_csv(self) -> None:
414
```

```
pass
415
416
       def get_doctor(self, email: str) -> Optional[UserOut]:
417
           columns, data = self.db.getDataFromTable("doctors", True)
418
           for d in data:
419
                if d[3] == email:
420
                    return UserOut(
421
                         first_name=d[1],
422
                         last_name=d[2],
                         email=d[3],
424
                         phone_number = d[4],
425
                         hashed_password=d[5],
426
                    )
427
           return None
428
429
       def create_doctor(self, user: UserRegister):
430
            columns, data = self.db.getDataFromTable("doctors", True)
431
           for d in data:
432
                if d[3] == user.email or d[4] == user.phone_number:
433
                    raise HTTPException(
434
                         status_code=404,
435
                         detail=f"Doctor with email {user.email} or phone
436
                            number {user.phone_number} already exists.",
437
            self.db.insertIntoTable(
438
                "doctors",
439
                Г
440
                    user.first_name,
441
                    user.last_name,
442
                    user.email,
443
                    user.phone_number,
444
                    password_utils.get_hashed_password(user.password),
                ],
446
                True,
447
           )
448
           return Response (
449
                f"Successfully created a doctor {user.first_name} {user.
450
                   last_name}"
           )
451
452
       def create_patient(self, patient: Patient):
453
           columns, data = self.db.getDataFromTable("patients", True)
454
           for d in data:
455
                if (
456
                    d[1] == patient.PESEL
457
                    or d[4] == patient.email
458
                    or d[5] == patient.phone_number
                ):
460
                    raise HTTPException(
461
                         status_code=404,
462
                         detail=f"Patient with PESEL {patient.PESEL} or email
463
                              {patient.email} or phone number {patient.
                            phone_number} already exists.",
                    )
464
            self.db.insertIntoTable(
465
```

```
"patients",
466
467
                    patient.PESEL,
468
                    patient.first_name,
469
                    patient.last_name,
470
                    patient.email,
471
                    patient.phone_number,
472
                    "".encode(),
473
                ],
                True,
475
           )
476
477
           return patient
478
479
       def add_historical_data_to_patient(
480
           self , patient_id: int , output: DiabetesHistoricalOutput
       ):
482
           columns, data = self.db.getDataFromTable("patients", True)
483
           exists = False
484
           patient_data = None
485
           print("hm")
486
           for d in data:
487
                if d[0] == patient_id:
488
                    exists = True
                    patient_data = d[6]
490
                    break
491
           if not exists:
492
                raise HTTPException(
                    status_code=404,
494
                    detail=f"Patient with ID {patient_id} does not exists.",
495
                )
496
           print("got patient")
497
           patient_data = patient_data.decode()
498
           print("patient current data:", patient_data)
499
500
           data_str = (
                f"{output.pregnancies}, {output.glucose}, {output.
501
                   blood_pressure}, {output.skin_thickness}, "
                f"{output.insulin},{output.bmi},{output.
502
                   diabetes_pedigree_function},{output.age},"
                f"{output.prediction},{output.created_at.timestamp()};"
503
           )
504
505
506
           patient_data += data_str
507
           print("patient current data:", patient_data)
508
           self.db.updateInTable(
509
                "patients", patient_id, "historical_data", patient_data.
510
                   encode(), True
511
512
       def get_patients(self):
           columns, data = self.db.getDataFromTable("patients", True)
514
           patients = []
515
           for d in data:
516
                historical_data = []
```

```
518
                for data_point in d[6].decode().split(";")[:-1]:
519
                     data_point = data_point.split(",")
520
                     historical_data.append(
521
                         DiabetesHistoricalOutput(
522
                              pregnancies=data_point[0],
523
                              glucose=data_point[1],
524
                              blood_pressure=data_point[2],
525
                              skin_thickness=data_point[3],
                              insulin=data_point[4],
527
                              bmi=data_point[5],
528
                              diabetes_pedigree_function=data_point[6],
529
                              age=data_point[7],
530
                              prediction=data_point[8],
531
                              created_at=data_point[9],
532
                         )
533
                     )
534
535
                patients.append(
536
                     Patient(
537
                         id=d[0],
538
                         PESEL=d[1],
539
                         first_name=d[2],
540
                         last_name=d[3],
541
                         email=d[4],
                         phone_number=d[5],
543
                         historical_data=historical_data,
544
                     )
545
                )
546
            return patients
547
548 //db_models.py
550 from datetime import datetime
551 from typing import List, Optional
553 from pydantic import BaseModel
554
555
  class DiabetesPredictionInput(BaseModel):
556
       pregnancies: int
557
       glucose: float
558
       blood_pressure: float
559
       skin_thickness: float
560
       insulin: float
561
       bmi: float
562
       diabetes_pedigree_function: float
563
       age: int
564
565
566
567 class DiabetesHistoricalOutput(DiabetesPredictionInput):
       prediction: bool
       created_at: datetime
569
570
572 class Patient(BaseModel):
```

```
573
       id: Optional[int]
       PESEL: str
574
       first_name: str
575
       last_name: str
576
       email: str
577
       phone_number: str
578
       historical_data: Optional[List[DiabetesHistoricalOutput]]
579
580
582 class PredictionInput(BaseModel):
       patient_id: Optional[int]
583
       input: DiabetesPredictionInput
584
585
586
587 class User(BaseModel):
       first_name: str
589
       last_name: str
       email: str
590
       phone_number: str
591
592
593
594 class UserOut(User):
       hashed_password: str
595
596
597
598 class UserRegister(User):
       password: str
599
600
601
602 class TokenSchema(BaseModel):
       access_token: str
603
604
       refresh_token: str
605
606
607 class TokenPayload(BaseModel):
      sub: str = None
       exp: int = None
609
610
611 jwt_utils.py
613 from datetime import datetime, timedelta
614 from typing import Union, Any, Optional
616 from dotenv import load_dotenv
617 from fastapi import HTTPException, Depends
618 from fastapi.security import OAuth2PasswordBearer
619 from jose import jwt
620
621 import os
622
623 from pydantic import ValidationError
625 from db_manager import DatabaseManager
626 from db_models import TokenPayload, UserOut, User
627
```

```
628 load_dotenv()
630 ACCESS_TOKEN_EXPIRE_MINUTES = 120 # 120 minutes
631 REFRESH_TOKEN_EXPIRE_MINUTES = 60 * 24 * 7 # 7 days
632 ALGORITHM = "HS256"
633 JWT_SECRET_KEY = os.getenv("JWT_SECRET_KEY") # should be kept secret
634 JWT_REFRESH_SECRET_KEY = os.getenv("JWT_REFRESH_SECRET_KEY") # should
      be kept secret
635
636
637 db = DatabaseManager()
639 reuseable_oauth = OAuth2PasswordBearer(tokenUrl="/login", scheme_name="
      JWT")
640
641
642 def create_access_token(subject: Union[str, Any]) -> str:
       expires_delta = datetime.utcnow() + timedelta(minutes=
643
          ACCESS_TOKEN_EXPIRE_MINUTES)
644
       to_encode = {"exp": expires_delta, "sub": str(subject)}
645
       encoded_jwt = jwt.encode(to_encode, JWT_SECRET_KEY, ALGORITHM)
646
       return encoded_jwt
647
648
650 def create_refresh_token(subject: Union[str, Any]) -> str:
       expires_delta = datetime.utcnow() + timedelta(minutes=
651
          REFRESH_TOKEN_EXPIRE_MINUTES)
652
       to_encode = {"exp": expires_delta, "sub": str(subject)}
653
       encoded_jwt = jwt.encode(to_encode, JWT_REFRESH_SECRET_KEY,
654
          ALGORITHM)
       return encoded_jwt
655
656
657
658 async def get_current_user(token: str = Depends(reuseable_oauth)) ->
      User:
       try:
659
           payload = jwt.decode(token, JWT_SECRET_KEY, algorithms=[
660
               ALGORITHM])
           token_data = TokenPayload(**payload)
661
662
           if datetime.fromtimestamp(token_data.exp) < datetime.now():</pre>
663
                raise HTTPException(
664
                    status_code=401,
665
                    detail="Token expired",
666
                    headers = { "WWW - Authenticate": "Bearer"} ,
667
                )
668
       except (jwt.JWTError, ValidationError):
669
           raise HTTPException(
670
                status_code=403,
671
                detail="Could not validate credentials",
672
                headers = { "WWW - Authenticate": "Bearer"},
673
           )
674
675
```

```
676
       user: Optional[UserOut] = db.get_doctor(token_data.sub)
677
        if user is None:
678
            raise HTTPException(
679
                 status_code=404,
                 detail="Could not find user",
681
682
683
        return user
685
686 //password_utils.py
  from passlib.context import CryptContext
690 password_context = CryptContext(schemes=["bcrypt"], deprecated="auto")
691
693 def get_hashed_password(password: str) -> str:
        return password_context.hash(password)
694
695
696
697 def verify_password(password: str, hashed_pass: str) -> bool:
        return password_context.verify(password, hashed_pass)
698
700 //requierments.txt
701
_{702} absl-py==2.0.0
703 annotated-types==0.6.0
_{704} \text{ anyio} == 3.7.1
705 astunparse == 1.6.3
706 bcrypt == 4.1.1
707 cachetools == 5.3.2
708 certifi == 2023.11.17
709 \text{ cffi} == 1.16.0
710 charset-normalizer==3.3.2
711 click == 8.1.7
712 cryptography == 41.0.7
713 \text{ dm-tree} = = 0.1.8
714 ecdsa==0.18.0
715 fastapi == 0.104.1
716 flatbuffers == 23.5.26
717 gast == 0.5.4
718 google-auth==2.24.0
719 google-auth-oauthlib==1.1.0
_{720} google-pasta==0.2.0
721 grpcio == 1.59.3
_{722} h11 == 0.14.0
_{723} h5py == 3.10.0
724 httpcore == 1.0.2
_{725} httpx==0.25.2
_{726} idna==3.4
727 iniconfig == 2.0.0
728 joblib == 1.3.2
_{729} keras == 2.15.0
730 keras-tuner==1.4.6
```

```
_{731} kt-legacy==1.0.5
732 libclang == 16.0.6
733 Markdown == 3.5.1
734 \text{ markdown-it-py} == 3.0.0
735 MarkupSafe == 2.1.3
736 mdurl == 0.1.2
737 ml-dtypes == 0.2.0
namex == 0.0.7
numpy = = 1.26.2
740 oauthlib == 3.2.2
741 onetimepad == 1.4
742 opt-einsum == 3.3.0
743 packaging == 23.2
744 pandas == 2.1.3
745 passlib == 1.7.4
746 pluggy == 1.3.0
747 protobuf == 4.23.4
748 pyasn1 == 0.5.1
_{749} pyasn1-modules==0.3.0
750 pycparser == 2.21
751 pydantic == 2.5.1
752 pydantic_core == 2.14.3
753 Pygments == 2.17.2
754 pysqlitecipher == 0.22
755 pytest == 7.4.3
756 python-dateutil==2.8.2
757 python-dotenv==1.0.0
758 python-jose==3.3.0
759 python-multipart==0.0.6
760 pytz == 2023.3.post1
761 requests == 2.31.0
762 requests - oauthlib == 1.3.1
rich = = 13.7.0
rsa == 4.9
765 scikit-learn==1.3.2
_{766} \text{ scipy} == 1.11.4
_{767} \text{ six} == 1.16.0
768 sniffio == 1.3.0
769 starlette == 0.27.0
770 tensorboard == 2.15.1
771 tensorboard-data-server==0.7.2
_{772} tensorflow == 2.15.0
773 tensorflow-estimator==2.15.0
774 tensorflow-io-gcs-filesystem == 0.34.0
775 tensorflow-macos==2.15.0
776 termcolor == 2.4.0
777 threadpoolctl==3.2.0
778 typing_extensions == 4.8.0
779 tzdata == 2023.3
780 urllib3 == 2.1.0
781 uvicorn == 0.24.0.post1
782 Werkzeug == 3.0.1
783 wrapt == 1.14.1
784 pysqlitecipher == 0.22
python-dotenv==1.0.0
```

```
786 python-multipart==0.0.6
787 passlib == 1.7.4
788 \text{ python-jose} == 3.3.0
789
790
  //global.css
791
792 *{
     margin: 0;
793
     padding: 0;
     box-sizing: border-box;
795
     font-family: "Inter", sans-serif;
796
797 }
798
799 : root {
     --color-lightest: #f9fdfe;
800
     --color-gray-light: #cdcfcf;
801
802
     --color-gray-medium: #686a69;
     --color-gray-dark: #414643;
803
     --color-darkest: #2a2f2c;
804
805 }
806
       .input-group {
807
            margin-top: 0.25rem;
808
            padding: 0.5rem 1rem;
809
810
811
       .contact-form label {
812
            display: block;
            color: var(--color-gray-dark);
814
            font-family: Lato, sans-serif;
815
            font-size: 90%;
816
817
            line-height: 1.125;
       }
818
819
       .group-label {
820
            display: inline-block;
821
            margin-right: 1rem;
822
            font-size: 90%;
823
       }
824
825
       .contact-form label.inline {
826
            display: inline-block;
827
828
            margin-left: 0.25rem;
       }
829
830
       .contact-form input,
831
832
       .contact-form select,
       .contact-form textarea{
833
            display: block;
834
            margin-top: 0.25rem;
835
            padding: 0.5rem 0.75rem;
836
            border-radius: 5px;
837
            border: 1px solid var(--color-gray-medium);
838
            width: 300px;
839
            font-family: "Open Sans", sans-serif;
840
```

```
font-size: 1rem;
841
            transition: 150ms border-color linear;
842
       }
843
844
845
       .contact-form input:focus,
846
       .contact-form input:active {
847
            border-color: var(--color-gray-medium);
848
       }
850
       button.glob{
851
852
            display: block;
853
            margin: 0.5rem 1rem 0;
854
            padding: 0 1rem 0.125rem;
855
            border-radius: 10px;
856
            background-color: var(--color-darkest);
857
            border: 0;
858
            color: var(--color-lightest);
859
            font-family: lato, sans-serif;
860
            font-size: 100%;
861
            letter-spacing: 0.05em;
862
            line-height: 1.5;
863
            text-transform: uppercase;
864
865
            transition: 150ms all linear;
866
867
       .contact-form button:hover,
       .contact-form button:active,
869
       .contact-form button:focus {
870
            background: var(--color-darkest);
871
            cursor: pointer;
       }
873
874
       .result {
875
            margin: 10px;
876
877
878
       .space {
879
            margin: 10px;
880
            height: 5px;
881
            background: gray;
882
            border-radius: 10px;
883
       }
884
885
886
887 a {
     text-decoration: none;
888
     color: white;
889
890 }
891 header {
   text-align: center;
     position: fixed;
893
     height: 80px;
894
     width: 100%;
895
```

```
896
     z-index: 100;
     padding: 20px 20px;
897
     border-style: none none solid none ;
898
     border-radius: Opx Opx 10px 10px;
899
     border-color: white;
     background-color: rgba(249, 18, 242, 0.5);
901
902 }
903
   .UploadFile {
904
       position: absolute;
905
       top: 50%;
906
       left: 50%;
907
       transform: translate(-50%, -50%);
908
       width: 400px;
909
       background: white;
910
       border-radius: 10px;
911
       box-shadow: 10px 10px 15px rgba(0,0,0,0.05);
912
       padding: 10px;
913
914
915
916
917 .hamburger {
     display: inline-block;
918
     width: fit-content;
919
920
     block-size: fit-content;
     padding-right: 20px;
921
     cursor: pointer;
922
923 }
924
  .hamburger .line {
925
    display: block;
926
     width: 40px;
     height: 5px;
928
     margin-bottom: 10px;
929
     background-color: #ff9776;
930
931 }
932
933 body {
     margin: 0;
934
     padding: 0;
935
     height: auto;
936
     overflow: hidden;
937
938
     background: var(--color-lightest);
     color: var(--color-gray-medium);
939
     font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto,
940
        Helvetica,
     Arial, sans-serif, "Apple Color Emoji", "Segoe UI Emoji", "Segoe UI
941
        Symbol";
     font-size: 18px;
942
     line-height: 1.45;
943
944 }
945
946 html
947 {
       background-color: rgb(0, 166, 255);
```

```
949 }
950
951 html.dark {
     background-color: #0d0950;
952
954 }
955
956 //checkToken.js
958 let tokenToSend = '${localStorage.getItem("SavedToken")}';
959
          fetch('http://127.0.0.1:8000/me', {
960
            credentials: "same-origin",
961
            method: 'GET',
962
            headers: {
963
              'accept': 'application/json',
964
              'Authorization': tokenToSend
965
            },
966
967
          }).then(response =>
968
969
            if (response.status == 401) {
970
                   response.json().then(data => {
971
                     console.log(tokenToSend);
972
                     window.location.href = "/login-page";
                   });
974
            }});
975
977 //train.astro
978
979 ---
980 import BaseLayout from '../layouts/BaseLayout.astro';
981 const pageTitle = "Train";
982 ---
983
984 <script src="../scripts/checkToken.js"/>
985
986 <BaseLayout pageTitle={pageTitle}>
       <label >Train </label>
987
   <div>
988
        <form id="trainForm">
989
        <section class="contact-form">
990
        <label for="csvToTrain">Select a file:</label>
991
        <input type="file" id="csvToTrain" name="myfile" accept="text/csv"/>
992
        <button class="glob">Train</button>
993
        <section/>
994
995 </form>
996 </div>
   </BaseLayout>
997
998
999
   <script>
1000
        function sendDataToTrain(event) {
1001
            event.preventDefault();
1002
1003
```

```
let tokenToSend = '${localStorage.getItem("SavedToken")}';
1004
1005
            let fileInput = (document.getElementById("csvToTrain") as
1006
                HTMLInputElement);
        let file = fileInput.files[0];
1007
1008
        let formData = new FormData();
1009
        formData.append('file', file, file.name);
1010
          fetch('http://127.0.0.1:8000/train', {
1012
             credentials: "same-origin",
1013
            method: 'POST',
1014
            headers: {
1015
               'accept': 'application/json',
1016
               'Authorization': tokenToSend
1017
            },
1018
            body: formData
1019
          }).then(response =>
1020
           {
1021
            if (response.status == 200) {
1022
1023
1024
            }
1025
           });
1026
        }
1028
        var button = document.querySelector(".glob");
1029
     button.addEventListener('click', sendDataToTrain);
   </script>
1031
1032
1033
1034 <style>
1035
        :root {
     --color-lightest: #f9fdfe;
1036
     --color-gray-light: #cdcfcf;
1037
     --color-gray-medium: #686a69;
      --color-gray-dark: #414643;
1039
      --color-darkest: #2a2f2c;
1040
1041 }
1042
1043 div {
        text-align: center;
1044
1045 }
1046
1047 label {
        text-align: center;
1048
1049 }
1050
1051
1052 button{
        display: inline-block;
1054 }
1055 </style>
1056
1057 //signup-page.astro
```

```
1058
1059 ---
1060 const pageTitle = "Sign Up";
import '../styles/global.css';
1063 <html lang="en">
     <head>
1064
        <meta charset="utf-8" />
1065
        <link rel="icon" type="image/svg+xml" href="/favicon.svg" />
        <meta name="viewport" content="width=device-width" />
1067
        <meta name="generator" content={Astro.generator} />
1068
        <title>{pageTitle}</title>
      </head>
1070
1071
     <body>
1072
   <div class="UploadFile">
1073
1074
            <h1 style="color:black;">Sign Up</h1>
1075
            <section class="contact-form">
1076
            <form method="post">
               <div class="txt_field">
1078
                 <input class="inptClass2" id="first_name" name="first_name"</pre>
1079
                    type="text" required>
                 <span > </span >
1080
                 <label > Name </label >
1081
               </div>
1082
               <div class="txt_field">
1083
                 <input class="inptClass3" id="last_name" name="last_name"</pre>
                    type="text" required>
                 <span > </span >
1085
                 <label > Surname </label >
1086
1087
              </div>
              <div class="txt_field">
1088
                 <input id="email" name="email" type="text" required>
1089
                 <span > </span >
1090
                 <label > Email </label >
1091
1092
               <div class="txt_field">
1093
                 <input class="inptClass5" id="phone_number" name="</pre>
1094
                     phone_number" type="number" type="number" min="0" step="
                    1" max="999999999" required>
                 <span></span>
1095
                 <label>Phone Numer</label>
1096
               </div>
1097
               <div class="txt_field">
1098
                 <input id="password" name="password" type="text" required>
1099
                 <span></span>
1100
                 <label > Password </label >
1101
               </div>
1102
               <button class="glob" type="submit">Sign Up</button>
1103
               <div class="signup_link">
1104
                  <a href="/login-page">Back to login page</a>
1105
               </div>
1106
            </form>
1107
        </section>
```

```
</div>
1109
1110
          <script>
1111
            function handleFormSubmit(event) {
1112
              event.preventDefault();
1113
1114
              const data = new FormData(event.target);
1115
              const formJSON = {};
1116
              data.forEach((value, key) => {
1118
                 formJSON[key] = /^\d+$/.test(value) ? String(value) : value;
1119
              });
1120
              console.log(JSON.stringify(formJSON, null, 2));
1121
1122
              fetch('http://127.0.0.1:8000/signup', {
1123
                 credentials: "same-origin",
1124
                 method: 'POST',
1125
                 headers: {
1126
                   'accept': 'application/json',
1127
                   'Content-Type': 'application/json'
1128
                },
1129
                 body: JSON.stringify(formJSON, null, 2)
1130
              })
1131
                .then(response => response.json())
1132
1133
                .then(response => console.log(JSON.stringify(response)));
1134
            document.querySelector(".inptClass2").addEventListener("keypress
1135
                ", function (evt) {
        var charCode1 = (evt.which) ? evt.which : evt.keyCode;
1136
        if ((charCode1 >= 65 && charCode1 <= 90) || (charCode1 >= 97 &&
1137
           charCode1 <= 122)) {</pre>
1138
            // Allow the key press
        } else {
1139
            // Prevent the key press if it doesn't meet the criteria
1140
            evt.preventDefault();
1141
        }
1142
1143 });
1144 document.querySelector(".inptClass3").addEventListener("keypress",
       function (evt) {
        var charCode2 = (evt.which) ? evt.which : evt.keyCode;
1145
        if ((charCode2 >= 65 && charCode2 <= 90) || (charCode2 >= 97 &&
1146
           charCode2 <= 122)) {</pre>
1147
            // Allow the key press
        } else {
1148
            // Prevent the key press if it doesn't meet the criteria
1149
            evt.preventDefault();
1150
1151
1152 });
1153
            document.querySelector(".inptClass5").addEventListener("keypress
1154
                ", function (evt) {
        if (evt.which != 8 && evt.which != 0 && evt.which < 48 || evt.which
1155
           > 57)
        {
1156
            evt.preventDefault();
1157
```

```
1158
1159 });
             const form = document.querySelector('.contact-form');
1160
             form.addEventListener('submit', handleFormSubmit);
1161
          </script>
1162
1163
1164
1165
1166 </body>
1167 </html>
1168
1169 <style>
1170
        .center{
1171
     position: absolute;
1172
     top: 50%;
1173
     left: 50%;
1174
     transform: translate(-50%, -50%);
1175
     width: 400px;
1176
     background: white;
1177
      border-radius: 10px;
1178
      box-shadow: 10px 10px 15px rgba(0,0,0,0.05);
1179
1180 }
1181 .center h1{
1182
     text-align: center;
     padding: 20px 0;
1183
     border-bottom: 1px solid silver;
1184
1185
1186 }
1187 .center form{
      padding: 0 40px;
      box-sizing: border-box;
1189
1190 }
1191 form .txt_field{
1192
     position: relative;
      border-bottom: 2px solid #adadad;
      margin: 30px 0;
1194
1195 }
1196 .txt_field input{
     width: 100%;
1197
     padding: 0 5px;
1198
     height: 40px;
1199
1200
     font-size: 16px;
     border: none;
     background: none;
1202
     outline: none;
1203
1204 }
1205 .txt_field label{
     position: absolute;
1206
     top: 50%;
1207
     left: 5px;
1208
     color: #adadad;
1209
     transform: translateY(-50%);
1210
     font-size: 16px;
1211
1212
      pointer-events: none;
```

```
1213
   transition: .5s;
1214 }
1215 .txt_field span::before{
content: ';
position: absolute;
1218 top: 40px;
1219 left: 0;
   width: 0%;
1220
   height: 2px;
     background: #2691d9;
1222
1223 transition: .5s;
1224 }
1225 .txt_field input:focus ~ label,
1226 .txt_field input:valid ~ label{
   top: -5px;
1227
   color: #2691d9;
1228
1229 }
1230 .txt_field input:focus ~ span::before,
1231 .txt_field input:valid ~ span::before{
1232 width: 100%;
1233 }
1234 .pass{
   margin: -5px 0 20px 5px;
1235
     color: #a6a6a6;
1237
    cursor: pointer;
1238 }
1239 .pass:hover{
1240 text-decoration: underline;
1242 input[type="submit"]{
1243 width: 100%;
    height: 50px;
1245
     border: 1px solid;
   background: #2691d9;
1246
border-radius: 25px;
1248 font-size: 18px;
1249 color: #e9f4fb;
1250 font-weight: 700;
     cursor: pointer;
1251
1252
    outline: none;
1253 }
1254 input[type="submit"]:hover{
   border-color: #2691d9;
1255
    transition: .5s;
1257 }
1258 .signup_link{
   float: right;
1259
    margin: 10px 20px;
1260
1261
   text-align: center;
   font-size: 16px;
1262
    color: #666666;
1263
1264 }
1265 .signup_link a{
    color: #2691d9;
1266
1267
     text-decoration: none;
```

```
1268 }
1269 .signup_link a:hover{
    text-decoration: underline;
1271 }
1272 </style>
1274 //patients.astro
1275
1276 ---
1277 import BaseLayout from '../layouts/BaseLayout.astro';
1278 const pageTitle = "Patients";
1279 ---
1281 <script src="../scripts/checkToken.js"/>
1282 <script>
1283
1284
1285 </script>
1286
1288 <BaseLayout pageTitle={pageTitle}>
        <label >Patients historical data</label>
1289
        <section class="contact-form" >
1290
1291
            <div class="input-group">
1292
                 <label for="pregnancies">List of patients</label>
1293
            <select class="input-group" id="selectedValue" size="4">
1294
1295
            </select>
1296
            <div/>
1297
1298
            <div class="input-group">
1299
                 <label for="pregnancies">Historical data</label>
1300
            <div id="histData" class="input-group">
1301
1302
            </div>
1303
            <div/>
1304
1305
            <section/>
1306
            <script>
1307
1308
        let dataGlob;
1309
        let tokenToSend = '${localStorage.getItem("SavedToken")}';
1310
1311
        fetch('http://127.0.0.1:8000/patient', {
1312
          credentials: "same-origin",
1313
          method: 'GET',
1314
          headers: {
1315
            'accept': 'application/json',
1316
            'Authorization': tokenToSend
1317
          },
1318
1319
        }).then(response =>
1320
1321
1322
```

```
response.json().then(data => {
            const select = document.querySelector('select');
1324
            console.log(data);
1325
            dataGlob = data;
1326
                    console.log(dataGlob);
1327
                for (var i = 0; i < data.length; i++){</pre>
1328
                    let newOption = new Option('${data[i]["first_name"]} ${
1329
                        ]["id"]}');
1330
                    select.add(newOption, undefined)}
1331
            });
1332
    });
1333
                function valueChanges() {
1334
                var e = document.getElementById("selectedValue");
1335
                var value = parseInt(e.value);
1336
1337
                const historicData = document.getElementById("histData");
                historicData.innerHTML = '';
1338
                var iDiv;
1339
                var label;
1340
                var dateAndTime;
1341
                var pregn;
1342
                var gluc;
1343
                var blood;
1344
                var skin;
                var insulin;
1346
                var bmi;
1347
                var dpf;
1348
                var age;
1349
                var pred;
1350
                var hrTag;
1351
                var dateAndTimeArray = {};
                             for (var i = 0; i < dataGlob[value]["</pre>
1353
                                historical_data"].length; i++){
                                 pregn = document.createElement('div');
1354
                                 gluc = document.createElement('div');
1355
                                 blood = document.createElement('div');
1356
                                 skin = document.createElement('div');
1357
                                 insulin = document.createElement('div');
1358
                                 bmi = document.createElement('div');
1359
                                 dpf = document.createElement('div');
1360
                                 age = document.createElement('div');
1361
                                 pred = document.createElement('div');
1362
                                 label = document.createElement('label');
1363
                                 hrTag = document.createElement('hr');
1364
                                console.log(dataGlob[value]["historical_data"
1365
                                    ][i]);
                                dateAndTime = dataGlob[value]["
1366
                                    historical_data"][i]["created_at"];
                                dateAndTimeArray = dateAndTime.split("T");
1367
                                console.log(dateAndTimeArray);
1368
                                historicData.appendChild(label).innerText = '
1369
                                    Date and Time: ${dateAndTimeArray[0]} ${
                                    dateAndTimeArray[1].substring(0,8)}';
                                historicData.appendChild(pregn).innerText = '
1370
```

```
Number of pregnancies: ${dataGlob[value]["
                                    historical_data"][i]["pregnancies"]}';
                                historicData.appendChild(gluc).innerText =
1371
                                    Glucose: ${dataGlob[value]["
                                    historical_data"][i]["glucose"]}';
                                historicData.appendChild(blood).innerText = '
1372
                                    Blood pressure: ${dataGlob[value]["
                                    historical_data"][i]["blood_pressure"]}';
                                historicData.appendChild(skin).innerText = '
1373
                                    Skin thickness: ${dataGlob[value]["
                                    historical_data"][i]["skin_thickness"]}';
                                historicData.appendChild(insulin).innerText =
1374
                                     'Insulin: ${dataGlob[value]["
                                    historical_data"][i]["insulin"]}';
                                historicData.appendChild(bmi).innerText = '
1375
                                    BMI: ${dataGlob[value]["historical_data"][
                                    i]["bmi"]}';
                                historicData.appendChild(dpf).innerText = '
1376
                                    Diabetes Pedigre Function: ${dataGlob[
                                    value]["historical_data"][i]["
                                    diabetes_pedigree_function"]}';
                                historicData.appendChild(age).innerText = '
1377
                                    Age: ${dataGlob[value]["historical_data"][
                                    i]["age"]}';
                                historicData.appendChild(pred).innerText = '
1378
                                    Has Diabetes? ${dataGlob[value]["
                                    historical_data"][i]["prediction"]}';
                                historicData.appendChild(hrTag);
1379
                             }
1380
                }
1381
1382
                document.getElementById('selectedValue').addEventListener('
1383
                    change', valueChanges);
                valueChanges();
1384
            </script>
1385
   </BaseLayout>
1386
1387
   <style>
1388
1389
       #histData{
            max-height: 800px;
1390
            overflow: auto;
1391
1392
   </style>
1393
1394
   //login-page.astro
1395
1396
1397 ---
1398 import Snack from '../components/Snack.astro';
1399 const pageTitle = "Login";
1400 import '../styles/global.css';
1401 ---
1402 <html lang="en">
     <head>
1403
       <meta charset="utf-8" />
1404
       <link rel="icon" type="image/svg+xml" href="/favicon.svg" />
```

```
<meta name="viewport" content="width=device-width" />
1406
        <title>{pageTitle}</title>
1407
      </head>
1408
1409
      <body>
1410
1411 <div class="UploadFile">
1412
            <h1 style="color:black;">Login</h1>
1413
            <section class="contact-form">
            <form method="post">
1415
               <div class="txt_field">
1416
                 <input id="email" name="username" type="email" required>
1417
                 <span > </span >
1418
                 <label > Email </label >
1419
               </div>
1420
               <div class="txt_field">
1421
                 <input id="password" name="password" type="password"</pre>
                    required>
                 <span></span>
1423
                 <label > Password </label >
1424
               </div>
1425
               <button class="glob" type="submit">Login</button>
1426
1427
            </form>
1428
            </section>
1430
            <Snack id="snackbar"></Snack>
1431
1432
   <script>
1433
      function handleFormSubmit(event) {
1434
        event.preventDefault();
1435
1436
        const data = new FormData(event.target);
1437
1438
        let username_to_change = data.get("username");
1439
        let final_username = username_to_change.replace('@','%40');
1440
1441
        let body_tosent = 'grant_type=&username=${final_username}&password=$
1442
           {data.get("password")}&scope=&client_id=&client_secret=';
        fetch('http://127.0.0.1:8000/login', {
1444
          credentials: "same-origin",
1445
          method: 'POST',
1446
          headers: {
            'accept': 'application/json',
1448
            'Content-Type': 'application/x-www-form-urlencoded'
1449
          },
1450
          body: body_tosent
1451
        }).then(response =>
1452
1453
          if (response.status == 200) {
1454
                 response.json().then(data => {
1455
                     localStorage.setItem("SavedToken", "Bearer " + data["
1456
                         access_token"]);
                 });
1457
```

```
1458
                 window.location.href = "/";
1459
1460
1461
          else {
1462
            response.json().then(data => {
1463
              var snack = document.getElementById("snackbar");
1464
1465 snack.innerHTML = "";
1466 console.log(data["detail"]);
1467 snack.innerHTML += data["detail"];
1468 snack.className = "show";
1469 setTimeout(function(){ snack.className = snack.className.replace("show",
        ""); }, 3000);
                 });
1470
          }
1471
         });
1472
     }
1473
1474
     const form = document.querySelector('.contact-form');
1475
     form.addEventListener('submit', handleFormSubmit);
1477 </script>
1478
            <div class="signup_link">
1479
                 You don't have an account? <a href="/signup-page">Signup</a>
1480
1481
               </div>
          </div>
1482
1483
1484 </body>
1485 </html>
1486
1487 <style>
    .center{
1489 position: absolute;
1490 top: 50%;
1491 left: 50%;
1492 transform: translate(-50%, -50%);
1493 width: 400px;
1494 background: white;
1495 border-radius: 10px;
1496 box-shadow: 10px 10px 15px rgba(0,0,0,0.05);
1497 }
1498 .center h1{
1499 text-align: center;
1500 padding: 20px 0;
1501 border-bottom: 1px solid silver;
1502
1503 }
1504 .center form{
1505 padding: 0 40px;
1506 box-sizing: border-box;
1507 }
1508 form .txt_field{
1509 position: relative;
1510 border-bottom: 2px solid #adadad;
1511 margin: 30px 0;
```

```
1512 }
1513 .txt_field input{
1514 width: 100%;
1515 padding: 0 5px;
1516 height: 40px;
1517 font-size: 16px;
1518 border: none;
1519 background: none;
1520 outline: none;
1521 }
1522 .txt_field label{
1523 position: absolute;
1524 top: 50%;
1525 left: 5px;
1526 color: #adadad;
1527 transform: translateY(-50%);
1528 font-size: 16px;
1529 pointer-events: none;
1530 transition: .5s;
1531 }
1532 .txt_field span::before{
1533 content: '';
1534 position: absolute;
1535 top: 40px;
1536 left: 0;
1537 width: 0%;
1538 height: 2px;
1539 background: #2691d9;
1540 transition: .5s;
1542 .txt_field input:focus ~ label,
1543 .txt_field input:valid ~ label{
1544 top: -5px;
1545 color: #2691d9;
1546 }
1547 .txt_field input:focus ~ span::before,
1548 .txt_field input:valid ~ span::before{
1549 width: 100%;
1550 }
1551 .pass{
1552 margin: -5px 0 20px 5px;
1553 color: #a6a6a6;
1554 cursor: pointer;
1555 }
1556 .pass:hover{
1557 text-decoration: underline;
1558 }
1560 .signup_link{
1561 margin: 30px 0;
1562 text-align: center;
1563 font-size: 16px;
1564 color: #666666;
1565 }
1566 .signup_link a{
```

```
1567 color: #2691d9;
1568 text-decoration: none;
1569 }
1570 .signup_link a:hover{
1571 text-decoration: underline;
1572 }
1573
1574 </style>
1576 /index.astro
1577
1578 ---
1579 import BaseLayout from '../layouts/BaseLayout.astro';
1580 const pageTitle = "Home";
1581 ---
1582
   <script src="../scripts/checkToken.js"/>
1583
1584
   <script>
1585
        let tokenToSend = '${localStorage.getItem("SavedToken")}';
1586
1587
            fetch('http://127.0.0.1:8000/patient', {
1588
               credentials: "same-origin",
1589
               method: 'GET',
1590
1591
               headers: {
                 'accept': 'application/json',
1592
                 'Authorization': tokenToSend
1593
               },
1594
1595
            }).then(response =>
1596
             {
1597
1598
                 response.json().then(data => {
1599
                      const select = document.querySelector('select');
1600
                      let defOption = new Option("----", null);
1601
                      console.log(data[1]);
1602
                      select.add(defOption, undefined);
1603
                          for (var i = 0; i < data.length; i++){</pre>
1604
                               let newOption = new Option('${data[i]["
1605
                                  first_name"]} ${data[i]["last_name"]} ${data[
                                  i]["PESEL"]}','${data[i]["id"]}');
1606
                               select.add(newOption, undefined)}
1607
                     });
1608
             });
1609
        </script>
1610
   <BaseLayout pageTitle={pageTitle}>
1612
1613
        <label >Predict </label>
1614
            <section class="contact-form" >
1615
                 <div class="input-group">
1616
                      <label for="pregnancies">List of patients</label>
1617
                 <select class="input-group" id="selectedValue" size="4">
1618
1619
```

```
</select>
1620
            </div>
1621
                 <form>
1622
1623
                     <div class="input-group">
1624
                          <label for="pregnancies">Pregnancies</label>
1625
                          <input class="inptClass" id="pregnancies" name="</pre>
1626
                             pregnancies" type="number" min="0" required/>
                     </div>
1627
1628
                     <div class="input-group">
1629
                          <label for="Glucose">Glucose</label>
1630
                          <input class="inptClass1" id="Glucose" name="glucose</pre>
1631
                              " type="number" min="0" required/>
                     </div>
1632
1633
1634
                     <div class="input-group">
                          <label for="name">Blood Pressure</label>
1635
                          <input class="inptClass2" id="name" name="</pre>
1636
                             blood_pressure" type="number" min="0" required/>
                     </div>
1637
1638
                     <div class="input-group">
1639
                          <label for="SkinThicc">Skin Thickness</label>
1640
                          <input class="inptClass3" id="SkinThicc" name="</pre>
1641
                              skin_thickness" type="number" min="0" required/>
                     </div>
1642
                     <div class="input-group">
1644
                          <label for="Insulin">Insulin</label>
1645
                          <input class="inptClass4" id="Insulin" name="insulin</pre>
1646
                             " type="number" min="0" required/>
                     </div>
1647
1648
                     <div class="input-group">
1649
                          <label for="BMI">BMI</label>
1650
                          <input class="inptClass5" id="BMI" name="bmi" type="</pre>
1651
                             number" min="0" required/>
                     </div>
1652
1653
                     <div class="input-group">
1654
                          <label for="dpf">Diabetes Pedigree Function</label>
1655
                          <input class="inptClass6" id="dpf" name="</pre>
1656
                              diabetes_pedigree_function" type="number" min="0"
                               required/>
                     </div>
1657
1658
                     <div class="input-group">
1659
                          <label for="age">Age</label>
1660
                          <input class="inptClass7" id="age" name="age" type="</pre>
1661
                             number" min="1" step="1" required/>
                     </div>
1662
1663
                     <div class="input-group">
1664
                          <button class="glob" type="submit">Predict</button>
```

```
<div class="pred"> </div>
1666
                     </div>
1667
1668
                 </form>
1669
            </section>
1670
   <!--
1671
            <div class="results">
1672
                 <h3>Form Data</h3>
1673
                 1674
            </div>
1675
        _ _ >
1676
1677
1678
        <script>
1679
            var predVal;
1680
            function handleFormSubmit(event) {
1681
1682
               event.preventDefault();
              let tokenToSend = '${localStorage.getItem("SavedToken")}';
1683
              var e = document.getElementById("selectedValue");
1684
              var value = parseInt(e.value);
1685
              var predd = document.querySelector('.pred');
1686
              const data = new FormData(event.target);
1687
              const formJSON = {};
1688
              const dataJSON = {};
1689
1690
              var y;
              var sth;
1691
              var isIt;
1692
               console.log(data);
1693
              data.forEach((value, key) => {
1694
                 dataJSON[key] = /^\d+$/.test(value) ? parseFloat(value) :
1695
                    value;
1696
              });
               formJSON["patient_id"] = value;
1697
               formJSON["input"] = dataJSON;
1698
               console.log(JSON.stringify(formJSON, null, 2));
1699
               fetch('http://127.0.0.1:8000/predict', {
1700
                 credentials: "same-origin",
1701
                 method: 'POST',
1702
                 headers: {
1703
                   'accept': 'application/json',
1704
                   'Authorization': tokenToSend,
1705
                   'Content-Type': 'application/json'
1706
                 },
1707
                 body: JSON.stringify(formJSON, null, 2)
1708
              })
1709
                .then(response => {
1710
                 response.json().then(data => {
1711
                     console.log(data);
1712
                     predd.innerHTML = '';
1713
                     sth = parseInt(data.prediction);
1714
                     if (sth != 0 ) {
1715
                          isIt = "Patient doesn't have diabetes";
1716
                     }
1717
                     else{
1718
                          isIt = "Patient does have diabetes";
1719
```

```
}
1720
               y = document.createTextNode(isIt);
1721
               predd.appendChild(y);
1722
               });
1723
            });
1724
1725
1726
            document.querySelector(".inptClass").addEventListener("keypress"
1727
                , function (evt) {
        if (evt.which != 8 && evt.which != 8 && evt.which != 0 && evt.which
1728
           < 48 \mid \mid evt.which > 57
        {
1729
            evt.preventDefault();
1730
        }
1731
1732 });
1733
   document.querySelector(".inptClass1").addEventListener("keypress",
1734
       function (evt) {
        if (evt.which != 8 && evt.which != 0 && evt.which == 46 && evt.which
1735
            < 48 || evt.which > 57)
1736
            evt.preventDefault();
1737
        }
1738
1739 });
1740 document.querySelector(".inptClass2").addEventListener("keypress",
       function (evt) {
        if (evt.which != 8 && evt.which != 0 && evt.which == 46 && evt.which
1741
            < 48 \mid \mid evt.which > 57
1742
            evt.preventDefault();
1743
        }
1744
1745 });
1746 document.querySelector(".inptClass3").addEventListener("keypress",
       function (evt) {
        if (evt.which != 8 && evt.which != 0 && evt.which == 46 && evt.which
1747
            < 48 \mid \mid evt.which > 57)
        {
1748
            evt.preventDefault();
1749
        }
1750
1751 });
1752 document.querySelector(".inptClass4").addEventListener("keypress",
       function (evt) {
        if (evt.which != 8 && evt.which != 0 && evt.which == 46 && evt.which
1753
            < 48 \mid \mid evt.which > 57
        {
1754
            evt.preventDefault();
1755
1756
1757 });
1758 document.querySelector(".inptClass5").addEventListener("keypress",
       function (evt) {
        if (evt.which != 8 && evt.which != 0 && evt.which == 46 && evt.which
1759
            < 48 || evt.which > 57)
        {
1760
            evt.preventDefault();
1761
        }
1762
```

```
1763 });
1764
1765 document.querySelector(".inptClass6").addEventListener("keypress",
       function (evt) {
        if (evt.which != 8 && evt.which != 0 && evt.which == 46 && evt.which
1766
            < 48 \mid \mid evt.which > 57
1767
            evt.preventDefault();
1768
        }
1769
1770 });
1771
1772 document.querySelector(".inptClass7").addEventListener("keypress",
       function (evt) {
        if (evt.which != 8 && evt.which != 0 && evt.which < 48 || evt.which
1773
           > 57)
        {
1774
            evt.preventDefault();
1776
1777 });
1778
            const form = document.querySelector('.contact-form');
1779
            form.addEventListener('submit', handleFormSubmit);
1780
          </script>
1781
1782 </BaseLayout>
1784 <style>
        .pred{
            float: right;
1787
1788 </style>
1789
1790
   //add-patient.astro
1791
1792 ---
1793 import Snack from '../components/Snack.astro';
1794 import BaseLayout from '.../layouts/BaseLayout.astro';
1795 const pageTitle = "Add Patient";
1796 ---
1797 <script src="../scripts/checkToken.js"/>
   <BaseLayout pageTitle={pageTitle}>
1799
        <label >Add patient </label>
1800
        <section class="contact-form">
1801
            <form>
1802
1803
                 <div class="input-group">
1804
                     <label for="PESEL">PESEL</label>
1805
                     <input class="inptClass1" id="PESEL" name="PESEL" type="</pre>
1806
                         number" min="0" step="1" required/>
                 </div>
1807
1808
                 <div class="input-group">
1809
                     <label for="first_name">Name</label>
1810
                     <input class="inptClass2" id="first_name" name="</pre>
1811
                         first_name" type="text" required/>
```

```
</div>
1812
1813
                 <div class="input-group">
1814
                     <label for="last_name">Surname</label>
1815
                     <input class="inptClass3" id="last_name" name="last_name</pre>
1816
                         " type="text" required/>
                 </div>
1817
1818
                 <div class="input-group">
                     <label for="email">Email</label>
1820
                     <input class="inptClass4" id="email" name="email" type="</pre>
1821
                         email" required/>
                 </div>
1822
1823
                 <div class="input-group">
1824
                     <label for="phone_number">Phone Number</label>
1825
                     <input class="inptClass5" id="phone_number" name="</pre>
1826
                         phone_number" type="number" min="0" step="1" max="
                         999999999" required/>
                 </div>
1827
1828
                 <button class="glob" type="submit">Add</button>
1829
            </form>
1830
        </section>
1831
        <Snack id="snackbar"></Snack>
1832
        <script>
1833
            function handleFormSubmit(event) {
1834
              event.preventDefault();
1836
              const data = new FormData(event.target);
1837
                   const formJSON = {};
1838
                   formJSON["id"] = null;
1839
                   data.forEach((value, key) => {
1840
                     formJSON[key] = /^\d+$/.test(value) ? String(value) :
1841
                         value;
                   });
                   formJSON["historical_data"] = null;
1843
                   console.log(JSON.stringify(formJSON, null, 2));
1844
1845
              let tokenToSend = '${localStorage.getItem("SavedToken")}';
1846
1847
              fetch('http://127.0.0.1:8000/patient', {
1848
                 credentials: "same-origin",
1849
                 method: 'POST',
1850
                 headers: {
1851
                   'accept': 'application/json',
1852
                   'Authorization': tokenToSend,
1853
                   'Content-Type': 'application/json'
1854
                 },
1855
                 body: JSON.stringify(formJSON, null, 2)
1856
              }).then(response =>
1857
1858
          if (response.status == 404) {
1859
            response.json().then(data => {
1860
              var snack = document.getElementById("snackbar");
```

```
1862 snack.innerHTML = "";
1863 console.log(data["detail"]);
1864 snack.innerHTML += data["detail"];
1865 snack.className = "show";
1866 setTimeout(function(){ snack.className = snack.className.replace("show",
       ""); }, 3000);
                });
1867
1868
          if (response.status == 200) {
1869
            response.json().then(data => {
1870
              var snack = document.getElementById("snackbar");
1871
1872 snack.innerHTML = "";
1873 snack.innerHTML += "Patient added correctly";
1874 snack.className = "show";
   setTimeout(function(){ snack.className = snack.className.replace("show",
        ""); }, 3000);
1876
                });
         }
1877
        });
1878
       };
1879
       document.querySelector(".inptClass1").addEventListener("keypress",
1880
           function (evt) {
       if (evt.which != 8 && evt.which != 0 && evt.which < 48 || evt.which
1881
           > 57)
1882
            evt.preventDefault();
1883
       }
1884
1885 });
1886 document.querySelector(".inptClass2").addEventListener("keypress",
       function (evt) {
       var charCode1 = (evt.which) ? evt.which : evt.keyCode;
1887
       if ((charCode1 >= 65 && charCode1 <= 90) || (charCode1 >= 97 &&
1888
           charCode1 <= 122)) {</pre>
            // Allow the key press
1889
       } else {
1890
            // Prevent the key press if it doesn't meet the criteria
1891
            evt.preventDefault();
1892
       }
1893
1894 });
   document.querySelector(".inptClass3").addEventListener("keypress",
       function (evt) {
       var charCode2 = (evt.which) ? evt.which : evt.keyCode;
1896
       if ((charCode2 >= 65 && charCode2 <= 90) || (charCode2 >= 97 &&
1897
           charCode2 <= 122)) {</pre>
            // Allow the key press
1898
       } else {
1899
            // Prevent the key press if it doesn't meet the criteria
1900
            evt.preventDefault();
1901
       }
1902
1903 });
1905 document.querySelector(".inptClass5").addEventListener("keypress",
       function (evt) {
       if (evt.which != 8 && evt.which != 0 && evt.which < 48 || evt.which
1906
           > 57)
```

```
1907
            evt.preventDefault();
1908
1909
1910 });
            const form = document.querySelector('.contact-form');
1911
            form.addEventListener('submit', handleFormSubmit);
1912
          </script>
1913
1914
1915 </BaseLayout>
1917 //BaseLayout.astro
1918
1920 import Header from '../components/Header.astro';
import '../styles/global.css';
1922 const { pageTitle } = Astro.props;
1924 <html lang="en">
     <head>
1925
       <meta charset="utf-8" />
1926
       <link rel="icon" type="image/svg+xml" href="/favicon.svg" />
1927
       <meta name="viewport" content="width=device-width" />
1928
       <meta name="generator" content={Astro.generator} />
1929
       <title>{pageTitle}</title>
1930
1931
     </head>
1932
     <body>
1933
       <Header/>
       <div class="UploadFile">
1935
       <slot>
1936
1937 </div>
1938
1939
     </body>
1940 </html>
1941
1942 <style>
1943 body {
       overflow: scroll;
1944
1945 }
1946 </style>
1947
1948 //ThemeIcon.astro
1949
1950 ---
1952 <button id="themeToggle">
       <svg width="30px" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24</pre>
1953
            24">
          <path class="sun" fill-rule="evenodd" d="M12 17.5a5.5 5.5 0 1 0</pre>
1954
             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 11-1.8 1.8a.8.8 0 0 1-1-111.7-1.8a
             .8.8 0 0 1 1 0zM6.3 17.7a.8.8 0 0 1 0 11-1.7 1.8a.8.8 0 1 1-1-1
             11.7-1.8a.8.8 0 0 1 1 0zM12 0a.8.8 0 0 1 .8.8v2.5a.8.8 0 0
```

```
1-1.6 OV.8A.8.8 O O 1 12 OzmO 20a.8.8 O O 1 .8.8v2.4a.8.8 O O
             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
             11.8 1.7a.8.8 0 0 1-1 11-1.8-1.7a.8.8 0 0 1 0-1z"/>
          <path class="moon" fill-rule="evenodd" d="M16.5 6A10.5 10.5 0 0 1</pre>
1955
             4.7\ 16.4\ 8.5\ 8.5\ 0\ 1\ 0\ 16.4\ 4.71.1\ 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>
1956
      </button>
1957
1958
     <style>
1959
        #themeToggle {
1960
          border: 0;
1961
1962
          background: none;
1963
          cursor: pointer;
        }
1964
        .sun { fill: white; }
1965
        .moon { fill: transparent; }
1966
1967
       button {
1968
            float: right;
1969
1970
            margin-top: 5px;
            display:inline-block
1972
1973
        :global(.dark) .sun { fill: transparent; }
1974
        :global(.dark) .moon { fill: white; }
1975
      </style>
1976
1977
   <script is:inline>
        const theme = (() \Rightarrow \{
1979
          if (typeof localStorage !== 'undefined' && localStorage.getItem('
1980
             theme')) {
            return localStorage.getItem('theme');
1981
          }
1982
          if (window.matchMedia('(prefers-color-scheme: dark)').matches) {
1983
            return 'dark';
1984
1985
            return 'light';
1986
        })();
1987
1988
        if (theme === 'light') {
1989
          document.documentElement.classList.remove('dark');
1990
       } else {
1991
          document.documentElement.classList.add('dark');
1992
1993
1994
        window.localStorage.setItem('theme', theme);
1995
1996
        const handleToggleClick = () => {
1997
          const element = document.documentElement;
1998
          element.classList.toggle("dark");
1999
2000
```

```
const isDark = element.classList.contains("dark");
          localStorage.setItem("theme", isDark ? "dark" : "light");
2002
2003
2004
        document.getElementById("themeToggle").addEventListener("click",
2005
           handleToggleClick);
     </script>
2006
2007
     //Snack.astro
2008
2009 ---
2010 ---
2011
2012 <div id="snackbar"></div>
2014 <style>
      #snackbar {
2015
    visibility: hidden;
2016
2017
    min-width: 250px;
   margin-left: -125px;
2018
    background-color: #333;
2019
    color: #fff;
2020
   text-align: center;
2021
    border-radius: 2px;
2022
     padding: 16px;
2023
     position: fixed;
2024
     z-index: 1;
2025
     left: 50%;
2026
2027
     bottom: 30px;
2028 }
2029
2030 #snackbar.show {
    visibility: visible;
2032
     -webkit-animation: fadein 0.5s, fadeout 0.5s 2.5s;
     animation: fadein 0.5s, fadeout 0.5s 2.5s;
2033
2034 }
2035
2036 @-webkit-keyframes fadein {
    from {bottom: 0; opacity: 0;}
     to {bottom: 30px; opacity: 1;}
2038
2039 }
2040
2041 Okeyframes fadein {
2042
     from {bottom: 0; opacity: 0;}
     to {bottom: 30px; opacity: 1;}
2044 }
2045
2046 @-webkit-keyframes fadeout {
     from {bottom: 30px; opacity: 1;}
2048
     to {bottom: 0; opacity: 0;}
2049 }
2050
2051 Okeyframes fadeout {
    from {bottom: 30px; opacity: 1;}
     to {bottom: 0; opacity: 0;}
2053
2054 }
```

```
2055 </style>
2056
   //ScriptSnack.jsx
2057
2058 import { useState } from 'preact/hooks';
2060 export default function Snack({messages}) {
2061
      const randomMessage = () => messages[(Math.floor(Math.random() *
2062
         messages.length))];
2063
     const [greeting, setGreeting] = useState(messages[0]);
2064
2065
     return (
2066
       <div>
2067
          <h3>{greeting}! Thank you for visiting!</h3>
2068
          <button onClick={() => setGreeting(randomMessage())}>
2069
            New Greeting
          </button>
2071
        </div>
2072
     );
2073
2074 }
2075
2076
2077 var snack = document.getElementById("snackbar");
2078 snack.innerHTML = "";
2079 console.log(data["detail"]);
2080 snack.innerHTML += data["detail"];
2081 snack.className = "show";
2082 setTimeout(function(){ snack.className = snack.className.replace("show",
        ""); }, 3000);
2083
2084
   //Header.astro
2085
2086 ---
2087 import ThemeIcon from './ThemeIcon.astro';
2089 <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font</pre>
       -awesome/4.7.0/css/font-awesome.min.css">
2090 <header>
     <nav>
2091
2092
          <a class="log-out"><i class="fa fa-sign-out" style="font-size:36px</pre>
2093
             ; color: white "></i></a>
          <div class="nav-container">
            <a class="link-to-pages" href="/add-patient">Add Patient</a>
2095
            <a class="link-to-pages" href="/">Predict</a>
2096
            <a class="link-to-pages" href="/patients">All Patients</a>
2097
            <a class="link-to-pages" href="/train">Train</a>
2098
          </div>
2099
          <ThemeIcon />
2100
     </nav>
2101
2102
2103 </header>
2104
2105 <script>
```

```
const button = document.querySelector(".log-out");
2106
2107
     button.addEventListener("click", (event) => {
2108
       localStorage.removeItem("SavedToken");
2109
       window.location.href = "/login-page";
2110
2111
     });
2112
            </script>
2113
2114
2115 <style>
     a.log-out{
2116
      float: left;
2117
     .nav-container{
2119
       display: inline-block;
2120
2121
    a.link-to-pages{
2123
     border-style: none none solid none;
     border-radius: 4px;
2124
     margin: Opx 5px Opx 5px;
2125
2126
      display:inline-block;
    }
2127
2128 </style>
```