

Dokumentacja Inżynierii Wymagań – System Kontroli Dostępu

Raport zawiera elementy wykonanej dokumentacji inżynierii wymagań dla projektu systemu kontroli dostępu w zakładzie przemysłowym.

1. Macierz kompetencji zespołu

Kompetencje	Paweł Kowalcze	Mateusz Gacek	Jan Ogiegło
Programowanie Python (Flask)	Posiada	Posiada	Posiada
Przetwarzanie obrazu (np. OpenCV)	Nie posiada	Nie posiada	Nie posiada
Bazy Danych <ul style="list-style-type: none">SQLSQLAlchemy	Posiada (podstawy)	Posiada (podstawy)	Posiada (podstawy)
Frontend <ul style="list-style-type: none">HTMLCSSJavaScriptReact	Posiada (zaawansowany)	Nie posiada	Nie posiada
DevOps (Docker)	Nie posiada	Posiada (podstawy)	Posiada
Znajomość UML	Posiada	Posiada	Posiada
Tworzenie wizualizacji frontendu (Figma)	Posiada	Posiada	Posiada
Testowanie Oprogramowania	Posiada (podstawy)	Posiada (podstawy)	Posiada (podstawy)
Kontrola wersji (Git)	Posiada	Posiada	Posiada

2. Pytania i odpowiedzi z fazy analitycznej (spotkanie z klientem)

W tabeli poniżej umieszczono zestaw pytań, które zostały sformułowane w celu uszczegółowienia projektu. Zanotowano odpowiedzi, które pojawiły się w trakcie dyskusji.

Pytanie	Odpowiedź	Uwagi
Jak wygląda proces dodawania nowego pracownika do systemu?	Administrator robi nowemu pracownikowi kilka zdjęć lub wideo i wprowadza jego dane do systemu. Następnie generuje dla niego unikalny kod QR.	Proces musi być obsługiwany przez dedykowany panel administracyjny.
Co w sytuacji, gdy system błędnie nie rozpozna uprawnionego pracownika?	Taka próba jest logowana jako nieudana. Procedura zakłada ręczną weryfikację przez pracownika ochrony, który może otworzyć bramkę po sprawdzeniu dokumentów.	Kluczowe jest raportowanie fałszywych negatywów w celu kalibracji systemu.
Jakie dane mają być zawarte w raportach?	Raporty powinny zawierać statystyki poprawnych i niepoprawnych wejść, identyfikować próby nadużyć (np. użycie kodu QR przez inną osobę) oraz logować czas wejścia	Raporty powinny być generowane w określonych przedziałach czasowych (dziennie).
W jaki sposób pracownik prezentuje kod QR?	Dopuszczalne jest skanowanie kodu z ekranu telefonu komórkowego lub z wydrukowanej przepustki.	Skaner musi być w stanie odczytać kody z różnych powierzchni.
Jak wygląda zarządzanie kodami QR?	Zarządzanie kodami QR w systemie obejmuje generowanie, weryfikację i unieważnianie kodów przez panel administracyjny,	Administrator musi mieć możliwość zarządzania kodami.

	powiązane z profilem pracownika i datą ważności.	
--	--	--

3. Ustalony format danych

Pracownicy	Kody QR	Próba Dostępu	Raport
id (int, unikalny)	id_kodu (int, unikalny)	timestamp (epoch)	timestamp_generowania (epoch)
imię (string)	kod_qr (int)(kod QR generowany używając id pracownika)	id_pracownika (z kodu QR)	zakres_dat (string)
nazwisko (string)	data_ważności (epoch)	wynik_weryfikacji_twarzy (float, 0-1)	statystyki (np. % poprawnych wejść)
zdjęcia (array, ścieżki do plików)	status (aktywny/nieaktywny)	zdjęcie_z_proby (ścieżka do pliku)	lista_nadużyć (array)
id_kodu (int)			

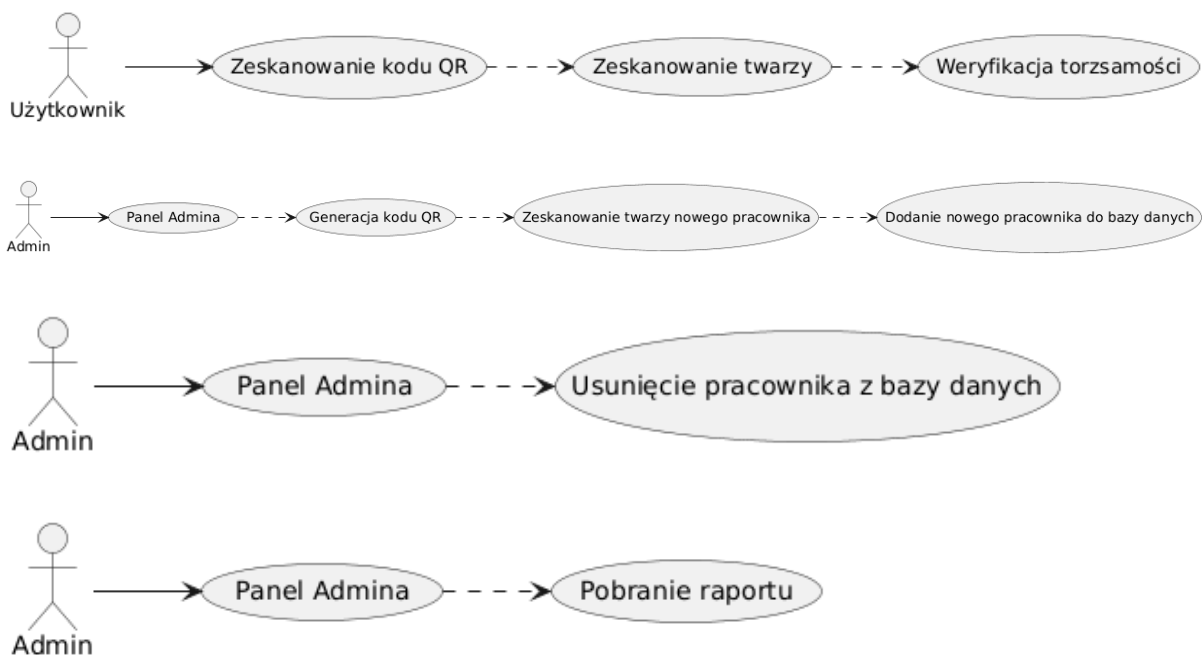
Przykładowa linia z danymi pracownika: `12,Jan,Kowalski,['/zdjecia/12_1.jpg','/zdjecia/12_2.jpg'];`

4. Modelowanie systemu

Aktorzy	Pracownik, Administrator Systemu
Opis	Celem systemu jest weryfikacja tożsamości pracowników wchodzących i wychodzących z terenu fabryki poprzez jednoczesne skanowanie kodu QR i rozpoznawanie twarzy. System ma zapobiegać nadużyciom związanym z przekazywaniem kart dostępowych i generować raporty.
Dane	Baza danych pracowników (dane osobowe, wzorce twarzy), kody QR, logi prób dostępu.
Wyzwalacz	Pracownik podchodzi do punktu kontroli i inicjuje skanowanie kodu QR.

Odpowiedź	System weryfikuje kod, analizuje obraz z kamery, porównuje twarz z wzorcem w bazie i podejmuje decyzję (dostęp przyznany/odmowa). Każda próba jest zapisywana w dzienniku zdarzeń.
Uwagi	Kluczowe jest zapewnienie odpowiedniego oświetlenia na stanowisku kontroli dostępu, aby zminimalizować błędy w rozpoznawaniu twarzy.

Diagram przypadków użycia:

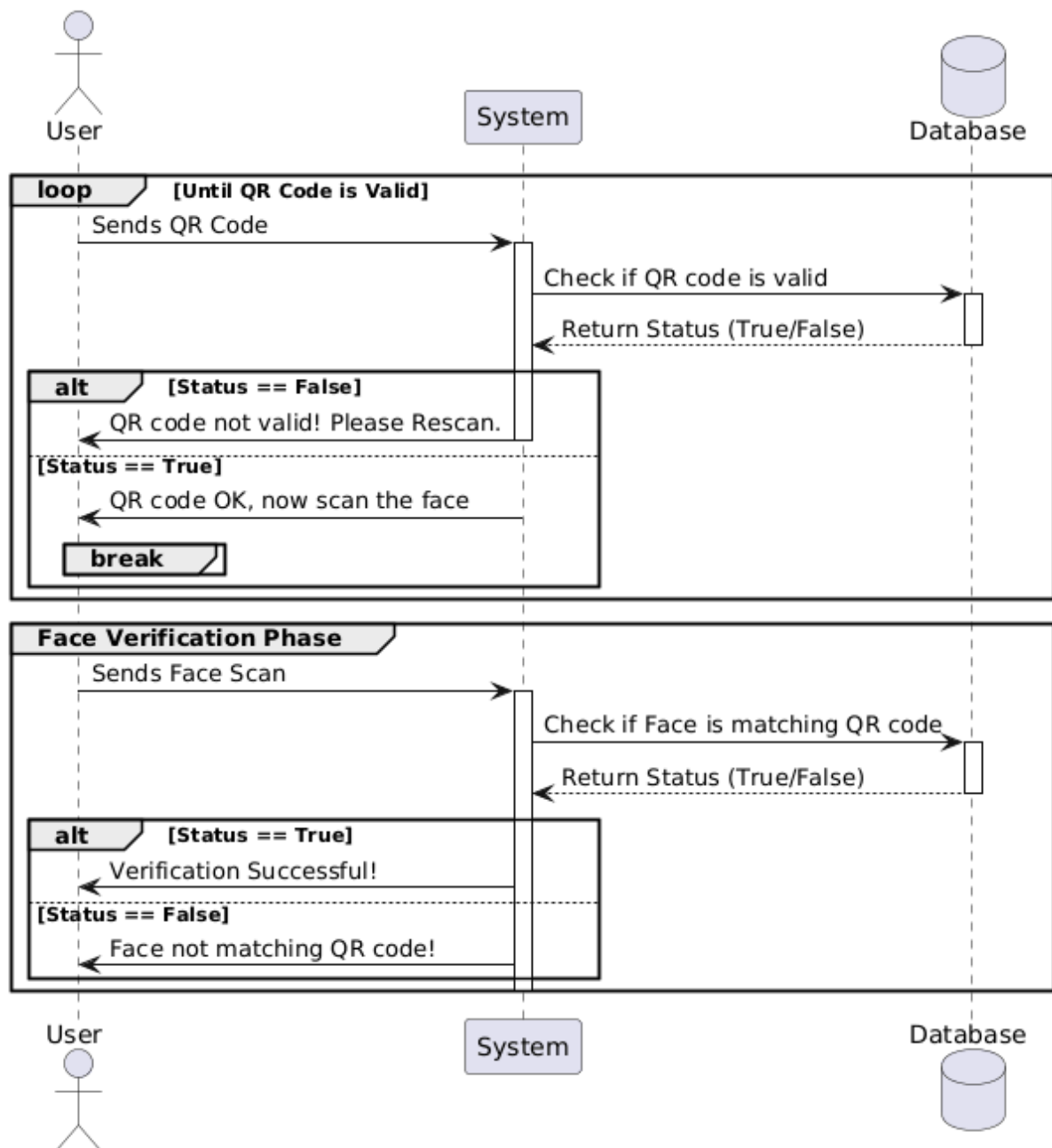


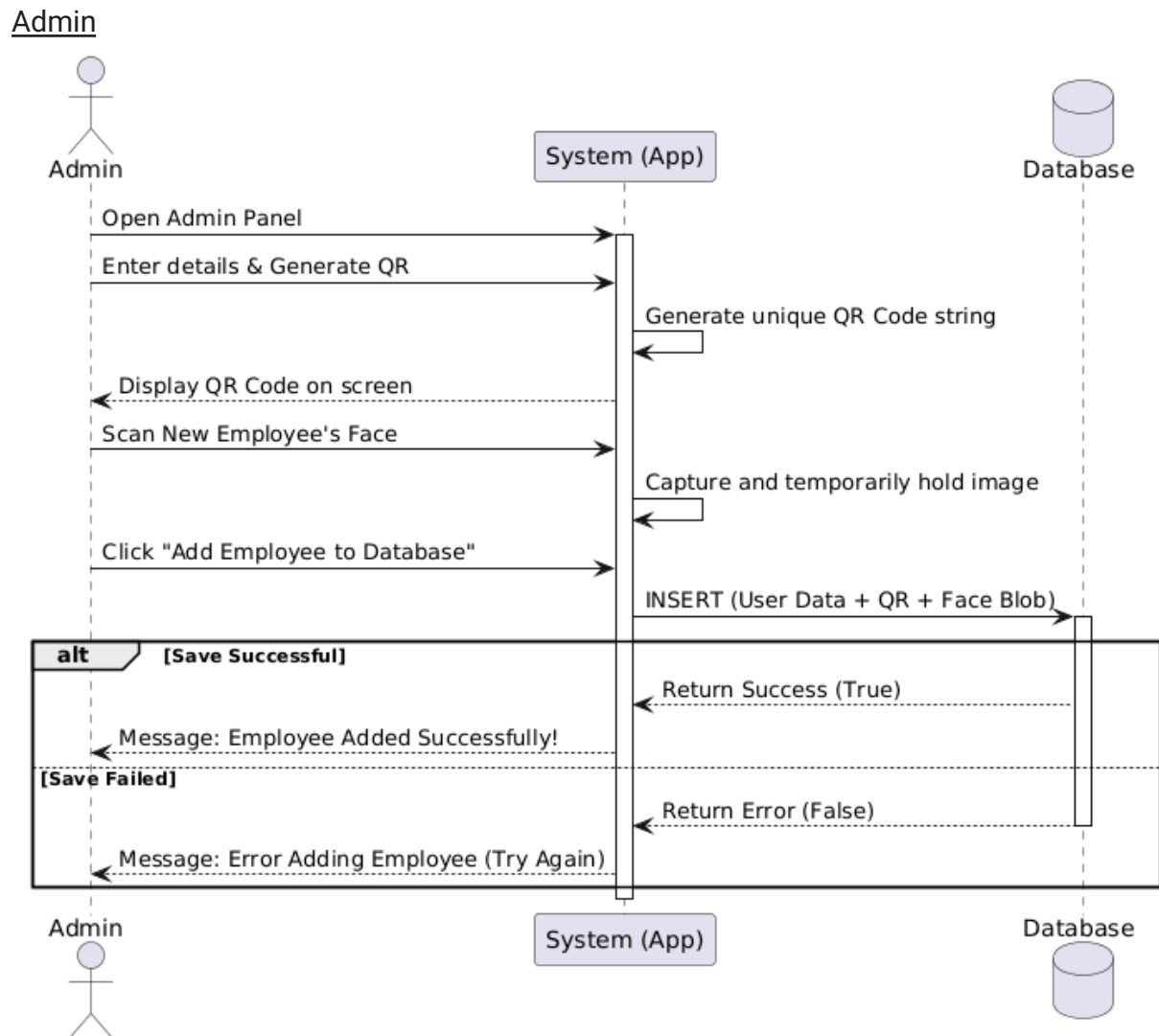
5. Diagram sekwencyjny UML

Użytkownik (User)

1. **Pracownik -> Skaner QR**: Prezentuje kod QR.
2. **Skaner QR -> System**: Przesyła odczytane dane z kodu QR.
3. **System -> Kamera**: Inicjuje przechwycenie obrazu.
4. **Kamera -> System**: Przesyła obraz (zdjęcie) twarzy.
5. **System -> Baza Danych**: Wysyła zapytanie o ważność kodu QR i pobranie wzorców twarzy dla pracownika.
6. **Baza Danych -> System**: Zwraca status kodu i wzorce twarzy.
7. **System -> System (Moduł Rozpoznawania)**: Uruchamia algorytm porównania twarzy z obrazu z wzorcami z bazy.
8. **System -> System**: Zwraca wynik porównania
9. **System -> Baza Danych**: Zapisuje wynik próby dostępu w logach.

10. **System -> Pracownik:** Wyświetla komunikat o przyznaniu lub odmowie dostępu.





6. Projekt architektury opracowywanego systemu

Start - wejście -> Przetwarzanie -> Koniec - wyjście

- **Komponenty wejściowe:**
 - Skaner kodów QR.
 - Kamera wideo/aparat.
-
- **Komponenty przetwarzające:**
 - **Moduł Weryfikacji Kodu QR:** Odpowiedzialny za dekodowanie i sprawdzanie ważności kodu w bazie danych.
 - **Moduł Rozpoznawania Twarzy:** Kluczowy komponent, który analizuje obraz, wykrywa twarz i porównuje ją z zapisanymi wzorcami.
 - **Baza Danych:** Przechowuje dane pracowników, ich zdjęcia/wzorce biometryczne, uprawnienia i historię logowań.

- **Panel Administracyjny:** Interfejs webowy dla administratorów do zarządzania systemem.
-
- **Komponenty wyjściowe:**
 - **Mechanizm sygnalizacyjny:** Wyświetlacz lub dioda informująca o statusie weryfikacji.
 - **Moduł Logowania:** Zapisuje każdą próbę dostępu do bazy danych.
 - **Moduł Raportowania:** Generuje raporty na żądanie administratora.
-

System musi być zaprojektowany w sposób modułowy. Krytycznym elementem jest algorytm rozpoznawania twarzy, który musi być zarówno szybki (aby nie tworzyć kolejek przy wejściu), jak i dokładny (aby zminimalizować liczbę fałszywych odrzuceń i akceptacji). Skuteczność tego modułu bezpośrednio wpływa na wiarygodność całego systemu i generowanych raportów.

7. Język implementacji Backendu: Python

Uzasadnienie:

Wybór języka Python jest rekomendowany ze względu na bogaty ekosystem bibliotek open-source, które znacząco przyspieszą i ułatwią rozwój projektu. Biblioteki takie jak **OpenCV** i **Dlib** oferują zaawansowane, gotowe do użycia narzędzia do przetwarzania obrazu i rozpoznawania twarzy. Frameworki webowe, takie jak **Django** lub **Flask**, umożliwią szybkie stworzenie bezpiecznego i funkcjonalnego panelu administracyjnego. Szerokie wsparcie dla różnorodnych baz danych (np. PostgreSQL, MySQL) zapewni elastyczność w warstwie przechowywania danych.

8. Implementacja interfejsu graficznego: Figma, HTML, CSS, JavaScript, React.

Uzasadnienie:

Tworzenie przykładowych widoków odbędzie się przy pomocy programu Figma. Jest to dedykowane narzędzie to tego typu zadań.

Interfejs graficzny będzie najłatwiej zaimplementować za pomocą biblioteki React. Jest ona bardzo rozbudowana i daje wiele możliwości tworzenia bardziej skomplikowanych projektów.

Funkcjonalność:

Ekran logowania / widok główny:

- przycisk do rozpoczęcia procesu weryfikacji
- ładna oprawa graficzna

Wyświetlanie obrazu przechwytywanego przez kamerę:

- Do kodu QR
 - Dopasowanie przechwytywanego kodu do odpowiedniego pola
 - Pop up / alert - Witaj! XYZ
- Do rozpoznawania twarzy - automatyczne rozpoczęcie rozpoznawania
 - Dopasowanie położenia głowy do odpowiedniego pola
 - Pop up / alert - Rozpoznano / Nie rozpoznano

Widok po skanowaniu:

- Komunikat odnośnie weryfikacji - potwierdzenie lub odrzucenie weryfikacji
- Przycisk do powrotu do widoku głównego (automatyczne przejście po paru sekundach do widoku głównego)
- Przycisk do ponownej próby weryfikacji
- Komunikat w przypadku przekroczenia liczby dozwolonych prób

Animacje i dźwięki:

- Ładowania strony
- Skanowania kodu QR
- Rozpoznawania twarzy

Panel administratora:

- okienko logowania
- dodanie pracowników
- generowanie kodów QR dla danego pracownika
- skanowanie twarzy pracowników (zdjęcia / wideo)
- usuwanie pracowników
- raporty z każdej próby wejścia - udanych lub nieudanych