UNIWERSYSTET IM. ADAMA MICKIEWICZA W POZNANIU

WYDZIAŁ MATEMATYKI I INFORMATYKI

INŻYNIERIA OPROGRAMOWANIA

DOKUMENTACJA PROJEKTU

Cat or not (Kot czy nie)

Wersja v2.1.0

CZŁONKOWIE ZESPOŁU

Jerzy Kwiatkowski - Project Manager, System Analyst

Aleksandra Malejka - Process Engineer, System Analyst

Artur Ziętkiewicz - Software Architect

Konrad Przysowa - Software Architect

Serhii Hrosh - Integrator

Jakub Sobkiewicz - System Administrator

Tomasz Torchalski - Test Manager

Daniil Karunou - User-interface Developer, Implementator

Paweł Łączkowski - Course Developer, System Tester

Yurii Borodiukh - Course Developer

Joanna Ładyko - System tester

Michał Szczygieł - Implementator

Dominik Mikołajczyk - Implementator

AUTORZY DOKUMENTACJI

Paweł Łączkowski

Yurii Borodiukh

Poznań, 2021/2022

Spis treści

1. Wstęp

Niniejsza dokumentacja przygotowana jest na potrzeby zespołowego projektu programistycznego z zajęć Inżynierii Oprogramowania prowadzonych na Uniwersytecie im. Adama Mickiewicza w Poznaniu na Wydziale Matematyki i Informatyki. Podmiotem dokumentacji jest projekt o nazwie *Cat or not? (Kot czy nie?)*.

2. Charakterystyka projektu

Zadaniem projektu jest przygotowanie aplikacji umożliwiającej wczytywanie fotografii i jej analizy w celu rozpoznawania czy na zdjęciu znajduje się kot czy też nie. Aplikacja będzie miała formę aplikacji internetowej.

2.1 Problemy stawiane przez projekt

- wymyślenie sposobu zapewnienia funkcjonalności rozpoznawania obiektu na fotografii
- zaprojektowanie przyjaznej dla użytkownika aplikacji

Do rozwiązania problemu stworzenia aplikacji wykorzystamy gotowe biblioteki oferujące funkcję webowe, dla funkcjonalności rozpoznawania biblioteki nauczania maszynowego.

2.2 Cele projektu

- implementacja modułu do wczytywania fotografii
- implementacja nauczania maszynowego do analizy fotografii
- przygotowanie i implementacja interfejsu użytkownika
- integracja modułów wczytywania, nauczania maszynowego i interfejsu użytkownika
- wdrożenie aplikacji internetowej

3. Specyfikacja wymagań funkcjonalnych

3.1 Opis wymagań funkcjonalnych

Projekt ma udostępniać aplikację internetową zawierającą interfejs użytkownika zintegrowany z modułem wczytywania fotografii w różnych formatach graficznych (np. *.jpg, .png, .bmp*) oraz modułem analizy fotografii.

3.1.1 Intefejs użytkownika

Interfejs użytkownika powinien zawierać dwa przyciski:

- przycisk ***Wczytaj*** odpowiedzialny za funkcjonalność wczytywania fotografii
- przycisk ***Analizuj*** odpowiedzialny za funkcjonalność rozpoznawania czy na zdjęciu znajduje się pożądany obiekt

Kliknięcie przycisku ***Wczytaj*** powinno wyświetlić odpowiednie okno dialogowe, umożliwiające wczytanie fotografii w obsługiwanym przez aplikację formacie.

W pobliżu przycisku ***Wczytaj*** powinna widnieć informacja o obsługiwanych typach plików graficznych przez aplikację.

Przycisk ***Analizuj*** powinien być dostępny dopiero po wczytaniu zdjęcia, tj. niemożliwe jest jego użycie bez wczytania zdjęcia.

Kliknięcie przycisku ***Analizuj*** powinno uruchomić moduł do analizy fotografii, który po wykonaniu analizy zwróci odpowiedź.

Odpowiedni komunikat, wraz z przeanalizowanym zdjęciem, powinnien wyświetlić się pod przyciskami funkcyjnymi ***Wczytaj*** i ***Analizuj***.

3.1.2 Moduł wczytywania fotografii

Moduł służący do wczytywania fotografii powinien udostępniać ***funkcję*** służącą do sprawdzenia, czy przesłany plik jest plikiem graficznym, czy jest to plik graficzny formatu obsługiwanego przez aplikację oraz czy plik ma rozmiar dopuszczalny przez aplikację, która jako parametr przyjmowałaby wczytany przez użytkownika plik, a w wyniku zwracałaby odpowiedni komunikat.

W przypadku niepomyślnego przejścia testów, powinniśmy otrzymać jedną z następujących odpowiedzi, w zależności od tego, czego dotyczy błąd:

- ***Rozmiar wczytanego pliku jest za duży!***
- ***Wczytany plik nie jest plikiem obsługiwanym przez aplikację!***

Odpowiedni komunikat powinnien wyświetlić się pod przyciskami funkcyjnymi ***Wczytaj*** i ***Analizuj***.

3.1.3 Moduł analizujący

Moduł służący do analizy fotografii w celu rozpoznania czy pożądany obiekt się na niej znajduje powinien wykorzystywać nauczanie maszynowe.

Moduł analizujący fotografię powinien uruchamiać się po kliknięciu przycisku ***Analizuj***, po ówczesnym wczytaniu fotografii.

Moduł powinien udostępniać odpowiednią ***funkcję*** służącą do analizy fotografii, która jako parametr przyjmowałaby wczytane przez użytkownika zdjęcie, a w wyniku zwracałaby odpowiedni komunikat.

W wyniku analizy powinniśmy otrzymać jedną z dwóch odpowiedzi:

- ***Kot widnieje na fotografii.***
- ***Kot nie widnieje na fotografii.***

3.2 Analiza wymagań funkcjonalnych

3.2.1 Użytkownicy systemu (aktorzy)

Wyróżnia się następujących aktorów systemu:

- ***Serwer/System/Analizator*** - system udostępniający usługę analizowania fotografii dla Użytkownika/Klienta, odpowiada za analizę fotografii oraz sprawdzenie, czy wczytano plik graficzny i czy jest on obsługiwany przez aplikację.
- ***Użytkownik/Klient*** - osoba korzystająca z usług oferowanych przez aplikacje, może wgrać fotografię do analizy oraz przesłać ją do Serwera/Systemu/Analizatora w celu analizy.

3.2.2 Procesy funkcjonalne (przypadki użycia)

Wyróżnia się następujące przypadki użycia:

- Scenariusz główny
  1. System wyświetla interfejs do wczytywania zdjęcia.
  2. Użytkownik wczytuje plik (fotografię) do analizy.
  3. Użytkownik przesyła plik (fotografię) do analizy.
  4. System sprawdza poprawność pliku pod kątem dopuszczalnego rozmiaru, czy jest plikiem graficznym i czy jest obsługiwany.

5. System analizuje fotografię i zwraca odpowiedź.
6. System wyświetla użytkownikowi odpowiedź.

- Scenariusz alternatywny - rozmiar wczytanego pliku jest za duży.

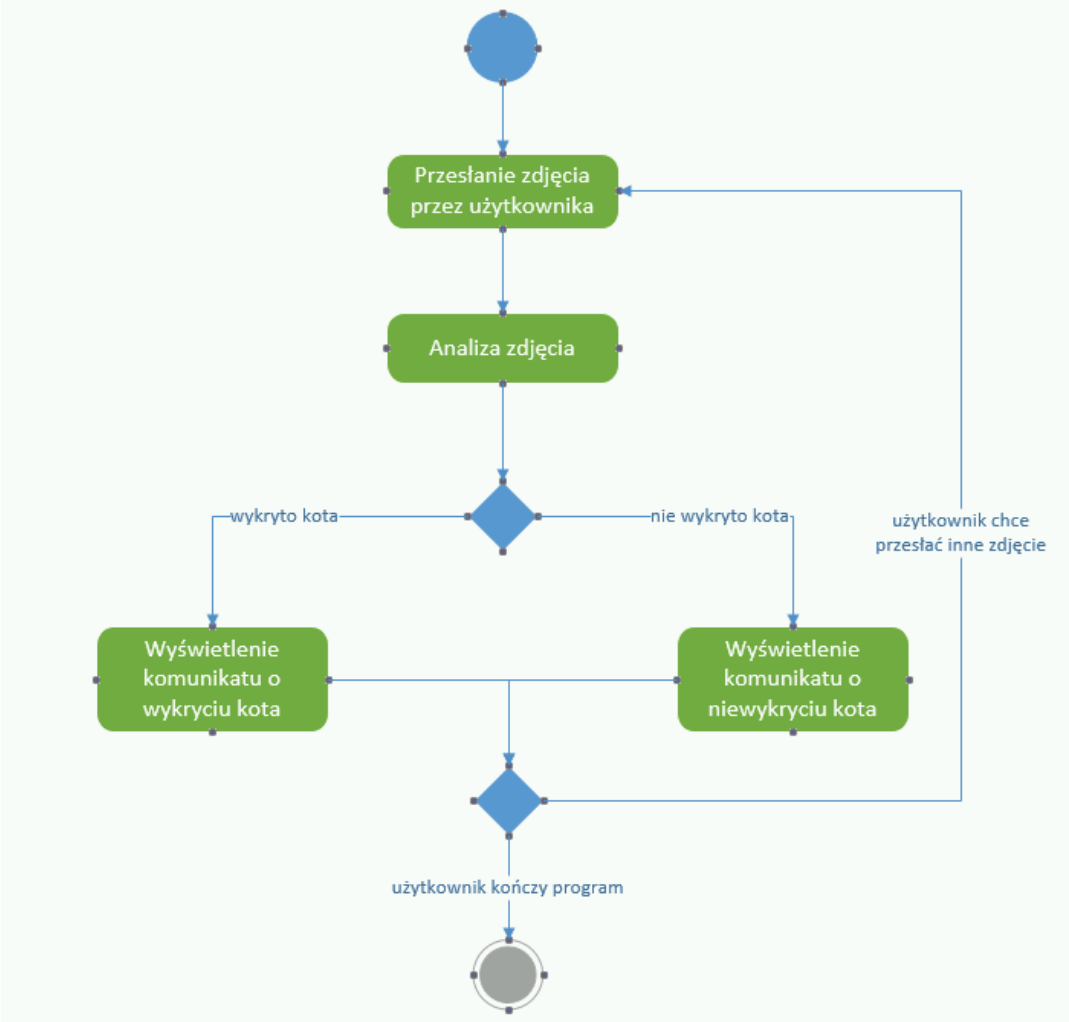
1-4. Jak w scenariuszu głównym.

5. System zwraca błąd związany ze sprawdzeniem poprawności.
6. System wyświetla użytkownikowi informację, że rozmiar wczytanego przez niego pliku za duży.

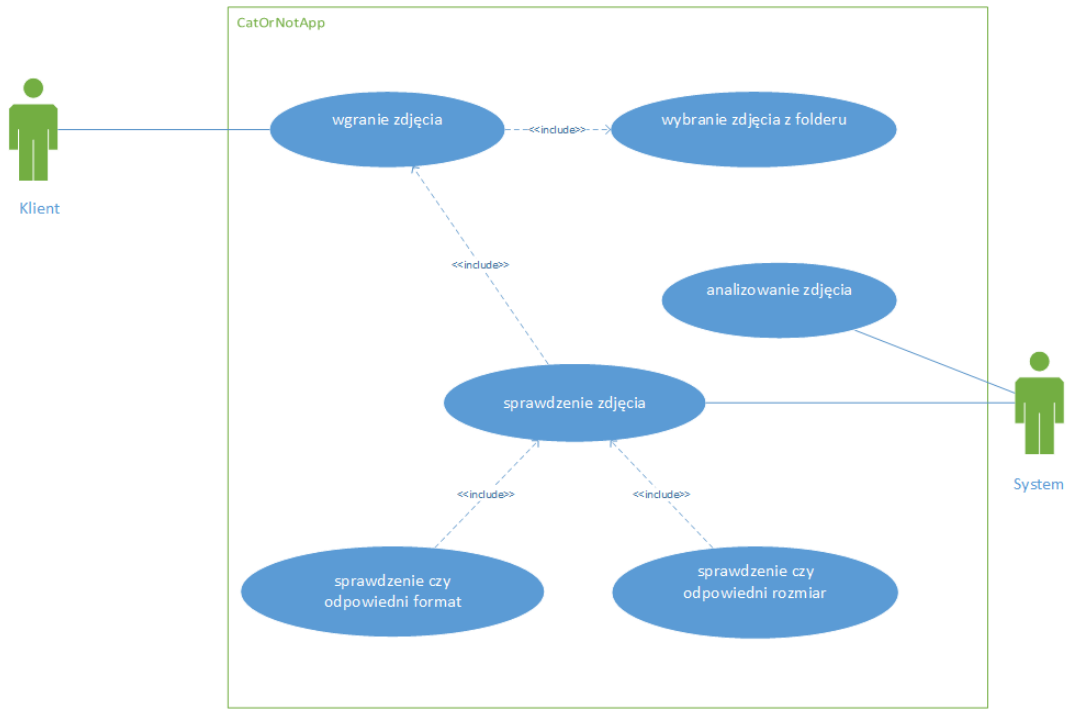
- Scenariusz alternatywny - wczytany plik nie jest plikiem obsługiwanym przez aplikację.

1-4. Jak w scenariuszu głównym.

5. System zwraca błąd związany ze sprawdzeniem poprawności.
6. System wyświetla użytkownikowi informację, że wczytany przez niego plik nie jest plikiem obsługiwanym przez aplikację.



Rysunek 1: Diagram czynności/aktywności



Rysunek 2: Diagram przypadków użycia

4. Specyfikacja wymagań niefunkcjonalnych

4.1 Platforma systemowa

Platformą systemową będzie system operacyjny z rodziny Windows lub Linux wraz z środowiskiem programistycznym dla języka Python.

4.2 Baza danych

Baza danych powinna przechowywać informację dotyczące historii przeprowadzonych analiz fotografii, takie jak:

- nazwę analizowanej fotografii (wraz z rozszerzeniem)
- rozmiar analizowanej fotografii (w MB)
- status przeprowadzonej analizy
- ewentualnie również procent prawdopodobieństwa uzyskany podczas analizy (0-100)

4.3 Język aplikacji

Aplikacja będzie dostępna w języku polskim lub angielskim. Przedstawione w powyższej dokumentacji *komunikaty* są jedynie przykładowe, natomiast ich treść może ulec zmianie, o ile przekaz komunikatu pozostanie bez zmian.

5. Projekt systemu

5.1 Interfejs użytkownika

Interfejs użytkownika, jak i sama struktura/wygląd aplikacji, będzie stworzony za pomocą języka *HTML*, służącego do opisu struktury budowanej aplikacji, oraz *CSS*, służącego do opisu formy prezentacji aplikacji.

W przypadku wykorzystywanego języka HTML - będzie on w wersji 5.

W przypadku CSS wykorzystamy gotową bibliotekę (framework), pozwalające na łatwiejsze, bardziej zgodne ze standardami projektowanie stron czy aplikacji internetowych. Tym frameworkiem będzie *bootstrap*.

5.2 Moduł wczytywania fotografii

Moduł będzie zawierał odpowiednie metody sprawdzające poprawność wczytanego pliku, pod kątem, czy jest on obsługiwanym plikiem (*.jpg*, *.png*, *.bmp*) i jego rozmiar nie przekracza 4MB.

5.3 Moduł analizujący i model sztucznej inteligencji

Moduł analizujący będzie wykorzystywał bibliotekę *ImageAI* posiadającą odpowiednie narzędzia do detekcji zadanych obiektów na fotografii na podstawie wczytanego modelu.

Moduł anlizujący będzie współpracował z wytrenowanym modelem sztucznej inteligencji *ResNet50*.

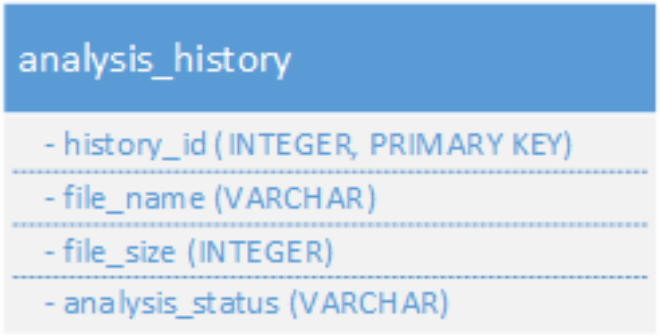
Minimalne wynikowe prawdopodobieństwo do stwierdzenia czy kot istnieje na fotografii wyniesie optymalne 80%, aby zapewnić jak najlepszą skuteczność przeprowadzonych analiz.

Model sztucznej inteligencji jest niezbędny do zapewnienia funkcjonalności analizy. Będzie wymagany dla funkcji wyszukującej/wykrywającej zadany obiekt (w tym przypadku kota) na fotografii.

5.4 Specyfikacja bazy danych

Projekt będzie wykorzystywał bazę danych SQLITE, tj. samodzielną, opartą na plikach bazę danych typu SQL. Dane będą przechowywane w specjalnym pliku o rozszerzeniu ***.db***.

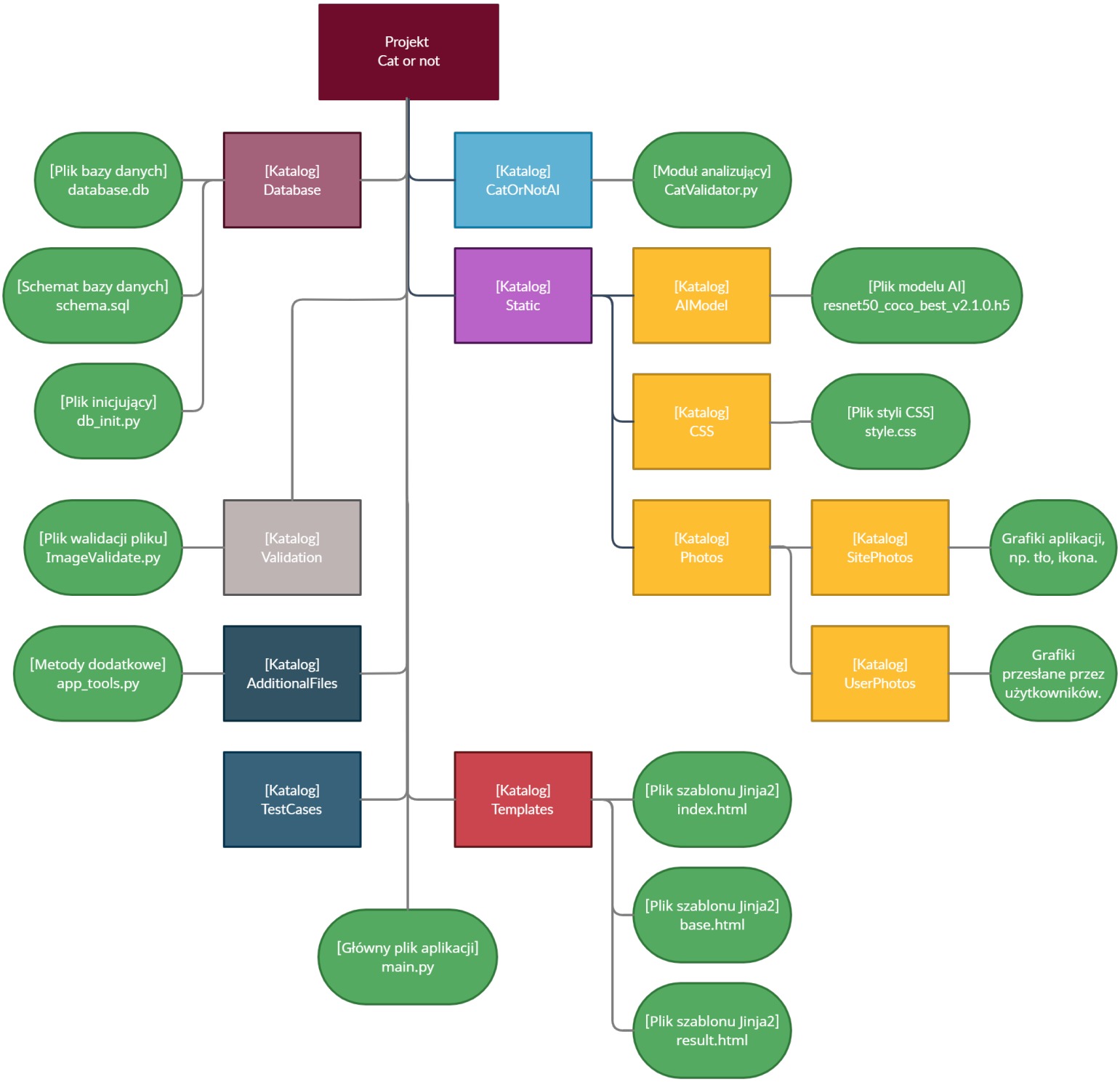
Poniżej znajduję się przykładowy diagram encji, prezentujący przykładową strukturę tabeli, którą baza danych będzie przechowywać, wraz z informacjami o typach danych znajdujących się w tabeli pól (kolumn, atrybutów).



Rysunek 3: Diagram struktury bazy danych

6. Opis wersji systemu

6.1 Struktura katalogowa projektu



6.2 Omówienie struktury katalogowej projektu

Wszystkie pliki i katalogi projektu znajdują się w głównym folderze *CatOrNot* .

- [KATALOG] ***CatOrNotAI*** - katalog zawierający moduł analizujący;
  - [PLIK] ***CatValidator.py*** - plik języka Python, będący modułem analizującym, zawiera on klasę CatValidator udostępniającą metodę *isCat(plik)*, która analizuje fotografię pod kątem istnienia na niej kota;
- [KATALOG] ***Static*** - katalog zawierający podkatalogi grupujące pliki statyczne, tj. style CSS, strony HTML, model sztucznej inteligencji, grafiki aplikacji i użytkowników;
  - [PODKATALOG] ***AIModel*** - katalog zawierający model sztucznej inteligencji *ResNet50*;
    - [PLIK] ***resnet50_coco_best_v2.1.0.h5*** - plik modelu sztucznej inteligencji *ResNet50* wykorzystywany przez moduł analizujący do analizy fotografii;
  - [PODKATALOG] ***CSS*** - katalog zawierający pliki arkuszy stylów CSS;
    - [PLIK] ***style.css*** - główny plik kaskadowych arkuszy stylów CSS, służący do nadania aplikacji odpowiedniego wyglądu;
    - [PLIK] ***bootstrap.min.css*** - główny plik framework'a *bootstrap*, wykorzystywanego do ostylowania wizualnej strony aplikacji;
  - [PODKATALOG] ***Photos*** - katalog zawierający podkatalogi, zawierające fotografie/pliki graficzne/ikony pogrupowane na wykorzystywane przez aplikację i przesłane przez użytkowników;
    - [PODPODKATALOG] ***SitePhotos*** - katalog zawierający pliki graficzne/ikony wykorzystywane przez aplikacje;
    - [PODPODKATALOG] ***UserPhotos*** - katalog zawierający fotografie przesłane przez użytkowników;
- [KATALOG] ***Templates*** - katalog zawierający szablony Jinja2 języka HTML dla języka Python;
- [KATALOG] ***Database*** - katalog zawierający pliki niezbędne dla bazy danych, tj. plik zawierający bazę danych, schemat bazy danych, plik inicjujący bazę danych;
  - [PLIK] ***database.db*** - plik bazy danych, w którym przechowywane są informacje/dane w niej zapisane;

  - [PLIK] ***schema.sql*** - plik schematu bazy danych, opisujący strukturę przechowywanych danych, ich typy i format;
- [PLIK] $\textbf{\textit{init}}_{db}.py - plik odpowiedzialny za uruchamienie bazy danych i wczytanie$
- [KATALOG] ***Validation*** - katalog zawierający pliki walidacyjne, sprawdzające poprawność np. wczytywanego pliku;
  - [PLIK] ***ImageValidator.py*** - plik walidacyjny, będący częścią modułu wczytywania pliku, zawiera on klasę ImageValidator udostępniającą metody sprawdzające, czy wczytany plik jest fotografią obsługiwaną przez aplikację oraz i rozmiar wczytanego pliku nie przekracza 4MB;
- [KATALOG] ***AdditionalFiles*** - katalog zawierający pliki dodatkowe projektu, np. plik z dodatkowymi metodami (funkcjami);
  - [PLIK] ***app_tools.py*** - plik języka Python, zawierający dodatkowe metody, niezbędne do poprawnego działania aplikacji, np. dotyczące historii wykonanych analiz;
- [KATALOG] ***TestCases*** - katalog zawierający pliki do przeprowadzania testów jednostkowych/automatycznych aplikacji;
- [PLIK GŁÓWNY] ***main.py*** - główny plik aplikacji, odpowiada za jej uruchomienie i działanie;

6.3 Wykorzystane technologie, języki programowania, biblioteki i komponenty dodatkowe

6.3.1 Wykorzystywane technologie, języki programowania

Nazwa	Wersja	Opis
HTML	5	Język znaczników stosowany do opisu struktury stron/aplikacji internetowych.
CSS	3	Język służący do opisu formy prezentacji stron/aplikacji internetowych.
Bootstrap	4.1.3	Framework kaskadowych arkuszy styli CSS udostępniający gotowe rozwiązania/style.
Python	3.8.6	Język programowania wykorzystywany w projekcie do stworzenia aplikacji.

6.3.2 Biblioteki, komponenty dodatkowe, wymagane zależności

Nazwa	Wersja	Opis
Flask	2.0.2	Prosty framework do budowania złożonych aplikacji internetowych.
Jinja2	3.0.3	Silnik szablonów dla języka programowania Python pozwalający na separację logiki aplikacji (Python) od jej warstwy prezentacyjnej (HTML).
MarkupSafe	2.0.1	Implementuje obiekt tekstowy, który zmienia znaki, dzięki czemu można go bezpiecznie używać w HTML i XML
Werkzeug	2.0.2	Werkzeug to kompleksowa biblioteka aplikacji internetowych.
Tensorflow	2.4.0	Kompleksowa platforma open source do uczenia maszynowego.
Tensorflow-estimator	2.4.0	Interfejs API TensorFlow, który znacznie upraszcza programowanie uczenia maszynowego, obejmujący uczenie, ocenę, przewidywanie i eksportowanie modelu.
Tensorboard	2.7.0	Narzędzie do dostarczania pomiarów i wizualizacji potrzebnych podczas przepływu pracy uczenia maszynowego.
Markdown	3.3.6	Biblioteka umożliwiająca konwersję tekstu Markdown do HTML na różne sposoby.
Keras	2.4.3	Biblioteka oprogramowania typu open source, która zapewnia interfejs Pythona dla sztucznych sieci neuronowych.
Keras-Preprocessing	1.1.2	Moduł wstępnego przetwarzania i rozszerzania danych w bibliotece Keras.
Pillow	7.0.0	Rozszerzenie, które dodaje obsługę grafiki np. otwieranie, modyfikowanie, zapisywanie plików graficznych.
Requests	2.26.0	Prosta biblioteka HTTP dla Pythona.
ImageAI	2.1.6	Biblioteka Pythona do tworzenia aplikacji i systemów z niezależnymi funkcjami Deep Learning i Computer Vision.
Urllib3	1.26.7	Przyjazny dla użytkownika klient HTTP dla Pythona.

Sqlite3	latest	Samodzielna, oparta na plikach baza danych SQL.

Pozostałe biblioteki/komponenty dodatkowe/wymagane zależności wraz z ich wersjami znajdują się w pliku ***requirements.txt*** w głównym folderze aplikacji projektowej i w następnej podsekcji ***Konfiguracja i wymagania środowiskowe***.

6.4 Konfiguracja i wymagania środowiskowe

Aplikacja*Cat or not* do poprawnego działania, oprócz samego kodu źródłowego aplikacji, wymaga również:

- interpreter języka Python w wersji 3.7 (3.7.6)
- modelu sztucznej inteligencji *ResNet50*
- instalacji odpowiednich żależności/bibliotek dodatkowych/modułów

Model sztucznej inteligencji wykorzystywany w aplikacji można pobrać klikając w następujący link:

- Model sztucznej inteligencji ResNet50.

Wymagania środowiskowe (zależności, moduły, biblioteki) niezbędne do poprawnego działania zawarte są w pliku ***requirements.txt*** znajdującym się w głównym folderze aplikacji.

Pełna lista zależności:

Nazwa	**Wersja**	**Nazwa**	**Wersja**
Flask	2.0.2	Click	8.0.3
Jinja2	3.0.3	Colorama	0.4.4
MarkupSafe	2.0.1	Itsdangerous	2.0.1
Werkzeug	2.0.2	absl-py	1.0.0
Tensorflow	2.4.0	astunparse	1.6.3
Tensorflow-estimator	2.4.0	cachetools	4.2.4
Tensorboard	2.7.0	certifi	2021.10.8
Markdown	3.3.6	charset-normalizer	2.0.9
Keras	2.4.3	cycler	0.11.0
Keras-Preprocessing	1.1.2	flatbuffers	1.12
Pillow	7.0.0	gast	0.3.3

Requests	2.26.0	google-auth	2.3.3
ImageAI	2.1.6	google-auth-oauthlib	0.4.6
Urllib3	1.26.7	google-pasta	0.2.0
grpcio	1.32.0	h5py	2.10.0
idna	3.3	importlib-metadata	4.10.0
Keras-Resnet	0.2.00	kiwisolver	1.3.2
matplotlib	3.3.2	numpy	1.19.3
oauthlib	3.1.1	opencv-python	4.5.4.60
opt-einsum	3.3.0	protobuf	3.19.1
pyasn1	0.4.8	pyasn1-modules	0.2.8
pyparsing	3.0.6	python-dateutil	2.8.2
PyYAML	6.0	Requests-oauthlib	1.3.0
rsa	4.8	scipy	14.1
six	1.15.0	termcolor	1.1.0
Tensorboard-data-server	0.6.1	Tensorboard-plugin-wit	1.8.0
typing-extensions	3.7.4.3	wrapt	1.12.1
zipp	3.6.0		

Zależności, o ile nie zostaną zainstalowane automatycznie, można zainstalować ręcznie. Aby to zrobić należy otworzyć plik ***requirements.txt***, kliknąć prawym przyciskiem myszy w oknie jego zawartości, a po pojawieniu się okna funkcyjnego należy wybrać odpowiednią opcję (taka opcja jest dostępna w środowisku *PyCharm*).

Wszystkie powyższe informacje/podane warunki/zależności/biblioteki itd. są niezbędne i wystarczające do poprawnej konfiguracji środowiska pod działanie aplikacji *Cat or not*.

7. Testy systemu

7.1 Plan testów

W trakcie realizacji projektu będą odbywały się testy sprawdzające poprawność działania modułów, pod kątem ich niezawodności, stabilności oraz spełniania założeń wynikających ze specyfikacji i wstępnej analizy.

7.2 Testowane komponenty

Testom podlegają następujące komponenty:

- interfejs użytkownika
- moduł wczytywania fotografii
- moduł analizujący

Testom podlega również współpraca powyższych komponentów zintegrowanych ze sobą.

7.3 Testy manulane

7.3.1 Raport nr 1 z dnia 16.12.2021r.

Podmioty testów:

- interfejs użytkownika
- moduł wczytywania fotografii

Pełny raport nr 1 z dnia 16.12.2021r. znajduje się w osobnym dokumencie.

7.3.2 Raport nr 2 z dnia 03.01.2022r.

Podmioty testów:

- moduł analizujący

Pełny raport nr 2 z dnia 03.01.2022r. znajduje się w osobnym dokumencie.

7.3.3 Raport nr 3 z dnia 10.01.2022r.

Podmioty testów:

- interfejs użytkownika
- moduł wczytywania fotografii
- moduł analizujący

- integracja modułów

Pełny raport nr 3 z dnia 10.01.2022r. znajduje się w osobnym dokumencie.

7.4 Testy automatyczne/jednostkowe

DO UZUPEŁNIENIA

8. Dokumentacja użytkownika

Dokumentacja użytkownika aplikacji Cat or not znajduje się w osobnym dokumencie.