

A. Identyfikacja zagadnienia biznesowego (problemu)

Celem projektu jest umożliwienie lekarzowi tworzenie/odczyt notatek z badania/wizyty pacjenta. Podstawową wersją systemu jest możliwość logowania lekarza, dodawanie pacjentów, dodawanie/odczyt notatek z wizyty/badania. System realizuje tylko wcześniej wymienione podstawowe funkcje i nie wyczerpuje problemu biznesowego całkowicie.

B. Wymagania systemowe i funkcjonalne

Problem biznesowy zostanie rozwiązany w formie aplikacji internetowej.

Wymagania systemowe użytkownika:

Dowolne urządzenie z przeglądarką internetową i dostępem do internetu.

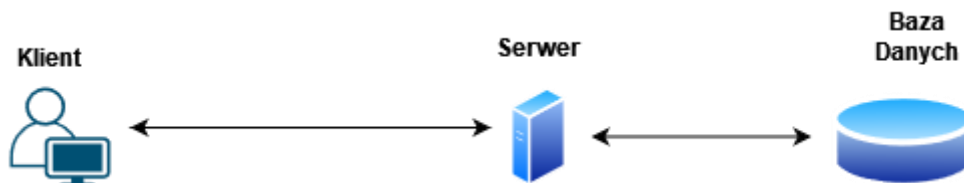
Wymagania funkcjonalne:

- Zarządzanie użytkownikami systemu (kontami lekarzy)
- System logowania i autoryzacji użytkownika(lekarza)
- Możliwość dodawania pacjentów (przez lekarzy)
- Możliwość sprawdzania notatek o pacjencie (przez lekarzy)
- Możliwość dodawania notatek o pacjencie (przez lekarzy)

Aplikacja oparta będzie na architekturze MVC w oparciu o:

- Node.js + Express.js, po stronie back-end'u
- React.js po stronie front-end'u
- Relacyjną bazę danych SQL

C. Analiza zagadnienia i jego modelowanie



Aplikacja bazuje na modelu MVC.

Po stronie klienta za widok odpowiedzialna jest biblioteka React i podstawowe komponenty

- Login.js – odpowiedzialny za logowanie użytkownika oraz pobranie tokenu do autoryzacji
- Logout.js – odpowiedzialny za wylogowanie użytkownika i usunięcie tokenu
- Patient.js – odpowiedzialny za możliwość dodania pacjenta, dodania notatki o pacjencie oraz wyświetlenie notatek o pacjencie
- UseToken.js - umożliwia ustawianie/pobieranie tokenu

Komponenty Login.js, Logout.js, Patient.js powinny móc uzyskać stosowne dane za pośrednictwem żądań http do serwera na którym działa back-end.

Podstawowe żądania http przez klienta:

- /patient/create metodą post przesyłane token i dane z formularza do utworzenia pacjenta w bazie
- /patient/createNote metodą post przesyłane token i dane z formularza do utworzenia notatki o pacjencie

- /patient/getNote metodą post przesyłane token i dane z formularza do pobrania notatek o pacjencie, oczekiwany zwrot z notatkami o pacjencie
- /login metodą post przesyłane dane z formularza do logowania, oczekiwany zwrot tokenu przy prawidłowych danych

Dane przechowywane w bazie są jednoznacznie identyfikowane przy pomocy numeru PESEL i nazwę użytkownika lekarza.

docker patient	docker notes	docker doctor
FirstName : varchar(255)	PESEL : varchar(11)	FirstName : varchar(255)
LastName : varchar(255)	Note : text	LastName : varchar(255)
PESEL : varchar(11)	Date : datetime	PESEL : varchar(11)
	DoctorUserName : varchar(255)	UserName : varchar(255)
		Password : varchar(255)

Za dodawanie kont lekarzy odpowiedzialny administrator bazy danych.

D. Implementacja

Komponent UseToken umożliwia w App.js wygenerowanie panelu logowania lub panelu użytkownika

```
import Login from './Components/Login'
import useToken from './Components/UseToken';
import Logout from './Components/Logout'
import Patient from './Components/Patient'

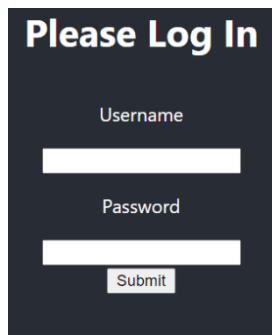
function App() {
  const { token, setToken } = useToken(); //*****

  if(!token) { //panel logowania
    return(
      <div className="App">
        <Login setToken={setToken} />
      </div>
    )...else panel użytkownika
```

```
const UseToken = () => {...
...
const [token, setToken] = useState(getToken());
const saveToken = userToken => {
  sessionStorage.setItem('token', JSON.stringify(userToken));
  setToken(userToken.token);
};

return {
  setToken: saveToken,
  token
}
```

```
const [token, setToken] = useState(getToken())(); //*****
***** getToken pobiera token z sessionStorage, jeżeli nie istnieje to zwraca undefined.
```



Please Log In

Username

Password

Submit

Komponent Login

```
const Login = ( {setToken} ) => {...
```

(Poprzez setToken pozwala na ustawienie tokena.)

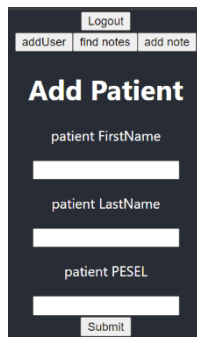
pobiera dane z formularza i zapisuje je przy użyciu

```
const [username, setUsername] = useState("");
const [password, setPassword] = useState("");
```

Po wprowadzeniu i zatwierdzeniu danych, username i password są walidowane. Jeżeli dane są poprawne następuje wywołanie funkcji

```
async function loginUser(credentials) {
  return fetch("http://localhost:81/login", {
    method: 'POST',...
```

Po stronie back-endu w routes/login.js realizowane jest zapytanie do bazy i sprawdzenie czy istnieje użytkownik z podanym loginem i hasłem. Jeżeli tak przy pomocy jsonwebtoken zostaje tworzony i zwracany token do autoryzacji, jeżeli nie zostaje zwrócony w odpowiedzi {status: 'error'}.



Logout | addUser | find notes | add note

Add Patient

patient FirstName

patient LastName

patient PESEL

Submit

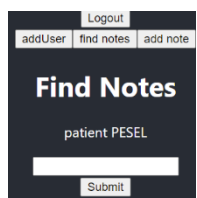
Komponent Logout

```
const logout = () =>{
  const [loggedOut, setloggedOut] = useState(false);
```

```
if(loggedOut) {
  window.location.href = '/';
}
```

po kliknięciu zmienia loggedOut na true i wraca do strony głównej.

W związku, że token zostaje usunięty z sessionStorage, ładowany jest panel logowania.



Logout | addUser | find notes | add note

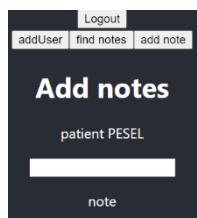
Find Notes

patient PESEL

Submit

Komponent Patient

```
const Patient = () => {
  const [FirstName, setFirstName] = useState(""); //Add
  const [LastName, setLastName] = useState(""); //Add
  const [PESEL, setPESEL] = useState(""); // Add, Find,
  const [note, setNote] = useState("");
  const [menuOpt, setMenuOpt] = useState(0);
  const [notes, setNotes] = useState("");
```



Logout | addUser | find notes | add note

Add notes

patient PESEL

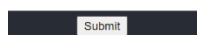
note

FirstName, LastName, PESEL używane są przy dodawaniu pacjenta

PESEL jest także użyty do identyfikacji dla jakiego pacjenta dodajemy/ pobieramy notatki

note przechowuje notatkę do wysłania do bazy

notes przechowuje notatki o użytkowniku z bazy



Submit

menuOpt używane jest do przełączania między dodawaniem pacjenta, dodawaniem notatki bądź pobrania notatek

Dodawanie pacjenta

```
async function addPatient(dataPatient) {  
  return fetch("http://localhost:81/patient/create", {  
    method: 'POST',
```

Dodawanie notatki

```
async function addPatientNote(dataPatient) {  
  return fetch("http://localhost:81/patient/createNote", {
```

Pobranie notatek

```
async function getPatientNote(dataPatient) {  
  return fetch("http://localhost:81/patient/getNote", {
```

razem z `dataPatient` zostaje wysłany token do autoryzacji.

Po stronie backendu powyższe żądania są realizowane przez `routes/patient.js` jeżeli token wygasł bądź jest nieprawidłowy zostaje zwrócona odpowiedź z `{ status: 'error', 'errorName: 'bad token' }` i następuje wylogowanie (usunięcie tokena z `sessionStorage` i powrót do strony logowania).

Połączenie z bazą jest zrealizowane w `routes/db.js`.

E. Podsumowanie

Udało się zrealizować podstawowe założenia projektu. Obecny stan jest jednak ubogą wersją systemu i należy pomyśleć o jej rozbudowaniu np. o modyfikację hasła lekarza, modyfikację i usuwanie błędnie wysłanych notatek.