

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/>)

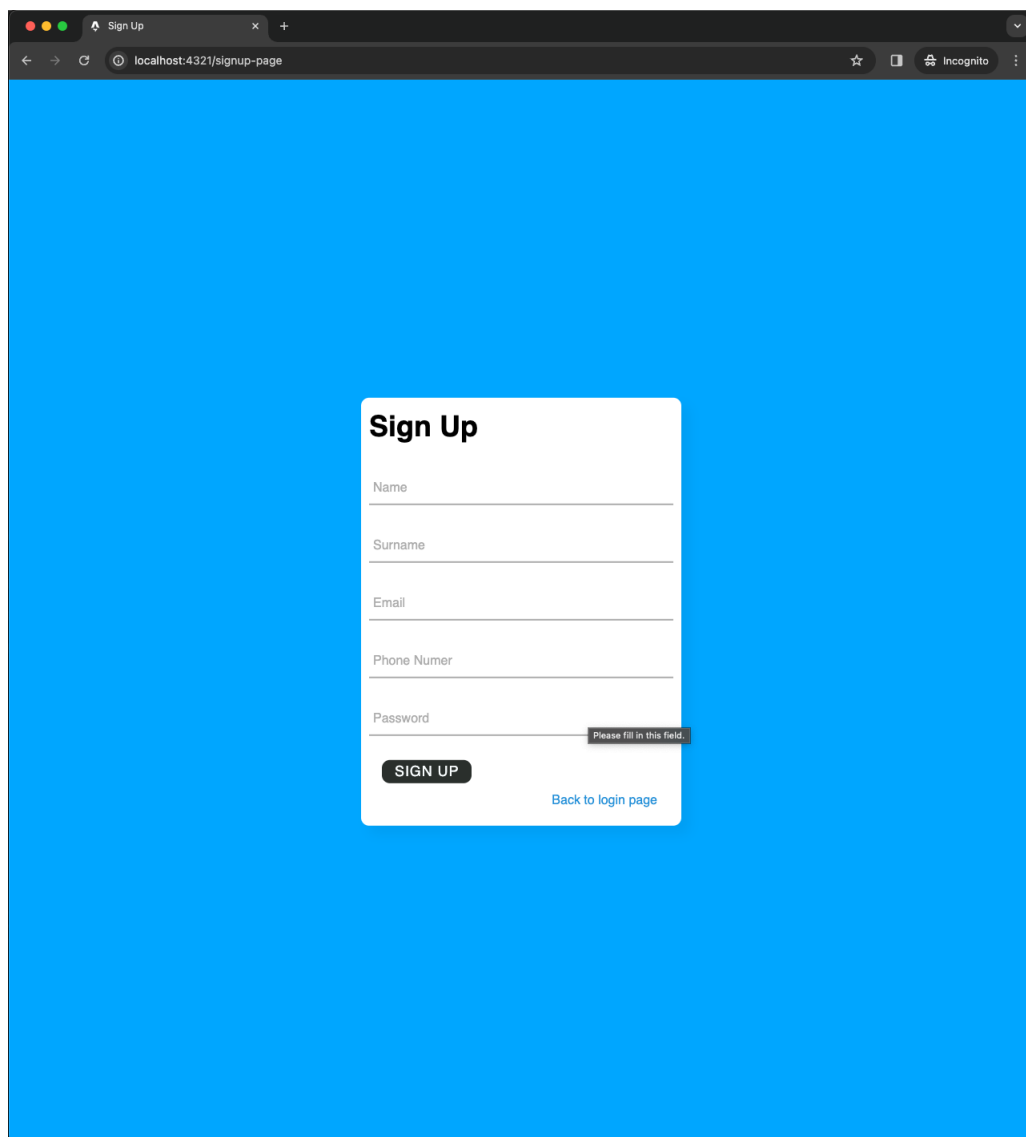
Ściągamy kod źródłowy i wypakowujemy w dowolny miejscu na dysku. Jeżeli Docker jest zainstalowany uruchamiamy konsolę i z poziomu głównego folderu naszej aplikacji wykonujemy następujące komendy:

- '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"

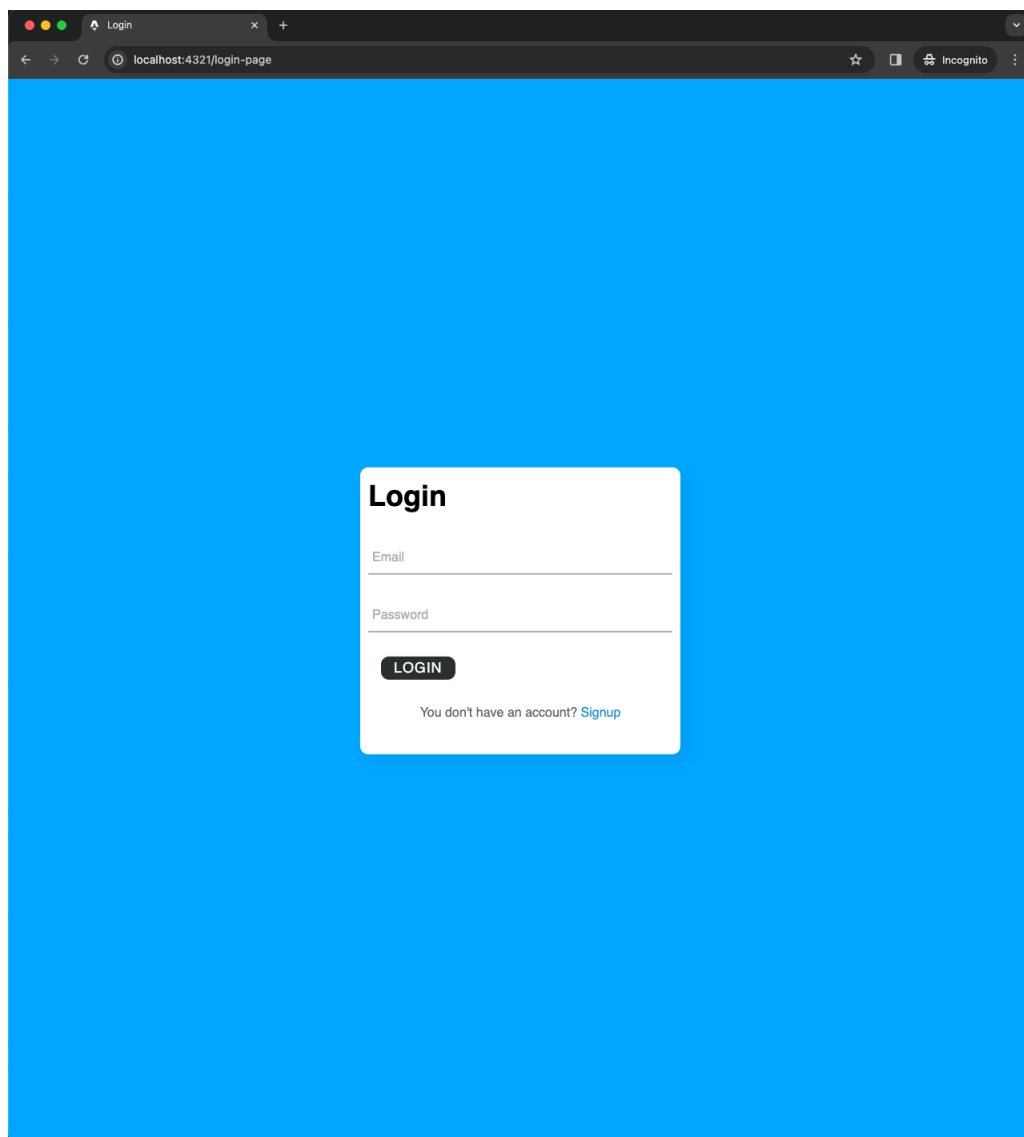
Aby zalogować się, do obu aplikacji domyślnym kontem adminastratora jest:
email: senior_registrar@hospital.com
hasło: passwd

Gdy obie aplikacje działają mamy dostępne możliwości jak poniżej.



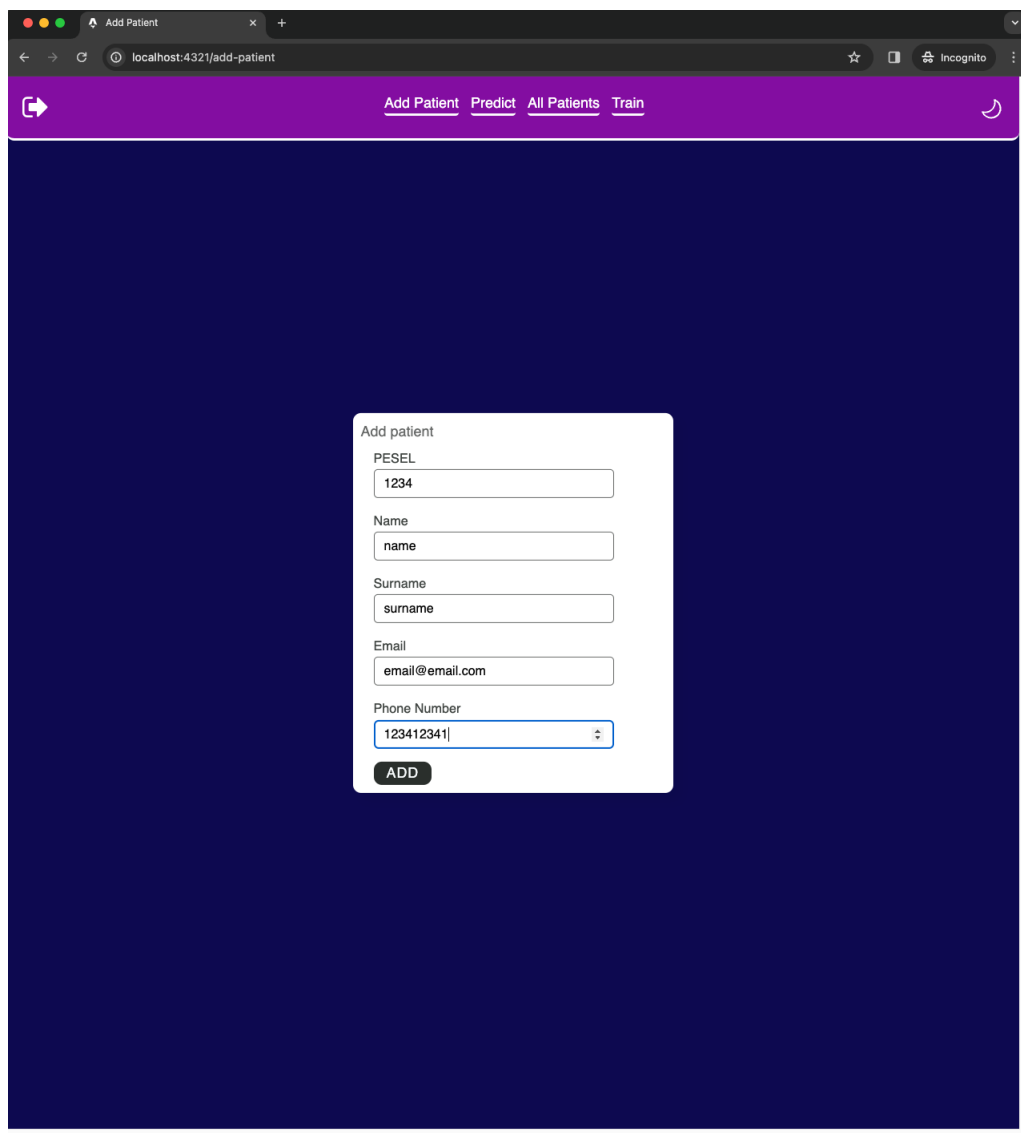
Rysunek 1: Możemy dodać lekarza

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.

Home x +

localhost:4321

Incognito

Add Patient Predict All Patients Train

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

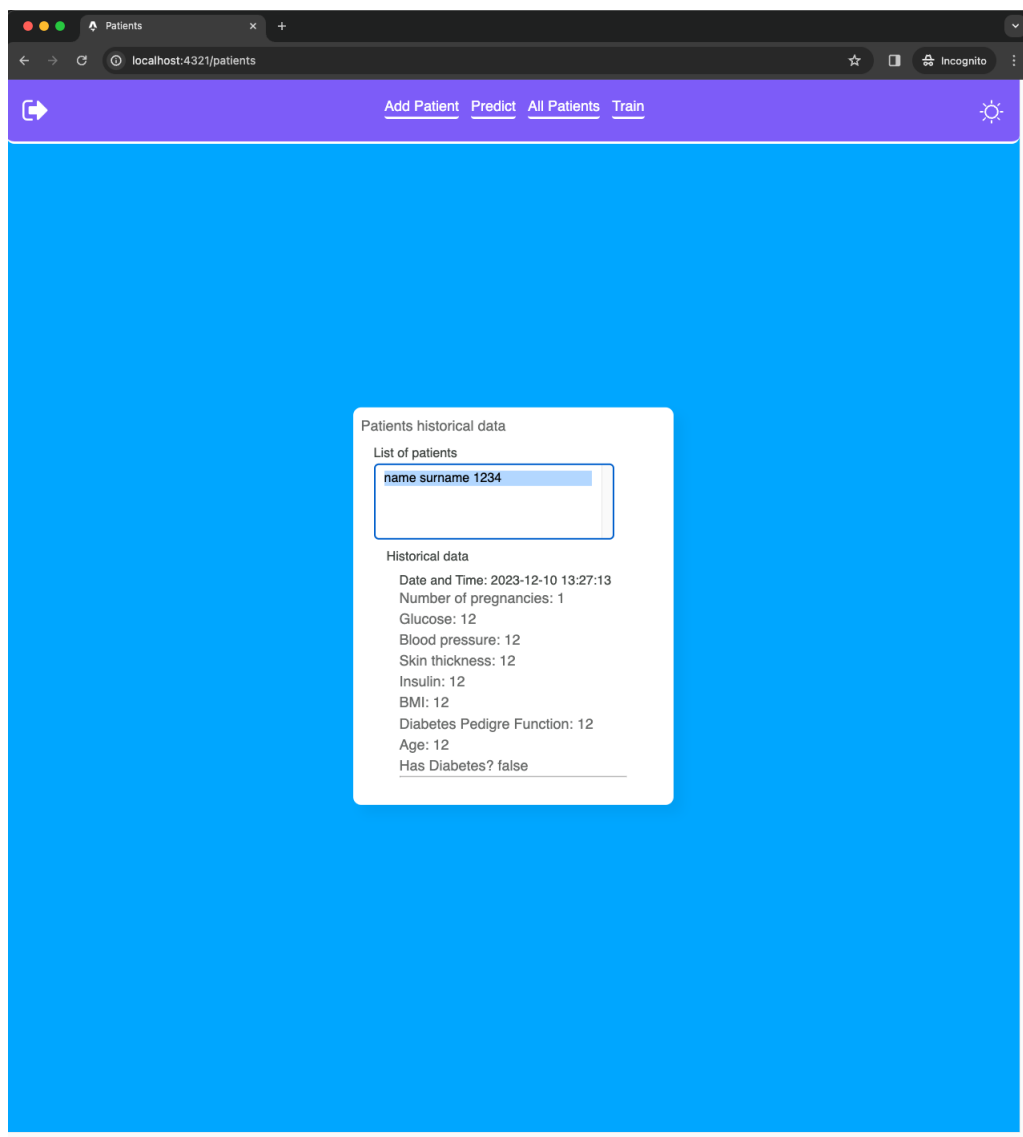
12

PREDICT

Patient does have diabetes

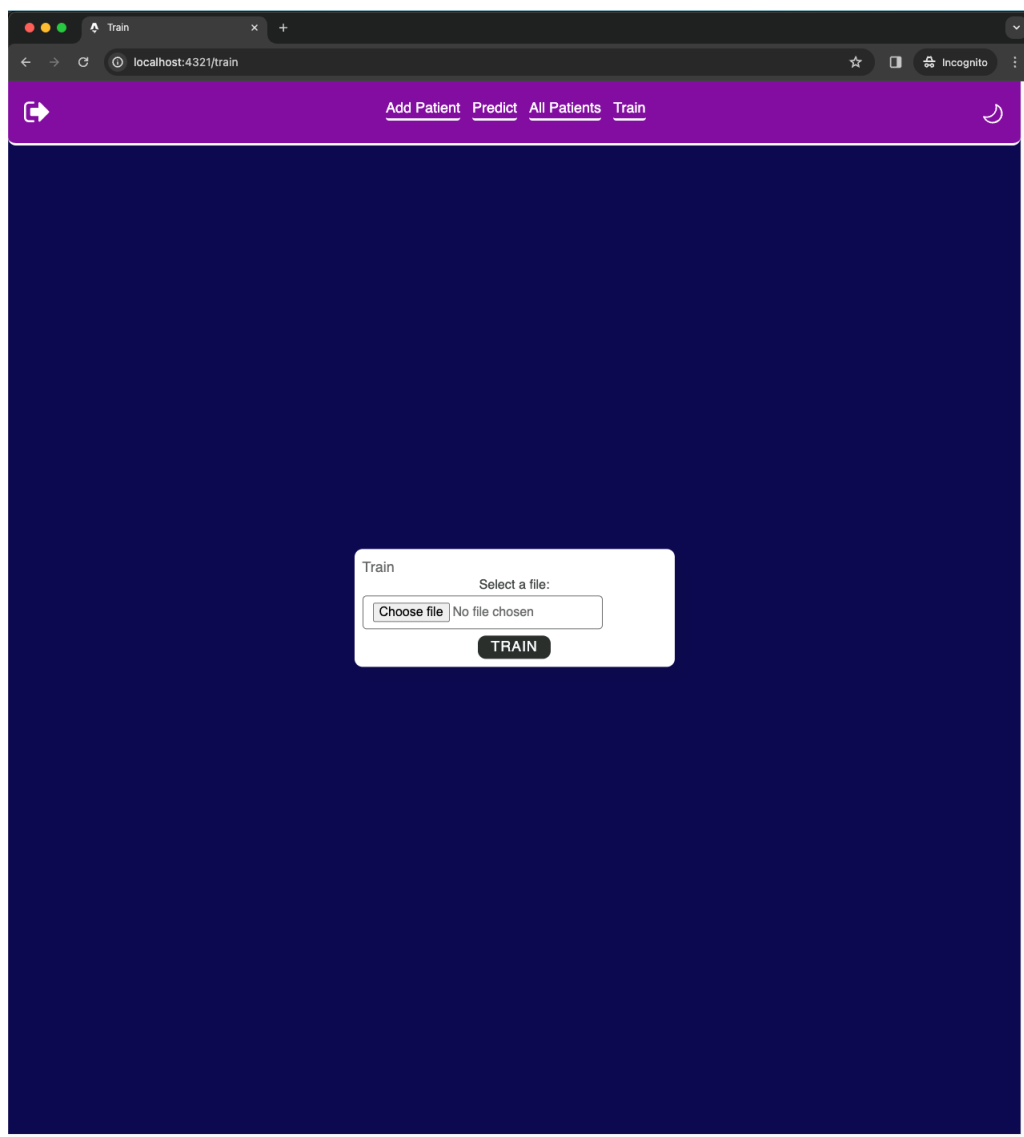
Rysunek 4: Wybieramy pacjenta bądź pole puste, aby przetestować bez zapisywania danych do pacjenta

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



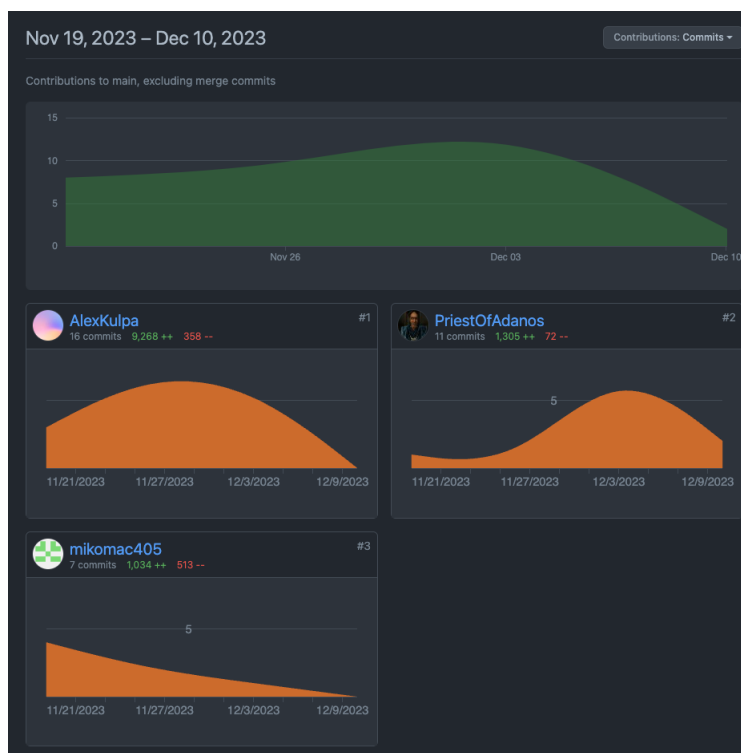
Rysunek 5: Wybierając z listy pacjenta możemy sprawdzić jego historyczne dane predykcji

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

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

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 - zawiera listę kont wraz z zaszyfrowanymi hasłami algorytmem bcrypt

Wszystkie table (łącznie z nazwą tabeli i kolumn) są zaszyfrowane algorytmem 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 definicje 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

```

1      def _setup_database_object(self) -> None:
2          try:
3              self.db = SqliteCipher(
4                  dataBasePath=os.getenv("DB_PATH"),
5                  checkSameThread=False,
6                  password=os.getenv("DB_PASSWORD"),
7              )
8          except Exception as ex:
9              logging.error(ex)
10
11         for table_name in ["diabetes", "doctors", "patients"]:
12             if not self.db.checkTableExist(table_name):
13                 self._init_database()
14                 break
15
16     def _init_database(self) -> None:
17         self.db.createTable(
18             "diabetes",
19             [
20                 ["pregnancies", "INT"],
21                 ["glucose", "REAL"],
22                 ["blood_pressure", "REAL"],
23                 ["skin_thickness", "REAL"],
24                 ["insulin", "REAL"],
25                 ["bmi", "REAL"],
26                 ["diabetes_pedigree_function", "REAL"],
27                 ["age", "INT"],
28                 ["outcome", "INT"],
29             ],
30             True,
31             True,
32         )
33
34         self.db.createTable(
35             "doctors",
36             [
37                 ["first_name", "TEXT"],
38                 ["last_name", "TEXT"],
39                 ["email", "TEXT"],
40                 ["phone_number", "TEXT"],
41                 ["hashed_password", "TEXT"],
42             ],
43             True,

```

```

44         True,
45     )
46
47     self.db.insertIntoTable(
48         "doctors",
49         [
50             "Senior",
51             "Registrar",
52             "senior_registrar@hospital.com",
53             "+48-111-222-333",
54             password_utils.get_hashed_password(os.getenv("
                    DEV_PASSWORD"))),
55         ],
56         True,
57     )
58
59     self.db.createTable(
60         "patients",
61         [
62             ["PESEL", "TEXT"],
63             ["first_name", "TEXT"],
64             ["last_name", "TEXT"],
65             ["email", "TEXT"],
66             ["phone_number", "TEXT"],
67             ["historical_data", "BLOB"],
68         ],
69         True,
70         True,
71     )

```

Elementy pomocnicze

Pliki `db_models.py`, `jwt_utils.py` oraz `password_utils.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 bezpieczne z punktu widzenia api w sposób ciągły

Pełen kod aplikacji

```
1 //test.py
2
3 import requests
4 port = 8000
5 def test_train_model_endpoint(token):
6     headers = {
7         'accept': 'application/json',
8         'Authorization': f'Bearer {token}',
9         # requests won't add a boundary if this header is set when you
10         # pass files=
11         # 'Content-Type': 'multipart/form-data',
12     }
13
14     files = {
15         'file': ('diabetes.csv', open('diabetes.csv', 'rb'), 'text/csv')
16     }
17
18     response = requests.post('http://localhost:8000/train', headers=
19                               headers, files=files)
20     assert response.status_code == 200
21     print("train 200")
22
23 def get_login():
24     headers = {
25         'accept': 'application/json',
26         'Content-Type': 'application/x-www-form-urlencoded',
27     }
28
29     data = {
30         'grant_type': '',
31         'username': 'senior_registrar@hospital.com',
32         'password': 'passwd',
33         'scope': '',
34         'client_id': '',
35         'client_secret': '',
36     }
37
38     response = requests.post('http://localhost:8000/login', headers=
39                               headers, data=data)
40
41     return response
42
43 def test_predict_diabetes_endpoint(token):
44     headers = {
45         'accept': 'application/json',
46         'Authorization': f'Bearer {token}',
47         'Content-Type': 'application/json',
48     }
49
50     json_data = {
51         'patient_id': None,
```

```

49         'input': {
50             'pregnancies': 0,
51             'glucose': 0,
52             'blood_pressure': 0,
53             'skin_thickness': 0,
54             'insulin': 0,
55             'bmi': 0,
56             'diabetes_pedigree_function': 0,
57             'age': 0,
58         },
59     }
60
61     response = requests.post(f'http://localhost:{port}/predict', headers
62                             =headers, json=json_data)
63     assert response.status_code == 200
64     print("predict: 200")
65
66 def test_get_data_endpoint(token):
67     headers = {
68         'accept': 'text/html',
69         'Authorization': f'Bearer {token}',
70     }
71
72     response = requests.get(f'http://localhost:{port}/data', headers=
73                             headers)
74     assert response.status_code == 200
75     print("data: 200")
76
77 def test_signup_endpoint(token):
78     print("signup")
79     url = f"http://localhost:{port}/signup"
80     # Replace this with the appropriate user data for your endpoint
81     user_data = {
82         "email": "test@example.com",
83         "password": "yourpassword"
84     }
85     response = requests.post(url, json=user_data)
86     assert response.status_code == 200
87     token = get_login().json().get("access_token")
88     test_get_data_endpoint(token)
89     test_predict_diabetes_endpoint(token)
90     test_train_model_endpoint(token)
91
92 //app.py
93
94 import datetime
95 from typing import List
96 from fastapi import FastAPI, HTTPException, UploadFile, Depends
97 from fastapi.responses import HTMLResponse
98 from fastapi.middleware.cors import CORSMiddleware
99 import pandas as pd
100 from fastapi.security import OAuth2PasswordRequestForm
101 from sklearn.model_selection import train_test_split
102 from sklearn.linear_model import LogisticRegression
103 from dotenv import load_dotenv

```



```

102 import joblib
103 from kerastuner.tuners import RandomSearch
104 from keras_tuner import HyperModel
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
118
119 from jwt_utils import (
120     create_access_token,
121     create_refresh_token,
122     get_current_user,
123 )
124 from db_models import (
125     UserRegister,
126     TokenSchema,
127     User,
128     Patient,
129     PredictionInput,
130     DiabetesHistoricalOutput,
131 )
132 from password_utils import verify_password
133
134 load_dotenv()
135 db = DatabaseManager()
136 app = FastAPI()
137 app.add_middleware(
138     CORSMiddleware,
139     allow_origins=["*"],
140     allow_credentials=True,
141     allow_methods=["*"],
142     allow_headers=["*"],
143 )
144
145 model_name = "model"
146
147 class MyHyperModel(HyperModel):
148     def __init__(self, input_shape):
149         self.input_shape = input_shape
150
151     def build(self, hp):
152         model = Sequential()
153         model.add(Dense(units=hp.Int('units', min_value=32, max_value
154                                     =512, step=32),
155                         activation='relu', input_shape=self.input_shape)
156         )

```

```

155         model.add(Dense(1, activation='linear'))
156         model.compile(optimizer='adam', loss='mean_squared_error')
157         return model
158
159 @app.post("/train")
160 async def train_model(file: UploadFile, user: User = Depends(
    get_current_user)):
161     db.insert_diabetes_data(file.file)
162
163     df = db.select_all_diabetes_data()
164
165     X = df.iloc[:, :-1]
166     y = df.iloc[:, -1]
167     X_train, X_test, y_train, y_test = train_test_split(
168         X, y, test_size=0.2, random_state=42
169     )
170     input_size = X_train.shape[1] # Number of features
171
172     model = Sequential()
173     model.add(Dense(64, input_shape=(input_size,), activation='linear'))
174     model.add(Dense(32, activation='linear'))
175     model.add(Dense(16, activation='linear'))
176     model.add(Dense(1, activation='linear'))
177
178     model.compile(optimizer='adam', loss='mean_squared_error')
179
180     input_shape = [X_train.shape[1]] # Assuming X_train is your input
    data
181     hypermodel = MyHyperModel(input_shape=input_shape)
182
183
184     tuner = RandomSearch(
185         hypermodel,
186         objective='val_loss',
187         max_trials=5,
188         executions_per_trial=3,
189         directory='my_dir',
190         project_name='hparam_tuning',
191     )
192
193     tuner.search(X_train, y_train, epochs=10, validation_data=(X_test,
    y_test))
194     best_model = tuner.get_best_models(num_models=1)[0]
195     best_model.save(model_name)
196
197     return {"message": "Model trained and saved successfully"}
198
199
200 @app.post("/predict")
201 async def predict_diabetes(
202     input_data: PredictionInput, user: User = Depends(get_current_user)
203 ):
204
205     try:
206         model = load_model(model_name)

```

```

207     except FileNotFoundError:
208         raise HTTPException(
209             status_code=500, detail="Model not found. Please train the
                model first."
210         )
211     i_data = input_data.dict()['input']
212     input_df = pd.DataFrame([i_data])
213     prediction = model.predict(input_df)
214     prediction_native_type = int(prediction[0]) # or float, as
        appropriate
215     if input_data.patient_id is not None:
216         db.add_historical_data_to_patient(
217             patient_id=input_data.patient_id,
218             output=DiabetesHistoricalOutput(
219                 pregnancies=input_data.input.pregnancies,
220                 glucose=input_data.input.glucose,
221                 blood_pressure=input_data.input.blood_pressure,
222                 skin_thickness=input_data.input.skin_thickness,
223                 insulin=input_data.input.insulin,
224                 bmi=input_data.input.bmi,
225                 diabetes_pedigree_function=input_data.input.
                    diabetes_pedigree_function,
226                 age=input_data.input.age,
227                 prediction=bool(prediction_native_type),
228                 created_at=datetime.datetime.today().timestamp(),
229             ),
230         )
231
232     return {"prediction": prediction_native_type}
233
234
235
236
237 @app.get("/data", response_class=HTMLResponse)
238 async def get_data_from_database(user: User = Depends(get_current_user))
    :
239     return db.select_all_diabetes_data().to_html(notebook=True)
240
241
242 @app.post("/signup", summary="Create new user")
243 async def create_user(data: UserRegister):
244     return db.create_doctor(data)
245
246
247 @app.post(
248     "/login",
249     summary="Create access and refresh tokens for user",
250     response_model=TokenSchema,
251 )
252 async def login(form_data: OAuth2PasswordRequestForm = Depends()):
253     user = db.get_doctor(form_data.username)
254     print(form_data.username)
255     if user is None:
256         raise HTTPException(status_code=400, detail="Incorrect email or
            password")

```

```

257
258     hashed_pass = user.hashed_password
259     print(hashed_pass)
260     if not verify_password(form_data.password, hashed_pass):
261         raise HTTPException(
262             status_code=400,
263             detail="Incorrect email or password",
264         )
265
266     return {
267         "access_token": create_access_token(user.email),
268         "refresh_token": create_refresh_token(user.email),
269     }
270
271
272 @app.get("/me", summary="Get details of currently logged in user",
273         response_model=User)
274 async def get_me(user: User = Depends(get_current_user)):
275     return user
276
277 @app.post("/patient", summary="Adds new patient to database",
278         response_model=Patient)
279 async def create_patient(patient_data: Patient, user: User = Depends(
280     get_current_user)):
281     return db.create_patient(patient_data)
282
283 @app.get(
284     "/patient", summary="Get list of patients with data", response_model
285     =List[Patient]
286 )
287 async def get_users(user: User = Depends(get_current_user)):
288     return db.get_patients()
289
290 //db_manager.py
291
292 import datetime
293 from typing import Optional, BinaryIO
294
295 import pandas as pd
296 from fastapi import HTTPException
297 from fastapi.responses import Response
298 from pysqlitecipher.sqlitewrapper import SqliteCipher
299 import os
300 import logging
301
302 import password_utils
303 from db_models import UserOut, UserRegister, Patient,
304     DiabetesHistoricalOutput
305
306 class SingletonMeta(type):
307     _instances = {}

```

```

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

```

```

362         True,
363     )
364
365     self.db.insertIntoTable(
366         "doctors",
367         [
368             "Senior",
369             "Registrar",
370             "senior_registrar@hospital.com",
371             "+48-111-222-333",
372             password_utils.get_hashed_password(os.getenv("
373                 DEV_PASSWORD"))),
374         ],
375         True,
376     )
377
378     self.db.createTable(
379         "patients",
380         [
381             ["PESEL", "TEXT"],
382             ["first_name", "TEXT"],
383             ["last_name", "TEXT"],
384             ["email", "TEXT"],
385             ["phone_number", "TEXT"],
386             ["historical_data", "BLOB"],
387         ],
388         True,
389         True,
390     )
391
392     def select_all_diabetes_data(self) -> pd.DataFrame:
393         columns, data = self.db.getDataFromTable("diabetes", True, True)
394         return pd.DataFrame(data=data, columns=columns)
395
396     def insert_diabetes_data(self, csv_file: BinaryIO) -> None:
397         df = pd.read_csv(csv_file)
398         df.columns = [
399             "pregnancies",
400             "glucose",
401             "blood_pressure",
402             "skin_thickness",
403             "insulin",
404             "bmi",
405             "diabetes_pedigree_function",
406             "age",
407             "outcome",
408         ]
409
410         for i in df.index:
411             self.db.insertIntoTable(
412                 "diabetes", [df[col_name][i] for col_name in df.columns
413                     ], True
414             )
415
416     def dump_diabetes_data_to_csv(self) -> None:

```

```

415         pass
416
417     def get_doctor(self, email: str) -> Optional[UserOut]:
418         columns, data = self.db.getDataFromTable("doctors", True)
419         for d in data:
420             if d[3] == email:
421                 return UserOut(
422                     first_name=d[1],
423                     last_name=d[2],
424                     email=d[3],
425                     phone_number=d[4],
426                     hashed_password=d[5],
427                 )
428         return None
429
430     def create_doctor(self, user: UserRegister):
431         columns, data = self.db.getDataFromTable("doctors", True)
432         for d in data:
433             if d[3] == user.email or d[4] == user.phone_number:
434                 raise HTTPException(
435                     status_code=404,
436                     detail=f"Doctor with email {user.email} or phone
437                             number {user.phone_number} already exists.",
438                 )
439         self.db.insertIntoTable(
440             "doctors",
441             [
442                 user.first_name,
443                 user.last_name,
444                 user.email,
445                 user.phone_number,
446                 password_utils.get_hashed_password(user.password),
447             ],
448             True,
449         )
450         return Response(
451             f"Successfully created a doctor {user.first_name} {user.
452                 last_name}"
453         )
454
455     def create_patient(self, patient: Patient):
456         columns, data = self.db.getDataFromTable("patients", True)
457         for d in data:
458             if (
459                 d[1] == patient.PESEL
460                 or d[4] == patient.email
461                 or d[5] == patient.phone_number
462             ):
463                 raise HTTPException(
464                     status_code=404,
465                     detail=f"Patient with PESEL {patient.PESEL} or email
466                             {patient.email} or phone number {patient.
467                                 phone_number} already exists.",
468                 )
469         self.db.insertIntoTable(

```

```

466         "patients",
467         [
468             patient.PESEL,
469             patient.first_name,
470             patient.last_name,
471             patient.email,
472             patient.phone_number,
473             "".encode(),
474         ],
475         True,
476     )
477
478     return patient
479
480 def add_historical_data_to_patient(
481     self, patient_id: int, output: DiabetesHistoricalOutput
482 ):
483     columns, data = self.db.getDataFromTable("patients", True)
484     exists = False
485     patient_data = None
486     print("hm")
487     for d in data:
488         if d[0] == patient_id:
489             exists = True
490             patient_data = d[6]
491             break
492     if not exists:
493         raise HTTPException(
494             status_code=404,
495             detail=f"Patient with ID {patient_id} does not exists.",
496         )
497     print("got patient")
498     patient_data = patient_data.decode()
499     print("patient current data:", patient_data)
500     data_str = (
501         f"{output.pregnancies},{output.glucose},{output.
502             blood_pressure},{output.skin_thickness},"
503         f"{output.insulin},{output.bmi},{output.
504             diabetes_pedigree_function},{output.age},"
505         f"{output.prediction},{output.created_at.timestamp()});"
506     )
507
508     patient_data += data_str
509
510     print("patient current data:", patient_data)
511     self.db.updateInTable(
512         "patients", patient_id, "historical_data", patient_data.
513             encode(), True
514     )
515
516 def get_patients(self):
517     columns, data = self.db.getDataFromTable("patients", True)
518     patients = []
519     for d in data:
520         historical_data = []

```



```

518
519         for data_point in d[6].decode().split(";")[:-1]:
520             data_point = data_point.split(",")
521             historical_data.append(
522                 DiabetesHistoricalOutput(
523                     pregnancies=data_point[0],
524                     glucose=data_point[1],
525                     blood_pressure=data_point[2],
526                     skin_thickness=data_point[3],
527                     insulin=data_point[4],
528                     bmi=data_point[5],
529                     diabetes_pedigree_function=data_point[6],
530                     age=data_point[7],
531                     prediction=data_point[8],
532                     created_at=data_point[9],
533                 )
534             )
535
536         patients.append(
537             Patient(
538                 id=d[0],
539                 PESEL=d[1],
540                 first_name=d[2],
541                 last_name=d[3],
542                 email=d[4],
543                 phone_number=d[5],
544                 historical_data=historical_data,
545             )
546         )
547     return patients
548 //db_models.py
549
550 from datetime import datetime
551 from typing import List, Optional
552
553 from pydantic import BaseModel
554
555
556 class DiabetesPredictionInput(BaseModel):
557     pregnancies: int
558     glucose: float
559     blood_pressure: float
560     skin_thickness: float
561     insulin: float
562     bmi: float
563     diabetes_pedigree_function: float
564     age: int
565
566
567 class DiabetesHistoricalOutput(DiabetesPredictionInput):
568     prediction: bool
569     created_at: datetime
570
571
572 class Patient(BaseModel):

```

```

573     id: Optional[int]
574     PESEL: str
575     first_name: str
576     last_name: str
577     email: str
578     phone_number: str
579     historical_data: Optional[List[DiabetesHistoricalOutput]]
580
581
582 class PredictionInput(BaseModel):
583     patient_id: Optional[int]
584     input: DiabetesPredictionInput
585
586
587 class User(BaseModel):
588     first_name: str
589     last_name: str
590     email: str
591     phone_number: str
592
593
594 class UserOut(User):
595     hashed_password: str
596
597
598 class UserRegister(User):
599     password: str
600
601
602 class TokenSchema(BaseModel):
603     access_token: str
604     refresh_token: str
605
606
607 class TokenPayload(BaseModel):
608     sub: str = None
609     exp: int = None
610
611 jwt_utils.py
612
613 from datetime import datetime, timedelta
614 from typing import Union, Any, Optional
615
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
624
625 from db_manager import DatabaseManager
626 from db_models import TokenPayload, UserOut, User
627

```

```

628 load_dotenv()
629
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()
638
639 reuseable_oauth = OAuth2PasswordBearer(tokenUrl="/login", scheme_name="
    JWT")
640
641
642 def create_access_token(subject: Union[str, Any]) -> str:
643     expires_delta = datetime.utcnow() + timedelta(minutes=
        ACCESS_TOKEN_EXPIRE_MINUTES)
644
645     to_encode = {"exp": expires_delta, "sub": str(subject)}
646     encoded_jwt = jwt.encode(to_encode, JWT_SECRET_KEY, ALGORITHM)
647     return encoded_jwt
648
649
650 def create_refresh_token(subject: Union[str, Any]) -> str:
651     expires_delta = datetime.utcnow() + timedelta(minutes=
        REFRESH_TOKEN_EXPIRE_MINUTES)
652
653     to_encode = {"exp": expires_delta, "sub": str(subject)}
654     encoded_jwt = jwt.encode(to_encode, JWT_REFRESH_SECRET_KEY,
        ALGORITHM)
655     return encoded_jwt
656
657
658 async def get_current_user(token: str = Depends(reuseable_oauth)) ->
    User:
659     try:
660         payload = jwt.decode(token, JWT_SECRET_KEY, algorithms=[
            ALGORITHM])
661         token_data = TokenPayload(**payload)
662
663         if datetime.fromtimestamp(token_data.exp) < datetime.now():
664             raise HTTPException(
665                 status_code=401,
666                 detail="Token expired",
667                 headers={"WWW-Authenticate": "Bearer"},
668             )
669     except (jwt.JWTError, ValidationError):
670         raise HTTPException(
671             status_code=403,
672             detail="Could not validate credentials",
673             headers={"WWW-Authenticate": "Bearer"},
674         )
675

```

```

676     user: Optional[UserOut] = db.get_doctor(token_data.sub)
677
678     if user is None:
679         raise HTTPException(
680             status_code=404,
681             detail="Could not find user",
682         )
683
684     return user
685
686 //password_utils.py
687
688 from passlib.context import CryptContext
689
690 password_context = CryptContext(schemes=["bcrypt"], deprecated="auto")
691
692
693 def get_hashed_password(password: str) -> str:
694     return password_context.hash(password)
695
696
697 def verify_password(password: str, hashed_pass: str) -> bool:
698     return password_context.verify(password, hashed_pass)
699
700 //requierments.txt
701
702 absl-py==2.0.0
703 annotated-types==0.6.0
704 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 cffi==1.16.0
710 charset-normalizer==3.3.2
711 click==8.1.7
712 cryptography==41.0.7
713 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 ini-config==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 markdown-it-py==3.0.0
735 MarkupSafe==2.1.3
736 mdurl==0.1.2
737 ml-dtypes==0.2.0
738 namex==0.0.7
739 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
763 rich==13.7.0
764 rsa==4.9
765 scikit-learn==1.3.2
766 scipy==1.11.4
767 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
785 python-dotenv==1.0.0

```

```

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

```

```

841         font-size: 1rem;
842         transition: 150ms border-color linear;
843     }
844
845
846     .contact-form input:focus,
847     .contact-form input:active {
848         border-color: var(--color-gray-medium);
849     }
850
851     button.glob{
852
853         display: block;
854         margin: 0.5rem 1rem 0;
855         padding: 0 1rem 0.125rem;
856         border-radius: 10px;
857         background-color: var(--color-darkest);
858         border: 0;
859         color: var(--color-lightest);
860         font-family: lato, sans-serif;
861         font-size: 100%;
862         letter-spacing: 0.05em;
863         line-height: 1.5;
864         text-transform: uppercase;
865         transition: 150ms all linear;
866     }
867
868     .contact-form button:hover,
869     .contact-form button:active,
870     .contact-form button:focus {
871         background: var(--color-darkest);
872         cursor: pointer;
873     }
874
875     .result {
876         margin: 10px;
877     }
878
879     .space {
880         margin: 10px;
881         height: 5px;
882         background: gray;
883         border-radius: 10px;
884     }
885
886
887     a {
888         text-decoration: none;
889         color: white;
890     }
891     header {
892         text-align: center;
893         position: fixed;
894         height: 80px;
895         width: 100%;

```

```

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

```



```

949 }
950
951 html.dark {
952   background-color: #0d0950;
953 }
954 }
955
956 //checkToken.js
957
958 let tokenToSend = `${localStorage.getItem("SavedToken")}`;
959
960 fetch('http://127.0.0.1:8000/me', {
961   credentials: "same-origin",
962   method: 'GET',
963   headers: {
964     'accept': 'application/json',
965     'Authorization': tokenToSend
966   },
967
968 }).then(response =>
969   {
970     if (response.status == 401) {
971       response.json().then(data => {
972         console.log(tokenToSend);
973         window.location.href = "/login-page";
974       });
975     }
976   });
977
978 //train.astro
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}>
987   <label >Train</label>
988   <div>
989     <form id="trainForm">
990       <section class="contact-form">
991         <label for="csvToTrain">Select a file:</label>
992         <input type="file" id="csvToTrain" name="myfile" accept="text/csv"/>
993         <button class="glob">Train</button>
994       </section>
995     </form>
996   </div>
997 </BaseLayout>
998
999
1000 <script>
1001   function sendDataToTrain(event) {
1002     event.preventDefault();
1003

```

```

1004         let tokenToSend = `${localStorage.getItem("SavedToken")}`;
1005
1006         let fileInput = (document.getElementById("csvToTrain") as
            HTMLInputElement);
1007         let file = fileInput.files[0];
1008
1009         let formData = new FormData();
1010         formData.append('file', file, file.name);
1011
1012         fetch('http://127.0.0.1:8000/train', {
1013             credentials: "same-origin",
1014             method: 'POST',
1015             headers: {
1016                 'accept': 'application/json',
1017                 'Authorization': tokenToSend
1018             },
1019             body: formData
1020         }).then(response =>
1021         {
1022             if (response.status == 200) {
1023
1024             }
1025         });
1026     });
1027 }
1028
1029     var button = document.querySelector(".glob");
1030     button.addEventListener('click', sendDataToTrain);
1031 </script>
1032
1033
1034 <style>
1035     :root {
1036         --color-lightest: #f9fdfe;
1037         --color-gray-light: #cdcfcf;
1038         --color-gray-medium: #686a69;
1039         --color-gray-dark: #414643;
1040         --color-darkest: #2a2f2c;
1041     }
1042
1043     div {
1044         text-align: center;
1045     }
1046
1047     label {
1048         text-align: center;
1049     }
1050
1051
1052     button{
1053         display: inline-block;
1054     }
1055 </style>
1056
1057 //signup-page.astro

```

```

1058
1059 ---
1060 const pageTitle = "Sign Up";
1061 import '../styles/global.css';
1062 ---
1063 <html lang="en">
1064   <head>
1065     <meta charset="utf-8" />
1066     <link rel="icon" type="image/svg+xml" href="/favicon.svg" />
1067     <meta name="viewport" content="width=device-width" />
1068     <meta name="generator" content={Astro.generator} />
1069     <title>{pageTitle}</title>
1070   </head>
1071
1072   <body>
1073     <div class="UploadFile">
1074
1075       <h1 style="color:black;">Sign Up</h1>
1076       <section class="contact-form">
1077         <form method="post">
1078           <div class="txt_field">
1079             <input class="inptClass2" id="first_name" name="first_name"
1080               type="text" required>
1081             <span></span>
1082             <label>Name</label>
1083           </div>
1084           <div class="txt_field">
1085             <input class="inptClass3" id="last_name" name="last_name"
1086               type="text" required>
1087             <span></span>
1088             <label>Surname</label>
1089           </div>
1090           <div class="txt_field">
1091             <input id="email" name="email" type="text" required>
1092             <span></span>
1093             <label>Email</label>
1094           </div>
1095           <div class="txt_field">
1096             <input class="inptClass5" id="phone_number" name="
1097               phone_number" type="number" type="number" min="0" step="
1098               1" max="999999999" required>
1099             <span></span>
1100             <label>Phone Numer</label>
1101           </div>
1102           <div class="txt_field">
1103             <input id="password" name="password" type="text" required>
1104             <span></span>
1105             <label>Password</label>
1106           </div>
1107           <button class="glob" type="submit">Sign Up</button>
1108           <div class="signup_link">
1109             <a href="/login-page">Back to login page</a>
1110           </div>
1111         </form>
1112       </section>

```

```

1109     </div>
1110
1111     <script>
1112         function handleFormSubmit(event) {
1113             event.preventDefault();
1114
1115             const data = new FormData(event.target);
1116             const formJSON = {};
1117
1118             data.forEach((value, key) => {
1119                 formJSON[key] = /^\d+$/.test(value) ? String(value) : value;
1120             });
1121             console.log(JSON.stringify(formJSON, null, 2));
1122
1123             fetch('http://127.0.0.1:8000/signup', {
1124                 credentials: "same-origin",
1125                 method: 'POST',
1126                 headers: {
1127                     'accept': 'application/json',
1128                     'Content-Type': 'application/json'
1129                 },
1130                 body: JSON.stringify(formJSON, null, 2)
1131             })
1132                 .then(response => response.json())
1133                 .then(response => console.log(JSON.stringify(response)));
1134         }
1135         document.querySelector(".inptClass2").addEventListener("keypress",
1136             function (evt) {
1137                 var charCode1 = (evt.which) ? evt.which : evt.keyCode;
1138                 if ((charCode1 >= 65 && charCode1 <= 90) || (charCode1 >= 97 &&
1139                     charCode1 <= 122)) {
1140                     // Allow the key press
1141                 } else {
1142                     // Prevent the key press if it doesn't meet the criteria
1143                     evt.preventDefault();
1144                 }
1145             });
1146         document.querySelector(".inptClass3").addEventListener("keypress",
1147             function (evt) {
1148                 var charCode2 = (evt.which) ? evt.which : evt.keyCode;
1149                 if ((charCode2 >= 65 && charCode2 <= 90) || (charCode2 >= 97 &&
1150                     charCode2 <= 122)) {
1151                     // Allow the key press
1152                 } else {
1153                     // Prevent the key press if it doesn't meet the criteria
1154                     evt.preventDefault();
1155                 }
1156             });
1157         document.querySelector(".inptClass5").addEventListener("keypress",
1158             function (evt) {
1159                 if (evt.which != 8 && evt.which != 0 && evt.which < 48 || evt.which
1160                     > 57)
1161                 {
1162                     evt.preventDefault();
1163                 }
1164             });
1165     </script>

```

```

1158     }
1159 });
1160     const form = document.querySelector('.contact-form');
1161     form.addEventListener('submit', handleFormSubmit);
1162 </script>
1163
1164
1165
1166 </body>
1167 </html>
1168
1169 <style>
1170
1171     .center{
1172     position: absolute;
1173     top: 50%;
1174     left: 50%;
1175     transform: translate(-50%, -50%);
1176     width: 400px;
1177     background: white;
1178     border-radius: 10px;
1179     box-shadow: 10px 10px 15px rgba(0,0,0,0.05);
1180 }
1181 .center h1{
1182     text-align: center;
1183     padding: 20px 0;
1184     border-bottom: 1px solid silver;
1185
1186 }
1187 .center form{
1188     padding: 0 40px;
1189     box-sizing: border-box;
1190 }
1191 form .txt_field{
1192     position: relative;
1193     border-bottom: 2px solid #adadad;
1194     margin: 30px 0;
1195 }
1196 .txt_field input{
1197     width: 100%;
1198     padding: 0 5px;
1199     height: 40px;
1200     font-size: 16px;
1201     border: none;
1202     background: none;
1203     outline: none;
1204 }
1205 .txt_field label{
1206     position: absolute;
1207     top: 50%;
1208     left: 5px;
1209     color: #adadad;
1210     transform: translateY(-50%);
1211     font-size: 16px;
1212     pointer-events: none;

```

```

1213     transition: .5s;
1214 }
1215 .txt_field span::before{
1216     content: '';
1217     position: absolute;
1218     top: 40px;
1219     left: 0;
1220     width: 0%;
1221     height: 2px;
1222     background: #2691d9;
1223     transition: .5s;
1224 }
1225 .txt_field input:focus ~ label,
1226 .txt_field input:valid ~ label{
1227     top: -5px;
1228     color: #2691d9;
1229 }
1230 .txt_field input:focus ~ span::before,
1231 .txt_field input:valid ~ span::before{
1232     width: 100%;
1233 }
1234 .pass{
1235     margin: -5px 0 20px 5px;
1236     color: #a6a6a6;
1237     cursor: pointer;
1238 }
1239 .pass:hover{
1240     text-decoration: underline;
1241 }
1242 input[type="submit"]{
1243     width: 100%;
1244     height: 50px;
1245     border: 1px solid;
1246     background: #2691d9;
1247     border-radius: 25px;
1248     font-size: 18px;
1249     color: #e9f4fb;
1250     font-weight: 700;
1251     cursor: pointer;
1252     outline: none;
1253 }
1254 input[type="submit"]:hover{
1255     border-color: #2691d9;
1256     transition: .5s;
1257 }
1258 .signup_link{
1259     float: right;
1260     margin: 10px 20px;
1261     text-align: center;
1262     font-size: 16px;
1263     color: #666666;
1264 }
1265 .signup_link a{
1266     color: #2691d9;
1267     text-decoration: none;

```

```

1268 }
1269 .signup_link a:hover{
1270     text-decoration: underline;
1271 }
1272 </style>
1273
1274 //patients.astro
1275
1276 ---
1277 import BaseLayout from '../layouts/BaseLayout.astro';
1278 const pageTitle = "Patients";
1279 ---
1280
1281 <script src="../scripts/checkToken.js"/>
1282 <script>
1283
1284
1285 </script>
1286
1287
1288 <BaseLayout pageTitle={pageTitle}>
1289     <label>Patients historical data</label>
1290     <section class="contact-form" >
1291
1292         <div class="input-group">
1293             <label for="pregnancies">List of patients</label>
1294             <select class="input-group" id="selectedValue" size="4">
1295
1296             </select>
1297         </div>
1298
1299         <div class="input-group">
1300             <label for="pregnancies">Historical data</label>
1301             <div id="histData" class="input-group">
1302
1303             </div>
1304         </div>
1305
1306     </section>
1307     <script>
1308
1309     let dataGlob;
1310     let tokenToSend = `${localStorage.getItem("SavedToken")}`;
1311
1312     fetch('http://127.0.0.1:8000/patient', {
1313         credentials: "same-origin",
1314         method: 'GET',
1315         headers: {
1316             'accept': 'application/json',
1317             'Authorization': tokenToSend
1318         },
1319
1320     }).then(response =>
1321     {
1322

```

```

1323     response.json().then(data => {
1324         const select = document.querySelector('select');
1325         console.log(data);
1326         dataGlob = data;
1327         console.log(dataGlob);
1328         for (var i = 0; i < data.length; i++){
1329             let newOption = new Option(`${data[i]["first_name"]} ${
                data[i]["last_name"]} ${data[i]["PESEL"]}`, `${data[i]
                }["id"]`);
1330
1331             select.add(newOption, undefined)}
1332     });
1333 });
1334
1335     function valueChanges() {
1336         var e = document.getElementById("selectedValue");
1337         var value = parseInt(e.value);
1338         const historicData = document.getElementById("histData");
1339         historicData.innerHTML = '';
1340         var iDiv;
1341         var label;
1342         var dateAndTime;
1343         var pregn;
1344         var gluc;
1345         var blood;
1346         var skin;
1347         var insulin;
1348         var bmi;
1349         var dpf;
1350         var age;
1351         var pred;
1352         var hrTag;
1353         var dateAndTimeArray = {};
1354         for (var i = 0; i < dataGlob[value]["
            historical_data"].length; i++){
1355             pregn = document.createElement('div');
1356             gluc = document.createElement('div');
1357             blood = document.createElement('div');
1358             skin = document.createElement('div');
1359             insulin = document.createElement('div');
1360             bmi = document.createElement('div');
1361             dpf = document.createElement('div');
1362             age = document.createElement('div');
1363             pred = document.createElement('div');
1364             label = document.createElement('label');
1365             hrTag = document.createElement('hr');
1366             console.log(dataGlob[value]["historical_data"
                ][i]);
1367             dateAndTime = dataGlob[value]["
                historical_data"][i]["created_at"];
1368             dateAndTimeArray = dateAndTime.split("T");
1369             console.log(dateAndTimeArray);
1370             historicData.appendChild(label).innerText = `
                Date and Time: ${dateAndTimeArray[0]} ${
                dateAndTimeArray[1].substring(0,8)}`;
            historicData.appendChild(pregn).innerText = `

```



```

1371         Number of pregnancies: ${dataGlob[value]["
        historical_data"][i]["pregnancies"]}'';
        historicData.appendChild(gluc).innerText = '
1372         Glucose: ${dataGlob[value]["
        historical_data"][i]["glucose"]}'';
        historicData.appendChild(blood).innerText = '
1373         Blood pressure: ${dataGlob[value]["
        historical_data"][i]["blood_pressure"]}'';
        historicData.appendChild(skin).innerText = '
1374         Skin thickness: ${dataGlob[value]["
        historical_data"][i]["skin_thickness"]}'';
        historicData.appendChild(insulin).innerText =
        'Insulin: ${dataGlob[value]["
1375         historical_data"][i]["insulin"]}'';
        historicData.appendChild(bmi).innerText = '
        BMI: ${dataGlob[value]["historical_data"][
        i]["bmi"]}'';
1376         historicData.appendChild(dpf).innerText = '
        Diabetes Pedigre Function: ${dataGlob[
        value]["historical_data"][i]["
        diabetes_pedigree_function"]}'';
1377         historicData.appendChild(age).innerText = '
        Age: ${dataGlob[value]["historical_data"][
        i]["age"]}'';
1378         historicData.appendChild(pred).innerText = '
        Has Diabetes? ${dataGlob[value]["
        historical_data"][i]["prediction"]}'';
1379         historicData.appendChild(hrTag);
1380     }
1381 }
1382
1383     document.getElementById('selectedValue').addEventListener('
        change',valueChanges);
1384     valueChanges();
1385 </script>
1386 </BaseLayout>
1387
1388 <style>
1389     #histData{
1390         max-height: 800px;
1391         overflow: auto;
1392     }
1393 </style>
1394
1395 //login-page.astro
1396
1397 ---
1398 import Snack from '../components/Snack.astro';
1399 const pageTitle = "Login";
1400 import '../styles/global.css';
1401 ---
1402 <html lang="en">
1403     <head>
1404         <meta charset="utf-8" />
1405         <link rel="icon" type="image/svg+xml" href="/favicon.svg" />

```

```

1406     <meta name="viewport" content="width=device-width" />
1407     <title>{pageTitle}</title>
1408 </head>
1409
1410 <body>
1411 <div class="UploadFile">
1412
1413     <h1 style="color:black;">Login</h1>
1414     <section class="contact-form">
1415     <form method="post">
1416         <div class="txt_field">
1417             <input id="email" name="username" type="email" required>
1418             <span></span>
1419             <label>Email</label>
1420         </div>
1421         <div class="txt_field">
1422             <input id="password" name="password" type="password"
1423                 required>
1424             <span></span>
1425             <label>Password</label>
1426         </div>
1427         <button class="glob" type="submit">Login</button>
1428
1429     </form>
1430 </section>
1431
1432     <Snack id="snackbar"></Snack>
1433 <script>
1434     function handleFormSubmit(event) {
1435         event.preventDefault();
1436
1437         const data = new FormData(event.target);
1438
1439         let username_to_change = data.get("username");
1440         let final_username = username_to_change.replace('@', '%40');
1441
1442         let body_tosent = 'grant_type=&username=${final_username}&password=${
1443             {data.get("password")}&scope=&client_id=&client_secret=';
1444
1445         fetch('http://127.0.0.1:8000/login', {
1446             credentials: "same-origin",
1447             method: 'POST',
1448             headers: {
1449                 'accept': 'application/json',
1450                 'Content-Type': 'application/x-www-form-urlencoded'
1451             },
1452             body: body_tosent
1453         }).then(response =>
1454         {
1455             if (response.status == 200) {
1456                 response.json().then(data => {
1457                     localStorage.setItem("SavedToken", "Bearer " + data["

```

```

1458
1459         window.location.href = "/";
1460     }
1461
1462     else {
1463         response.json().then(data => {
1464             var snack = document.getElementById("snackbar");
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",
1470                 ""); }, 3000);
1471         });
1472     });
1473 }
1474
1475 const form = document.querySelector('.contact-form');
1476 form.addEventListener('submit', handleFormSubmit);
1477 </script>
1478
1479 <div class="signup_link">
1480     You don't have an account? <a href="/signup-page">Signup</a>
1481 </div>
1482 </div>
1483
1484 </body>
1485 </html>
1486
1487 <style>
1488     .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;
1541 }
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 }
1559
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>
1575
1576 /index.astro
1577
1578 ---
1579 import BaseLayout from '../layouts/BaseLayout.astro';
1580 const pageTitle = "Home";
1581 ---
1582
1583 <script src="../../scripts/checkToken.js"/>
1584
1585 <script>
1586     let tokenToSend = `${localStorage.getItem("SavedToken")}`;
1587
1588     fetch('http://127.0.0.1:8000/patient', {
1589         credentials: "same-origin",
1590         method: 'GET',
1591         headers: {
1592             'accept': 'application/json',
1593             'Authorization': tokenToSend
1594         },
1595
1596     }).then(response =>
1597     {
1598
1599         response.json().then(data => {
1600             const select = document.querySelector('select');
1601             let defOption = new Option("-----",null);
1602             console.log(data[1]);
1603             select.add(defOption,undefined);
1604             for (var i = 0; i < data.length; i++){
1605                 let newOption = new Option(`${data[i]["first_name"]} ${data[i]["last_name"]} ${data[i]["PESEL"]}`,`${data[i]["id"]}`);
1606
1607                 select.add(newOption,undefined)}
1608             });
1609         });
1610     </script>
1611
1612 <BaseLayout pageTitle={pageTitle}>
1613
1614     <label >Predict</label>
1615     <section class="contact-form" >
1616         <div class="input-group">
1617             <label for="pregnancies">List of patients</label>
1618             <select class="input-group" id="selectedValue" size="4">
1619

```

```

1620         </select>
1621     </div>
1622     <form>
1623
1624         <div class="input-group">
1625             <label for="pregnancies">Pregnancies</label>
1626             <input class="inptClass" id="pregnancies" name="
pregnancies" type="number" min="0" required/>
1627         </div>
1628
1629         <div class="input-group">
1630             <label for="Glucose">Glucose</label>
1631             <input class="inptClass1" id="Glucose" name="glucose
" type="number" min="0" required/>
1632         </div>
1633
1634         <div class="input-group">
1635             <label for="name">Blood Pressure</label>
1636             <input class="inptClass2" id="name" name="
blood_pressure" type="number" min="0" required/>
1637         </div>
1638
1639         <div class="input-group">
1640             <label for="SkinThicc">Skin Thickness</label>
1641             <input class="inptClass3" id="SkinThicc" name="
skin_thickness" type="number" min="0" required/>
1642         </div>
1643
1644         <div class="input-group">
1645             <label for="Insulin">Insulin</label>
1646             <input class="inptClass4" id="Insulin" name="insulin
" type="number" min="0" required/>
1647         </div>
1648
1649         <div class="input-group">
1650             <label for="BMI">BMI</label>
1651             <input class="inptClass5" id="BMI" name="bmi" type="
number" min="0" required/>
1652         </div>
1653
1654         <div class="input-group">
1655             <label for="dpf">Diabetes Pedigree Function</label>
1656             <input class="inptClass6" id="dpf" name="
diabetes_pedigree_function" type="number" min="0"
required/>
1657         </div>
1658
1659         <div class="input-group">
1660             <label for="age">Age</label>
1661             <input class="inptClass7" id="age" name="age" type="
number" min="1" step="1" required/>
1662         </div>
1663
1664         <div class="input-group">
1665             <button class="glob" type="submit">Predict</button>

```

```

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

```

```

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

```



```

1763 });
1764
1765 document.querySelector(".inptClass6").addEventListener("keypress",
    function (evt) {
1766     if (evt.which !== 8 && evt.which !== 0 && evt.which === 46 && evt.which
        < 48 || evt.which > 57)
1767     {
1768         evt.preventDefault();
1769     }
1770 });
1771
1772 document.querySelector(".inptClass7").addEventListener("keypress",
    function (evt) {
1773     if (evt.which !== 8 && evt.which !== 0 && evt.which < 48 || evt.which
        > 57)
1774     {
1775         evt.preventDefault();
1776     }
1777 });
1778
1779     const form = document.querySelector('.contact-form');
1780     form.addEventListener('submit', handleFormSubmit);
1781 </script>
1782 </BaseLayout>
1783
1784 <style>
1785     .pred{
1786         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"/>
1798
1799 <BaseLayout pageTitle={pageTitle}>
1800     <label >Add patient</label>
1801     <section class="contact-form">
1802         <form>
1803
1804             <div class="input-group">
1805                 <label for="PESEL">PESEL</label>
1806                 <input class="inptClass1" id="PESEL" name="PESEL" type="
                    number" min="0" step="1" required/>
1807             </div>
1808
1809             <div class="input-group">
1810                 <label for="first_name">Name</label>
1811                 <input class="inptClass2" id="first_name" name="
                    first_name" type="text" required/>

```

```

1812         </div>
1813
1814         <div class="input-group">
1815             <label for="last_name">Surname</label>
1816             <input class="inptClass3" id="last_name" name="last_name"
1817                 type="text" required/>
1818         </div>
1819
1820         <div class="input-group">
1821             <label for="email">Email</label>
1822             <input class="inptClass4" id="email" name="email" type="
1823                 email" required/>
1824         </div>
1825
1826         <div class="input-group">
1827             <label for="phone_number">Phone Number</label>
1828             <input class="inptClass5" id="phone_number" name="
1829                 phone_number" type="number" min="0" step="1" max="
1830                 999999999" required/>
1831         </div>
1832
1833         <button class="glob" type="submit">Add</button>
1834     </form>
1835 </section>
1836 <Snack id="snackbar"></Snack>
1837 <script>
1838     function handleFormSubmit(event) {
1839         event.preventDefault();
1840
1841         const data = new FormData(event.target);
1842         const formJSON = {};
1843         formJSON["id"] = null;
1844         data.forEach((value, key) => {
1845             formJSON[key] = /\d+$/ .test(value) ? String(value) :
1846                 value;
1847         });
1848         formJSON["historical_data"] = null;
1849         console.log(JSON.stringify(formJSON, null, 2));
1850
1851         let tokenToSend = `${localStorage.getItem("SavedToken")}`;
1852
1853         fetch('http://127.0.0.1:8000/patient', {
1854             credentials: "same-origin",
1855             method: 'POST',
1856             headers: {
1857                 'accept': 'application/json',
1858                 'Authorization': tokenToSend,
1859                 'Content-Type': 'application/json'
1860             },
1861             body: JSON.stringify(formJSON, null, 2)
1862         }).then(response =>
1863         {
1864             if (response.status == 404) {
1865                 response.json().then(data => {
1866                     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     }
1869     if (response.status == 200) {
1870         response.json().then(data => {
1871             var snack = document.getElementById("snackbar");
1872             snack.innerHTML = "";
1873             snack.innerHTML += "Patient added correctly";
1874             snack.className = "show";
1875             setTimeout(function(){ snack.className = snack.className.replace("show",
    ""); }, 3000);
1876         });
1877     }
1878     });
1879     };
1880     document.querySelector(".inptClass1").addEventListener("keypress",
    function (evt) {
1881         if (evt.which != 8 && evt.which != 0 && evt.which < 48 || evt.which
    > 57)
1882         {
1883             evt.preventDefault();
1884         }
1885     });
1886     document.querySelector(".inptClass2").addEventListener("keypress",
    function (evt) {
1887         var charCode1 = (evt.which) ? evt.which : evt.keyCode;
1888         if ((charCode1 >= 65 && charCode1 <= 90) || (charCode1 >= 97 &&
    charCode1 <= 122)) {
1889             // Allow the key press
1890         } else {
1891             // Prevent the key press if it doesn't meet the criteria
1892             evt.preventDefault();
1893         }
1894     });
1895     document.querySelector(".inptClass3").addEventListener("keypress",
    function (evt) {
1896         var charCode2 = (evt.which) ? evt.which : evt.keyCode;
1897         if ((charCode2 >= 65 && charCode2 <= 90) || (charCode2 >= 97 &&
    charCode2 <= 122)) {
1898             // Allow the key press
1899         } else {
1900             // Prevent the key press if it doesn't meet the criteria
1901             evt.preventDefault();
1902         }
1903     });
1904
1905     document.querySelector(".inptClass5").addEventListener("keypress",
    function (evt) {
1906         if (evt.which != 8 && evt.which != 0 && evt.which < 48 || evt.which
    > 57)

```

```

1907     {
1908         evt.preventDefault();
1909     }
1910 });
1911     const form = document.querySelector('.contact-form');
1912     form.addEventListener('submit', handleFormSubmit);
1913 </script>
1914
1915 </BaseLayout>
1916
1917 //BaseLayout.astro
1918
1919 ---
1920 import Header from '../components/Header.astro';
1921 import '../styles/global.css';
1922 const { pageTitle } = Astro.props;
1923 ---
1924 <html lang="en">
1925     <head>
1926         <meta charset="utf-8" />
1927         <link rel="icon" type="image/svg+xml" href="/favicon.svg" />
1928         <meta name="viewport" content="width=device-width" />
1929         <meta name="generator" content={Astro.generator} />
1930         <title>{pageTitle}</title>
1931     </head>
1932
1933     <body>
1934         <Header/>
1935         <div class="UploadFile">
1936             <slot>
1937 </div>
1938
1939     </body>
1940 </html>
1941
1942 <style>
1943 body {
1944     overflow: scroll;
1945 }
1946 </style>
1947
1948 //ThemeIcon.astro
1949
1950 ---
1951 ---
1952 <button id="themeToggle">
1953     <svg width="30px" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24
1954         24">
1955         <path class="sun" fill-rule="evenodd" d="M12 17.5a5.5 5.5 0 1 0
1956             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
1957             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
1958             .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
1959             .8.8zm16.5-8.5a.8.8 0 0 1 0 1.6h-1.8a.8.8 0 0 1-1.1.7-1.8a
1960             .8.8 0 0 1 1 0zM6.3 17.7a.8.8 0 0 1 0 1.6h-1.7 1.8a.8.8 0 1 1-1-1
1961             1.7-1.8a.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
11.8 1.7a.8.8 0 0 1-1 1l-1.8-1.7a.8.8 0 0 1 0-1z"/>
1955 <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 4.7l1.1 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"/>
1956 </svg>
1957 </button>
1958
1959 <style>
1960 #themeToggle {
1961 border: 0;
1962 background: none;
1963 cursor: pointer;
1964 }
1965 .sun { fill: white; }
1966 .moon { fill: transparent; }
1967
1968 button {
1969 float: right;
1970 margin-top: 5px;
1971 display: inline-block
1972 }
1973
1974 :global(.dark) .sun { fill: transparent; }
1975 :global(.dark) .moon { fill: white; }
1976 </style>
1977
1978 <script is:inline>
1979 const theme = (() => {
1980 if (typeof localStorage !== 'undefined' && localStorage.getItem('
theme')) {
1981 return localStorage.getItem('theme');
1982 }
1983 if (window.matchMedia('(prefers-color-scheme: dark)').matches) {
1984 return 'dark';
1985 }
1986 return 'light';
1987 })();
1988
1989 if (theme === 'light') {
1990 document.documentElement.classList.remove('dark');
1991 } else {
1992 document.documentElement.classList.add('dark');
1993 }
1994
1995 window.localStorage.setItem('theme', theme);
1996
1997 const handleToggleClick = () => {
1998 const element = document.documentElement;
1999 element.classList.toggle("dark");
2000

```

```

2001     const isDark = element.classList.contains("dark");
2002     localStorage.setItem("theme", isDark ? "dark" : "light");
2003 }
2004
2005     document.getElementById("themeToggle").addEventListener("click",
2006         handleToggleClick);
2007 </script>
2008 //Snack.astro
2009 ---
2010 ---
2011
2012 <div id="snackbar"></div>
2013
2014 <style>
2015     #snackbar {
2016         visibility: hidden;
2017         min-width: 250px;
2018         margin-left: -125px;
2019         background-color: #333;
2020         color: #fff;
2021         text-align: center;
2022         border-radius: 2px;
2023         padding: 16px;
2024         position: fixed;
2025         z-index: 1;
2026         left: 50%;
2027         bottom: 30px;
2028     }
2029
2030 #snackbar.show {
2031     visibility: visible;
2032     -webkit-animation: fadein 0.5s, fadeout 0.5s 2.5s;
2033     animation: fadein 0.5s, fadeout 0.5s 2.5s;
2034 }
2035
2036 @-webkit-keyframes fadein {
2037     from {bottom: 0; opacity: 0;}
2038     to {bottom: 30px; opacity: 1;}
2039 }
2040
2041 @keyframes fadein {
2042     from {bottom: 0; opacity: 0;}
2043     to {bottom: 30px; opacity: 1;}
2044 }
2045
2046 @-webkit-keyframes fadeout {
2047     from {bottom: 30px; opacity: 1;}
2048     to {bottom: 0; opacity: 0;}
2049 }
2050
2051 @keyframes fadeout {
2052     from {bottom: 30px; opacity: 1;}
2053     to {bottom: 0; opacity: 0;}
2054 }

```

```

2055 </style>
2056
2057 //ScriptSnack.jsx
2058 import { useState } from 'preact/hooks';
2059
2060 export default function Snack({messages}) {
2061
2062     const randomMessage = () => messages[(Math.floor(Math.random() *
        messages.length))];
2063
2064     const [greeting, setGreeting] = useState(messages[0]);
2065
2066     return (
2067         <div>
2068             <h3>{greeting}! Thank you for visiting!</h3>
2069             <button onClick={() => setGreeting(randomMessage())}>
2070                 New Greeting
2071             </button>
2072         </div>
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';
2088 ---
2089 <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font
    -awesome/4.7.0/css/font-awesome.min.css">
2090 <header>
2091     <nav>
2092
2093         <a class="log-out"><i class="fa fa-sign-out" style="font-size:36px
            ;color:white"></i></a>
2094         <div class="nav-container">
2095             <a class="link-to-pages" href="/add-patient">Add Patient</a>
2096             <a class="link-to-pages" href="/">Predict</a>
2097             <a class="link-to-pages" href="/patients">All Patients</a>
2098             <a class="link-to-pages" href="/train">Train</a>
2099         </div>
2100         <ThemeIcon />
2101     </nav>
2102
2103 </header>
2104
2105 <script>

```

```
2106     const button = document.querySelector(".log-out");
2107
2108     button.addEventListener("click", (event) => {
2109         localStorage.removeItem("SavedToken");
2110         window.location.href = "/login-page";
2111     });
2112
2113     </script>
2114
2115 <style>
2116     a.log-out{
2117         float: left;
2118     }
2119     .nav-container{
2120         display: inline-block;
2121     }
2122     a.link-to-pages{
2123         border-style: none none solid none;
2124         border-radius: 4px;
2125         margin: 0px 5px 0px 5px;
2126         display:inline-block;
2127     }
2128 </style>
```
