НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:

Завідувач кафедри

_____ Олександр Коваль

«___»_____2020 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Програмне забезпечення веб-технологій та мобільних пристроїв»

спеціальності 121 «Інженерія програмного забезпечення»

на тему: «Система ідентифікації та трекінгу об'єктів за допомогою камер відеоспостереження»

Виконав:
студент IV курсу, групи ТІ-62
Полюга Дмитро Сергійович _____

Керівник:
старший викладач
Мірошніченко Іван Володимирович _____

Консультант:
к.т.н., доцент,
Шалденко Олексій Вікторович _____

Рецензент:
_____
_____ _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки: 121 Інженерія програмного забезпечення

Спеціалізація: Програмне забезпечення веб-технологій та мобільних пристроїв

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2020р.

ЗАВДАННЯ
на дипломну роботу студенту

_Полюзі Дмитру Сергійовичу_____
(прізвище, ім’я, по батькові)

1. Тема роботи ___Система ідентифікації та трекінгу об’єктів за допомогою камер відеоспостереження_____
_____

керівник роботи _____ст. викл. Мірошніченко Іван Володимирович_____
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”25” травня 2020р. № 1168-с

2. Строк подання студентом роботи «10» червня 2020р._____

3. Вихідні дані до роботи __мова програмування Python, фреймворк Django, бібліотека PyTorch, система керування базами даних PostgreSQL, інструмент для ізоляції та віртуалізації середи розробки Docker._____

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) проаналізувати предметну область ідентифікації та трекінгу об’єктів за допомогою камер відеоспостереження, сформувати задачі, шо повинна вирішувати система, проаналізувати аналогічні рішення, виділити їх недоліки та переваги, спроектувати систему за допомогою мови моделювання UML, розробити архітектуру системи, навести опис взаємодії складових частин системи, реалізувати алгоритм, створити програмне забезпечення веб-системи для демонстрації роботи алгоритму, проілюструвати користувацький інтерфейс системи_____

5. Перелік ілюстративного матеріалу
Програмні модулі, Алгоритм системи, Архітектура веб-додатку, Діаграма прецедентів, Діаграма послідовності._____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
	к.т.н., доц. Шалденко О. В.		

7. Дата видачі завдання ”11”____жовтня____ 2019 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	15.10.2019	
2.	Вивчення та аналіз задачі	13.04.2020	
3.	Розробка архітектури та загальної структури системи	20.04.2020	
4.	Розробка структур окремих підсистем	27.04.2020	
5.	Програмна реалізація системи	11.05.2020	
6.	Оформлення пояснювальної записки	18.05.2020	
7.	Захист програмного продукту	26.05.2020	
8.	Передзахист	10.06.2020	
9.	Захист	17.06.2020	

Студент _____ _Полюга Д. С._____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ _Мірошніченко І. В._____
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Мета роботи: розробити систему, що вирішить проблему трекінгу автотранспорту за допомогою ідентифікації номерних знаків через камери відеоспостереження для контролю доступу та аналізу трафіку.

Проаналізовано наявні програмні рішення, виділено їх переваги та недоліки та сформульовано вимоги й задачі, яким повинна відповідати система.

Обсяг дипломної роботи складає 56 сторінок, 25 рисунків, 14 використаних джерел літератури, та 3 додатка.

Функціонал системи розбито на підсистеми. Обґрунтовано доцільне використання кожного з підмодулів. Спроектовано та реалізовано багаторівневу архітектуру взаємодії всіх компонентів в рамках цілісної системи, що дозволяє легко розширювати та масштабувати систему, а також швидко замінити будь-яку з підсистем у разі необхідності.

Ключові слова: MVT архітектура, моделювання, ідентифікація об’єктів, трекінг об’єктів, веб-система.

ABSTRACT

Purpose: to develop a system that will solve the problem of vehicle tracking by identifying license plates through video surveillance cameras to control access and traffic analysis.

The existing software solutions are analyzed, their advantages and disadvantages are highlighted and the requirements and tasks to be met by the system are formulated.

The volume of the thesis is 56 pages, 25 figures, 14 sources of literature and 3 appendices.

The functionality of the system is divided into subsystems. The expedient use of each of the submodules is substantiated. A multi-level architecture of interaction of all components within the system is designed and implemented, which allows to easily expand and scale the system, as well as to quickly replace any of the subsystems if necessary.

Keywords: MVT architecture, modeling, object identification, object tracking, web system.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ – Програмне забезпечення

ПК – Персональний Комп’ютер

UI – User Interface

UML – Unified Modeling Language

OCR – Optical Character Recognition

API – Application Programming Interface

SDK – Software Development Kit

JSON – JavaScript Object Notation

HTTP – HyperText Transfer Protocol

REST – Representational State Transfer

IDE – Integrated Development Environment

MVT – Model View Template

ВСТУП

Ідентифікація об'єктів — поширена задача комп’ютерного зору, яка стосується виявлення та розмітки об'єктів певних класів на зображенні. Локалізацію об'єкта можна здійснювати різними способами, включаючи створення обмежувального поля навколо об'єкта або позначення кожного пікселя на зображенні, яке містить об'єкт (називається сегментацією).

Трекінг об'єктів — це завдання взяти набір попередньо виявлених об'єктів, створити унікальний ідентифікатор для кожного з них, а потім відстежити кожен з об'єктів, в момент, коли вони рухаються на кожному з кадрів відео, враховуючи попередньо присвоєний ідентифікатор.

Для трекінгу об'єктів відео алгоритм аналізує послідовні відеокадри та рух об'єктів між кадрами. Існують різноманітні алгоритми, кожен з яких має сильні та слабкі сторони. При виборі алгоритму важливо враховувати цільове використання системи. Кожен алгоритм має два основних компоненти візуального відстеження: цільове представлення та локалізація, а також фільтрація та об'єднання даних [1].

Технічно відстеження об'єктів починається з виявлення об'єктів — ідентифікації об'єктів на зображенні та призначення їм обмежувальних полів. Алгоритм відстеження об'єктів присвоює ідентифікатор кожному об'єкту, визначеному на зображенні, а на наступних кадрах намагається перенести цей ідентифікатор і ідентифікувати нове положення того ж об'єкта.

Серед головних проблем у наявних системах можна виокремити:

- перекриття і зникнення — об’єкти можуть непередбачувано переміщатися в кадрі, або виходити з нього, тому необхідно з'єднати їх з об'єктами, які раніше були на відео;
- розмиття руху — об'єкти можуть виглядати по-різному через власний рух або рух камери;

- точка зору — об’єкти можуть виглядати різними з різних точок, тому необхідно послідовно ідентифікувати один і той самий об’єкт з усіх точок;
- зміна масштабу — об’єкти у відео можуть різко змінити масштаб, наприклад, завдяки масштабуванню камери.

Метою роботи є створення системи ідентифікації та трекінгу об'єктів за допомогою камер та збереження отриманих результатів у базу даних для можливості подальшого аналізу та використання, основою якої буде алгоритм який буде виявляти об'єкти на відео та здійснювати їх трекінг. Такий продукт надасть можливість здійснювати трекінг окремих об'єктів за допомогою камер відеоспостереження.

На даний час дуже актуальна організація автоматичного контролю та трекінгу автомобілів, зокрема, здійснення якого виробляється за допомогою розпізнавання реєстраційного номерного знаку транспортного засобу.

Використовувати системи розпізнавання автомобільного номера та трекінгу машин можна для різних цілей:

1. Обмеження доступу. На території, що охороняються в'їзд дозволений далеко не всім і системи розпізнавання автомобільних номерів один з найзручніших і недорогих способів обмежити доступ небажаного автотранспорту.
2. Організація платного доступу для автомобілів. Подібні системи можна використовувати на платних парковках для ідентифікації транспортних засобів що в'їжджають і виїжджають в торгові і бізнес-центри, паркінги, призначені для стоянки автомобілів в темний час доби і багатьох інших, а також з їх допомогою можна автоматизувати процес оплати.
3. Управління потоками автотранспорту. Існує така необхідність для пропуску уповноважених транспортних засобів на територію. Наприклад, автомобілі спеціальних служб, автомобілі міських служб та інші. За допомогою систем розпізнавання автомобільних номерів можна гнучко налаштовувати рівні доступу і створювати території, на які в'їзд дозволений тільки окремим типам транспорту.
4. Управління часом знаходження автотранспортного засобу на території. У деяких випадках потрібно обмежити не в'їзд на територію, а час перебування на ній

автотранспортного засобу. Це може бути використано в аеропортах, на вокзалах, станціях метро, транспортних вузлах, прибудинкових територіях.

5. Реєстрація автотранспорту. При необхідності реєстрації проїжджаючого автотранспорту, наприклад, для збору статистики, яка дозволяє аналізувати транспортну завантаженість автомагістралей.

6. Відстеження автотранспорту внесеного, в список спостереження. За рахунок функції трекінгу система може відстежувати появу автотранспортних засобів з спеціально створеного для цих цілей списку спостереження і інформувати при їх появі.

Вхідні дані: відео або фото зображення автомобілів. Вихідні дані: текстовий номер автомобіля.

Наприклад, для роботи системи розпізнавання номерів з метою стеження за автомобілем встановлюється о кілька відеокамер. Потім зображення, отримане з цих камер, передається на сервер розпізнавання номерів. Система в режимі реального часу розпізнає номер автомобіля, який з’являється на відео і шукає в базі даних інформацію про появу автомобілю з таким номером на інших відео. Алгоритм аналізує частоту появи та різницю у часі між відео та, як результат, видає траєкторію руху автомобіля.

1. ПОСТАНОВКА ЗАДАЧІ РОЗРОБКИ СИСТЕМИ ІДЕНТИФІКАЦІЇ ТА ТРЕКІНГУ ОБ’ЄКТІВ ЗА ДОПОМОГОЮ КАМЕР ВІДЕОСПОСТЕРЕЖЕННЯ

Метою дипломної роботи є створення системи ідентифікації та трекінгу об'єктів за допомогою камер відеоспостереження. Отриманий результат повинен зберігатися у базу даних для можливості подальшого аналізу та використання. Основою цієї системи буде алгоритм який виявлятиме об'єкти на відео та здійснювати їх трекінг. Такий продукт надасть можливість здійснювати виділення окремих об’єктів на відео та їх трекінгу.

Задача полягає у розробці комплексної системи ідентифікації та трекінгу об’єктів за допомогою камер відеоспостереження.

1.1 Завдання роботи

Систему декомпозовано на список завдань, які мають бути вирішені:

1. Дослідити існуючі системи зі схожим функціоналом, проаналізувати їх переваги та недоліки.
2. Спроектувати архітектуру веб-системи за допомоги мови проектування UML.
3. Визначити вимоги до кожної з підсистем.
4. Описати як реалізована кожна з систем програмно.
5. Створити діаграму прецедентів для опису взаємодії користувачів із системою.
6. Створити діаграму прецедентів для опису потоків даних та взаємодії частин системи.
7. Розробити схему таблиць бази даних для системи.

1.2 Опис підсистем програмного комплексу

Система зчитування автомобільних номерів складається з наступних програмних модулів (рисунок 1.1):

- Модуль локалізації номера;
- Модуль розпізнавання;
- База даних та база зберігання медіа файлів;
- Користувацький інтерфейс.

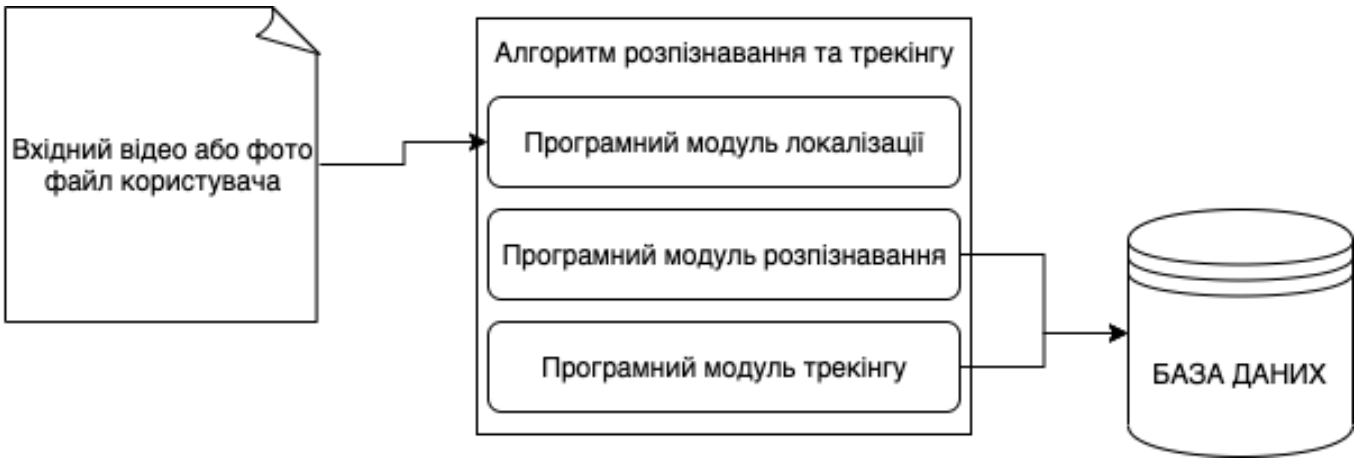

Рисунок 1.1 — Програмні модулі

Вимоги до модулів:

- Модуль легко модифікувати. При правильному проектуванні та архітектурі, розширення модулів обходиться без особливих тимчасових і технічних витрат.
- Модулі системи не повинні бути тісно пов'язані між собою, частини повинні бути частково абстрактним і самодостатнім. Кожна розроблена сутність повинна відповідати тільки за свою частину функціональності.
- Модуль повинен бути стабільним, передбачуваним, безпечним і надійним. Яким би простим код та цілий модуль не був в читанні, він повинен бути покритий тестами. Важливо не тільки кількість тестів, але і їх якість. З таким модулем не виникає проблем при запуску і налагодження, він не викликає змін у навколишньому середовищі.

- Захищеність. При написанні системних модулів не можна забувати про загальну безпеку продукту.
- Гнучкість модуля визначається легкістю, з якою можна його змінити, щоб виконати певну мету, яка не передбачалася під час написання. Фактично, гнучкий код - це специфічний код. Цільовою системою є та, яка максимально стисло виражає рішення проблеми. Гнучкі модулі легко читаються та розуміються, код містить меншу кількість синтаксичних елементів, таких як змінні, функції та класи. Він також швидкозмінний.
- Мають можливість багаторазового використання. Повторне використання - це використання існуючого програмного забезпечення для створення нових систем та алгоритмів. Це один із основних підходів сучасної розробки програмного забезпечення. API надають механізм включення повторного використання коду.

Кожен модуль системи містить в собі підсистеми які взаємодіють між собою.

Перелік підсистем:

- Система користувачів;
- Система завантаження вхідної інформації та файлів;
- Система зчитування завантаженої інформації;
- Система виведення результатів роботи програми.

Висновки до розділу

У розділі Постановка задачі розробка системи ідентифікації та трекінгу об’єктів за допомогою камер відеоспостереження було розглянуто вимоги до системи. Було описано всі задачі що повинно розв’язувати дане програмне забезпечення, розглянуто вхідні і вихідні дані додатку. Сформульовано вимоги до реалізації та перелік підсистем.

2. ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

Аналіз проблеми розпізнавання та трекінгу об’єктів. Об’єкт — автомобіль, предмет розпізнавання — автомобільний номер.

У ході аналізу предметної області було з'ясовано, що всі системи розпізнавання автомобільних номерів можна поділити на дві великі групи – програмні та апаратні системи розпізнавання автомобільних номерів.

Перша група виробляє розпізнавання автомобільних номерів безпосередньо в спеціалізованому програмному забезпеченні, встановленому на сервер відеоспостереження. Однак, так як процес розпізнавання вимагає великої ресурсоємності процесора, вимоги до апаратної частини сервера будуть високими, що призведе до того, що велика частина витрат припаде саме на вартість ПО і сервера.

Друга група систем з'явилася відносно недавно. На відміну від першої групи, великою перевагою таких систем є відсутність навантаження на сервер відеоспостереження, так як розпізнавання автомобільних номерів відбувається безпосередньо на камері відеоспостереження. За рахунок відсутності спеціалізованого програмного забезпечення, вартість системи розпізнавання автомобільних номерів в даному випадку фактично буде дорівнювати вартості смарт-камери відеоспостереження.

Програмні системи розпізнавання вперше з'явилися на українському і світовому ринку в 90-х роках. Плюсом даних систем є низька вартість і величезний вибір відеокамер, на відміну від смарт-камер з вбудованими функціями розпізнавання. Мінусами ж є висока вартість сервера, на якому встановлено програмне забезпечення по розпізнаванню номерів автівок, високе навантаження на локальну мережу, а також те, що в разі виходу сервера з ладу, вся система втрачає працездатність.

2.1 Огляд існуючих методів

Незважаючи на всі відмінності програмних і апаратних систем розпізнавання, їх об'єднує те, що всі вони працюють з алгоритмами і використовують певні методи обробки зображень.

Розглянемо різні методи і алгоритми розпізнавання автомобільних номерів. Процес ідентифікації автомобільного номера можна розділити на два етапи: попередній пошук номера (локалізація номерний пластини); розпізнавання символів.

2.1.1 Локалізація номерної пластини

Існує кілька способів попереднього пошуку номера:

1. Аналіз кордонів і фігур, контурний аналіз.

Найбільш очевидний спосіб виділення номера - пошук прямокутного контуру. Даний спосіб працює тільки при наявності ясної читабельності контуру, з досить високою роздільною здатністю і з рівною границею, при відсутності перешкод між камерою і номерний пластиною.

Суть контурного аналізу: проводиться фільтрація зображення для знаходження кордонів, потім проводиться виділення всіх знайдених контурів і їх аналіз. Даний спосіб непрактичний, оскільки контур номерного знаку може бути нечітким (брудним, зливатися з кольором машини, наявність перешкод і т.д.).

2. Аналіз частини кордонів.

Більш цікавим, стабільним і практичним представляється підхід, де аналізується тільки частина рамки номерного знаку. Спочатку виділяються контури, після чого знаходяться всі вертикальні прямі. Для будь-яких двох прямих, розташованих недалеко один від одного, з невеликим зрушенням по осі Y, з правильним відношенням відстані між ними до їх довжини, розглядається гіпотеза того, що номер розташовується між ними [2].

3. Статистичний аналіз, класифікатори.

Недоліки попередніх методів полягають в тому, що на реальних номерах, забруднених пилом, немає ні виражених кордонів, ні вираженої статистики, обов'язкових для їх коректної роботи. Кращі, але рідко використовувані методи - це методи, які спираються на різні класифікатори. Ці методи дають можливість аналізувати задану область на предмет наявності в ній характерних для номера відношень, точок або градієнтів і дозволяють знаходити номерний знак в складних і нетипових умовах. Наприклад, навчений каскад Хаара [3], робота якого проводиться за методом Віоли-Джонса, в основі якого лежать примітиви Хаара, що представляють собою розбивку заданої прямокутної області на набори різнотипних прямокутних областей.

У реальних алгоритмах багато методів побічно або безпосередньо базуються на наявності кордонів номера. Так само якщо при виявленні номера кордони не використовуються, вони можуть використатися в майбутньому аналізі.

Існують ситуації, в яких даний метод буде неефективний. Наприклад, при ситуації з досить чистим номером у світлій або хромованій рамці на білій машині, так як такі авто зустрічаються дуже рідко, в порівнянні з авто з брудними номерами і їх число може виявитися недостатнім для якісного навчання. У таких випадках метот буде не ефективним.

2.1.2 Розпізнавання символів

Методи розпізнавання символів:

1. Tesseract OCR.

Це відкрите програмне забезпечення, яке виконує автоматичне розпізнавання символів (букв) і тексту цілком. Переваги Tesseract полягають в тому, що ця програма існує для будь-яких ОС, легко навчається і стабільно працює. Недоліком же є те, що алгоритм дуже погано працює з битим, брудним і деформованим текстом. При розпізнаванні номерів за допомогою даного ПЗ ймовірність правильного розпізнавання лежить в діапазоні близько 20-30% з існуючої бази: найчистіші і прямі.

2. K-nearest.

Дуже простий для розуміння метод розпізнавання символів, який, незважаючи на свою примітивність, часто дає гарний результат.

Принцип роботи:

a. Попередньо записується певна кількість зображень реальних символів, коректно розбитих на класи;

b. Вводиться міра відстані між символами (якщо зображення бінарного типу, то операція XOR буде оптимальною);

c. При спробі розпізнавання символу, по черзі розраховується дистанція між розпізнаваним символом і всіма символами в базі. Серед k найближчих сусідів, можливо, будуть представники різних класів. Символ отримує такий же клас, як і у більшості сусідніх символів.

Недоліком даного методу може послужити лише необхідність швидкого розрахунку дистанції між зображеннями, а, отже, переведення їх до бінарного типу і використання XOR. Бінаризація зовсім непередбачувано змінює символ, і тому у випадку з забрудненими або потертими номерами будуть проблеми. Незважаючи на це при наявності великої бази з прикладами символів, зафіксованих в різних умовах, теоретично k-nearest - це все, що потрібно [4].

У багатьох випадках дуже важливо розуміти, як працює алгоритм. У даного методу є одне дуже важливе в цьому плані перевага: через простоту і «прозорість» алгоритму налагодження та налаштування на оптимальний результат проводиться без затруднень.

3. Кореляційний метод.

Кореляційний метод розпізнавання - метод розпізнавання образів, при якому для кожного класу об'єктів в декартовому просторі ознак задається еталонна область і будь-який розпізнаний об'єкт відноситься до класу, що відповідає найближчій еталонної області; остання формується шляхом допустимих перетворень одного або декількох еталонних векторів класу. Якщо зображення не приведено до бінарного типу, амплітуда сигналу невідома, отже, немає можливості визначити яскравість символу.

Розпізнавання зводиться до операції розрахунку коваріації вхідного сигналу з гіпотетичним (з урахуванням заданих зсувів і поворотів):

cov(X,Y) E[(X EX)(Y EY)] , де X - вхідний сигнал, Y - гіпотеза. Позначення E – математичне очікування.

Іноді такі методи називаються «template - matching» (зіставлення шаблонів). Дана назва повністю відображає їх суть, яка полягає у виборі зразків і наступному порівнянні вхідного зображення із зразками. У разі виникнення невизначеності щодо параметрів, проводиться перебір всіх можливих варіантів, або застосовуються адаптивні підходи.

Переваги методу:

- Передбачуваний і добре вивчений результат, якщо шум, навіть в малому ступені, відповідає обраній моделі;
- Можливість розпізнати навіть сильно запорошений/брудний/потертий символ при строго заданому шрифті.

Недоліки:

- Чималі витрати на процедури обчислення.

4. Адаптивне розпізнавання.

Існує два підходи:

а. Шрифтовий підхід.

Програма проводить аналіз і виміри різних характеристик шрифту, заносячи їх в свою базу еталонних характеристик [5]. Даний процес можна назвати навчанням програми. Після цього процесу шрифтова програма розпізнавання символів може розпізнати даний конкретний шрифт. Далі навчання повторюється для необхідних шрифтів, типи яких будуть залежати від області застосування програми [6].

До недоліків даного підходу можна віднести наступні фактори:

- Алгоритму необхідно заздалегідь знати шрифт;
- Необхідність наявності блоку налаштування на конкретний шрифт.

З іншого боку, у шрифтового підходу є істотна перевага: при наявності детальної апріорної інформації про символи, існує можливість побудови досить точних і надійних алгоритмів розпізнавання.

b. Безшрифтовий підхід.

Алгоритми, засновані на даному підході, вимірюють і аналізують різні характеристики (ознаки), властиві буквах будь-якого шрифту і розміру (кегля).

Недоліки цього підходу:

- Якість розпізнавання суттєво нижче, в порівнянні з шрифтовими алгоритмами;
- Низька значення коефіцієнта надійності розпізнавання.

Переваги:

- Універсальність (робота алгоритму не залежить від типу шрифту і інших його характеристик);
- Технологічність (простота навчання, можливість реалізувати автоматизоване навчання).

5. Нейромережі.

Штучна нейронна мережа (ШНМ, нейронна мережа) - це набір нейронів, з'єднаних між собою. Робота нейронної мережі полягає в перетворенні вхідного вектору даних в вихідний вектор [7].

Практично будь-яке завдання можна звести до задачі, розв'язуваної нейронною мережею. Для того, щоб нейронна мережа почала працювати, її необхідно навчити. Для цього в нейрони записується певний еталон, потім подається вхідний сигнал, який порівнюється з еталоном і відмінність між еталоном і вхідним впливом фіксується (додається до еталону). Чим більше буде вхідних впливів, тим краще навчиться нейромережа [8].

Переваги:

- при правильному налаштуванні і навчанні може працювати краще за інших методів;
- при великому навчальному масиві даних з'являється стійкість до спотворень символів [9].
- Недоліки
- складність реалізації;
- в багатошарових мережах неможлива діагностика аномальної поведінки.

2.2 Огляд існуючих систем

2.2.1 Програмне забезпечення «Автомаршал»

«Автомаршал» [10] являє собою самостійний додаток, розроблений науково-виробничою компанією «Малленом Системс», призначений для автоматичного розпізнавання та реєстрації автомобільних реєстраційних знаків.

Додаток має цілий ряд платних і безкоштовних доповнень (плагінів) для розширення функціоналу, наприклад, плагін "Парковка" допоможе повноцінно автоматизувати будь-які парковки, в тому числі платні. ПЗ «Автомаршал» встановлюється на комп'ютер з операційною системою сімейства Windows. До комп'ютера по мережі Ethernet підключається одна або більше IP-відеокамер. Система розпізнає номери автомобілів за рахунок аналізу відео, яке надходить із камер, і зберігає інформацію про всі автомобілях (дата / час проїзду, напрямок проїзду, зображення автомобіля, номер автомобіля, найменування камери, коментарі та ін.) в базу даних. Функціонал системи дозволяє створювати призначені для користувача списки доступу, формувати різні види звітів, завантажувати сторонні бази даних, управляти шлагбаумами і світлофорами, взаємодіяти з системами контролю і управління доступом (СКУД) і системами відеоспостереження

Базова версія ПЗ «Автомаршал» включає в себе:

- Розпізнавання державних реєстраційних знаків автомобілів на відео;
- Ведення бази даних автомобілів, збереження їх зображень з можливістю перегляду і редагування записів;
- Підтримка користувацьких баз даних автомобілів, з можливістю завантаження бази даних в ручному режимі з файлів формату * .Xls, * .xlsx і * .csv;
- Автоматичну перевірку розпізнаних номерів по призначених для користувача баз даних;
- Візуальне і звукове сповіщення оператора при збігу розпізнаного номера з записом в призначених для користувача базах даних автомобілів;

- Пошук в базі даних виявлених транспортних засобів за вказаними користувачем критеріям, формування та друк звіту за результатами пошуку;
- Створення списків доступу для автоматичного управління шлагбаумом з урахуванням результату розпізнавання номера машини.

Ключові характеристики системи «Автомаршал»:

- Ймовірність розпізнавання до 98% при швидкості автомобілів до 150 км/год;
- Підтримка номерів СНД і Євросоюзу (29 країн);
- 2 паралельних алгоритму розпізнавання;
- До 8 каналів розпізнавання на одному комп'ютері;
- Швидкий старт без додаткових налаштувань;
- Web-клієнт для дистанційного доступу з будь-якого браузера або мобільного пристрою;
- Управління зовнішніми пристроями: шлагбаумами, світлофорами;
- SDK для вбудовування в сторонні системи;
- Безкоштовна технічна підтримка протягом 1 року після придбання.

2.2.2 Система автоматичного розпізнавання «SecurOS Auto»

«SecurOS Auto» [11] являє собою програмну систему широкого застосування для розпізнавання автомобільних номерів. Даний плагін встановлюється на інтеграційну платформу «SecurOS». Програмне рішення в автоматичному режимі проводить розпізнавання номерного знака і звіряє його з внутрішніми списками. Підключаючи стаціонарне і мобільне обладнання спостереження і контрольно-пропускного режиму, існує можливість використання системи «SecurOS» для підвищення рівня безпеки, пов'язаного з транспортними засобами.

Можливості системи:

- Розпізнавання номерних знаки понад 60 країн світу;
- Розпізнавання номерів по окремих кадрів без використання відео;

- Швидко налаштувати оновлення для розпізнавання номерів нових стандартів;
- Формувати базу даних розпізнаних номерів зі збереженням супутньої інформації про дату, час і місце виявлення автомобіля, швидкості і напрямку його руху, а також посилання на відеофрагмент;
- Організувати пошук розпізнаних раніше номерів по заданих параметрам;
- Налаштувати необхідні реакції системи на розпізнавання номера, на результати пошуку;
- Оперативно інформувати оператора (підтримується, в тому числі, і голосове сповіщення) і / або відправити повідомлення (e-mail, SMS та ін.) зовнішнім службам про результати розпізнавання номера і / або про результати зіставлення даних про автомобіль зі списками номерів;
- Автоматично генерувати звіти різних видів на основі результатів розпізнавання, пошуку по базах даних.

Можливості інтеграції та спільне використання:

- Стаціонарне і мобільне обладнання спостереження й прилади контролю та обліку: бар'єри та шлагбауми, автоматичні ворота, радари, телекомунікаційні мережі, термінали оплати та ін.;
- Сторонні програмні комплекси (автоматизовані системи розрахунку оплати, СУБД та ін.);
- Інтеграція з «SecurOS Traffic Scanner» (системою автоматичного визначення фактів порушення правил дорожнього руху) – можливість побудувати потужний комплексне рішення для контролю і управління ситуацією на дорогах.

Все це дозволяє використовувати «SecurOS Auto» для будь-яких завдань, які передбачають контроль номерів транспортних засобів, включаючи контроль діяльності господарських служб, дотримання пропускного режиму, виявлення потенційно небезпечних ситуацій на дорогах. Система дозволяє підключати апаратні засоби і використовувати програмні продукти сторонніх розробників, забезпечує повнофункціональне використання інтегрувального обладнання, включаючи

передачу відео- і аудіо потоків, управління телеметрії, зберігання і відтворення записаної інформації.

Поряд з базовими механізмами, на основі «SecurOS Auto» розроблені допоміжні сервіси - модулі інтеграції з апаратними засобами, використання яких дозволяє істотно розширити список завдань, що вирішуються відеоаналітикою.

Ключові переваги SecurOS Auto в бізнес-рішеннях:

- Запобігання несанкціонованого проїзду на об'єкти;
- Контроль пересувань автотранспорту по території;
- Контроль ввезення і вивезення вантажів, їх відповідності супровідної документації;
- Автоматичне ведення журналу проїздів транспортних засобів з декількох КПП з можливістю подальшого аналізу часу перебування автомобілів на і поза територією;
- Контроль дій персоналу КПП;
- Організація диференційованого в'їзду на охоронювані об'єкти.

2.2.3 АСР «ANPR Access»

На відміну від більшості аналогічних систем, «ANPR Access» [12] від компанії Nedap AVI вже включає в себе всі необхідні компоненти системи розпізнавання номерів: відеокамеру, інфрачервоне підсвічування і блок обробки даних, поміщені в одному компактному корпусі, стійкому до атмосферного впливу. управління пристроєм відбувається по локальній мережі за допомогою web-інтерфейсу.

Доступ до всіх налаштувань через web-інтерфейс дозволяє здійснити первинне налаштування та управління під час експлуатації, переглядати список подій і логічних помилок при розпізнаванні номерів.

АСР «ANPR Access» розпізнає номери відповідно до зазначеного державним форматом номера. Підтримуються формати практично всіх європейських країн, в тому числі України, Росії, Білорусії.

У пам'яті «ANPR Access» можна задати список номерів машин, доступ яким дозволений (white list) або навпаки, список номерів машин, доступ яким заборонений (black list). «ANPR Access» дозволяє здійснити управління шлагбаумом за рахунок наявності релейного виходу. Можливість зберігати не тільки розпізнані номери, але і фотографії машин на внутрішню карту пам'яті, а також всі вищезгадані можливості роблять «ANPR Access» привабливим пристроєм для організації автономного контролю доступу автотранспорту на парковку або територію, що охороняється.

Дане рішення, з точки зору системи контролю доступу, нічим не відрізняється від будь-якого стандартного зчитувача карт і не вимагає ніякої додаткової інтеграції на стороні контролера.

2.2.4 CarFlow II

Основою системи ТОВ Мега Піксела- CarFlow II є відеопроцесор MegaFrame-4 (розробка мегапікселів) на базі PCI-шини. Розмір оброблюваного зображення дорівнює 768 × 288 сірих або кольорових пікселя.

Одним з найбільш важливих параметрів системи зчитування автомобільних номерів є її швидкодія. У системах Мега Піксела використовуються оригінальні нейроподібні алгоритми. Алгоритм може працювати як з передніми, так і з задніми номерами. Зовнішній запуск, при цьому, не потрібен. Це дуже важливо для систем з широким полем зору, так як дуже часто відразу кілька автомобілів можуть одночасно перебувати в зоні контролю. Алгоритм Мега Піксела мультизонний - до 16 номерних пластин можуть детектуватиметься одночасно. При цьому швидкість обробки не залежить від кількості зон детекції. Час детекції для CarFlow II - 60мс.

Після детекції частина зображення, що містить номер (192x24 пікселя) піддається процедурі передобробки: видалення помилкових спрацьовувань детектора, "чистка", визначення типу номера, збільшення і бінаризація. Час видалення помилкового спрацьовування - 0.5мс. Повний час передобробки однієї зони - 3.0мс.

Заключною стадією процесу є OCR (оптичне розпізнавання символів). Час розпізнавання становить приблизно 20мс.

Іншим важливим параметром є мінімально допустима контрастність зображення номерної пластини. Розробник констатує наступне обмеження для своєї системи: "Різниця між середнім рівнем яскравості символів і середньою яскравістю фону номерний пластини повинна становити не менше 25% від повного розмаху відеосигналу". Використовуючи дане визначення контрастності, можна констатувати, що система може працювати з 5% контрастністю зображень.

Висновки до розділу

Було проведено аналіз існуючих алгоритмів та наявних на ринку програмних продуктів для визначення номерних знаків та. Ці комерційні проекти мають досить високу вартість $ 1500-3000. Заявлена точність розпізнавання зазвичай завищена і не збігається з реальною. При випробуванні демо-версій надійно розпізнаються лише чисті номери високої контрастності. У підсумку на заявлену точність 90-98%, припадає реальна - 80-87%.

Використовувані алгоритми локалізації і розпізнавання номерних знаків не публікуються, тільки лише деякі компанії називають їх типи. Для розпізнавання зазвичай використовують нейро-подібні і шаблонні алгоритми.

Вагомим обмежуючим фактором використання системи є максимальна швидкість автомобіля, при якій програма здатна локалізувати і розпізнати номер на рухомому транспорті. На що впливають по-перше спосіб установки камери - висота і нахил; а по-друге швидкодію обробки зображення автомобіля. Отже розробка системи є доцільною на сьогодні.

3. ЗАСОБИ РОЗРОБКИ

3.1 Середовище розробки

Visual Studio Code поєднує простоту редактора вихідного коду з потужним інструментом для розробників, як-от завершення та налагодження коду IntelliSense. Цикл редагування-збирання-налагодження, що не потребує зусиль, означає менший час роботи з середовищем та більше часу на виконання ідей та розробку.

VS Code включає вбудовану підтримку розробки Node.js з JavaScript та TypeScript, що працює на тих самих базових технологіях, які керують Visual Studio. Код VS також включає інструменти для веб-технологій, таких як JSX / React, HTML, CSS, SCSS, Less та JSON.

3.2 Мови програмування, розмітки та стилів, фреймворки

3.3.1 Мова програмування Python

Python - потужна і проста для вивчення мова програмування. Він дозволяє використовувати ефективні високорівневі структури даних і пропонує простий, але ефективний підхід до об'єктно-орієнтованого програмування. Поєднання витонченого синтаксису, динамічної типізації в інтерпритованій мові робить Python ідеальною мовою для написання сценаріїв та прискореної розробки додатків в різних сферах і на більшості платформ.

Інтерпретатор Python та стандартна бібліотека знаходяться у вільному доступі в вигляді вихідних кодів та бінарних файлів для всіх основних платформ на офіційному сайті Python і можуть поширюватися без обмежень. Крім цього на сайті містяться дистрибутиви і посилання на численні модулі сторонніх розробників для мови Python, різні програми і інструменти, а також додаткова документація.

Інтерпретатор Python може бути легко розширений за допомогою нових функцій і типів даних, написаних на C/C ++ (або іншими мовами, до яких можна отримати доступ із C). Також Python можна застосовувати як мову розширень для настроюваних додатків.

3.3.2 Мова програмування JavaScript

JavaScript - це мова програмування, що є прототипно-орієнтованою. Він відображає мову ECMAScript, чиїм прототипом спочатку і був. Перша варіація з'явилася ще в 1995 році і з тих пір постійно вдосконалювалася, поки не прийшла до нинішнього вигляду.

Найчастіше ця мова використовується в розробці додатків у браузерах з метою надання їм інтерактивності і «жвавості».

3.3.3 Мова розмітки HTML5

HTML5 - остання версія стандарту HTML.

Термін має два визначення:

- Нова версія мови HTML, з новими елементами, атрибутами і новою поведінкою;
- Набір технологій, що дозволяє створювати різноманітні сайти і Web-додатки.

Для розробників HTML5 допомагає писати зрозумілий семантичний код, управляти багатьма процесами на сторінці своїми стандартними методами, без використання javascript або сторонніх плагінів і сервісів. Це означає, що вирішуються деякі проблеми кросбраузерності, оскільки браузери однаково реалізують нові можливості. Також html5 робить зручною роботу в мережі для звичайних користувачів. Наприклад, збільшується швидкість роботи, використання браузера стає зручнішим.

3.3.4 Мова стилів CSS3

CSS3 - це новий стандарт оформлення HTML документів, що значно розширює можливості попереднього стандарту CSS2.1. Багато можливостей, які були важкодоступні в CSS2.1, тобто вимагали використання додаткових зовнішніх програм (таких як Adobe Photoshop), скриптів (таких як JavaScript) або спеціальних "хитрощів" можуть легко досягатися в CSS3 за рахунок використання нових властивостей оформлення.

Можливості:

- Створення елементів зі згладженими кутами;
- Створення лінійніх та сферичних градієнтів;
- Гнучке оформлення фонової картинки елементів;
- Додання до елементів і тексту елементів тіні;
- Використання різних шрифти (не боячись при цьому, що вони будуть не підтримуватися браузером користувача);
- Створення анімацій і різних ефектів переходів.

3.3.5 Фреймворк Django 3

Django - фреймворк для веб-розробки на Python. Django - це програмний каркас з багатими можливостями, що підходить для розробки складних сайтів і веб-додатків, написаний на мові програмування Python.

Django - фреймворк для веб-додатків на мові Python. Один з основних принципів фреймворка - DRY (do not repeat yourself). Веб-системи на Django будуються з одного або декількох додатків, які рекомендується робити відчужуваними і з можливістю підключення. Це одна з помітних архітектурних відмінностей цього фреймворка від деяких інших (наприклад, Ruby on Rails). Також, на відміну від багатьох інших фреймворків, обробники URL в Django конфігуруються явно, а не автоматично задаються зі структури контролерів.

3.3.6 Бібліотека NumPy

NumPy - одна з найпотужніших бібліотек Python. Вона використовується для обчислення масивів та матриць даних. NumPy - це числова бібліотека Python з відкритим кодом.

NumPy містить багатовимірний масив та матричні структури даних. Він може бути використаний для виконання ряду математичних операцій на масивах, таких як тригонометричні, статистичні та алгебраїчні процедури.

Бібліотека містить велику кількість математичних, алгебраїчних та перетворювальних функцій. NumPy також містить генератори випадкових чисел.

NumPy - це обгортка навколо бібліотеки, реалізованої в C.

3.3.7 Бібліотека PyTorch

PyTorch - науковий обчислювальний пакет на основі Python, який використовує потужність одиниць графічної обробки. Це також одна з дослідницьких платформ для глибокого навчання, побудована для забезпечення максимальної гнучкості та швидкості. Він відомий тим, що надає два найбільш необхідних модуля машинного навчання; а саме, тензорні обчислення із сильною підтримкою прискорення графічного процесора та побудова глибоких нейронних мереж на автоградієнтних системах.

3.3 Система управління базою даних

PostgreSQL - це система управління базами даних загального призначення та об'єктно-реляційною системою, найдосконаліша системою баз даних з відкритим кодом. PostgreSQL була розроблена на базі POSTGRES 4.2.

PostgreSQL була розроблена для роботи на UNIX-подібних платформах. Однак PostgreSQL також була розроблена портативною, щоб вона могла працювати на різних платформах, таких як Mac OS X, Solaris та Windows.

PostgreSQL вимагає дуже мінімальних зусиль, що підтримуються через свою стабільність. Тому, якщо користувач розробляє додатки на основі PostgreSQL, загальна вартість володіння є низькою порівняно з іншими системами управління базами даних.

3.4 Програмне забезпечення ПК

Вимоги до програмного забезпечення ПК:

- MacOS 10.15.0 +;
- Інтерпретатор Python 3.8;
- Docker Compose;
- Hypercorn application server;
- Nginx web server.

3.4.1 Docker Compose

Compose - це інструмент для визначення та запуску багатоконтейнерних програм Docker. За допомогою Compose використовується файл YAML для налаштування служб своєї програми. Потім за допомогою однієї команди створюються та запускаються всі служби зі своєї конфігурації.

Використання Compose - це триетапний процес:

a. Визначення середовища програми за допомогою Dockerfile, щоб воно могло бути відтворене в будь-якому місці;

b. Визначення сервісів, які складають вашу програму в docker-compose.yml, щоб вони могли працювати разом в ізольованому середовищі;

c. Запуск докер-композиції і Compose запускає весь додаток.

3.4.2 Hypercorn

Hypercorn - це сервер додатку, веб-сервер ASGI, заснований на бібліотеках sans-io hyper, h11, h2, wsproto та натхненний Gunicorn. Hypercorn підтримує специфікації HTTP/1, HTTP/2, WebSockets (понад HTTP/1 та HTTP / 2), ASGI/2 та ASGI/3.

3.4.3 NGINX

NGINX - це програмне забезпечення з відкритим кодом для розміщення в Інтернеті, зворотного проксі, кешування, балансування завантаження, потокового передавання медіа тощо. Він розпочинався як веб-сервер, розроблений для досягнення максимальної продуктивності та стабільності. У додаток до можливостей HTTP-сервера, NGINX також може функціонувати як проксі-сервер для електронної пошти і зворотний проксі-сервер і балансир завантаження для HTTP, TCP і UDP-серверів.

Мета NGINX полягала в тому, щоб створити найшвидший веб-сервер, і підтримка цієї майстерності все ще залишається центральною метою проекту. NGINX послідовно перемагає Apache та інші сервери в орієнтирах, що вимірюють продуктивність веб-сервера. З часу початкового випуску NGINX веб-сайти розширилися від простих HTML-сторінок до динамічного, багатогранного контенту. NGINX тепер підтримує всі компоненти сучасного Web, включаючи WebSocket, HTTP / 2 та потокове передавання декількох форматів відео (HDS, HLS, RTMP та інші).

Висновки до розділу

В розділі розглянуто засоби розробки системи ідентифікації та трекінгу об’єктів за допомогою камер відеоспостереження. Описано програмні засоби, мови розмітки та стилів, мови програмування, фреймворки, бібліотеки та середовище розробки, вимоги до програмного забезпечення комп’ютера та систему управління базами даних. У розділі наведено підтвердження доцільності використання цих засобів для розробки сучасної системи.

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

4.1 Опис алгоритму

Система ідентифікації та трекінгу об’єктів має наступний (рис. 4.1.1) набір модулів та алгоритм роботи:

a. Обробка вхідних відео та отримання набору кадрів;
b. Сегментація кадрів, виділення областей з номерним знаком [13];
c. Розпізнавання номерного знаку з кожної виділеної області [14];
d. Відображення номерного знаку на поточному кадрі;
e. Створення результуючого відео з опрацьованих кадрів;
f. Опрацювання та аналіз даних з кожного відеопотоку;
g. Виведення результатів ідентифікації та трекінгу.

Рис. 4.1.1 – Алгоритм системи

Модуль виділення областей з номерним знаком розроблений на основі алгоритму виявлення в реальному часі YOLO (You Only Look Once). Це один із

найефективніших алгоритмів виявлення об'єктів, який також охоплює багато інноваційніх ідей, що виходять із спільноти комп'ютерних досліджень.

Виявлення об'єктів - одна з класичних проблем у комп’ютерному зорі, де алгоритм щоб розпізнає, які об’єкти знаходяться всередині даного зображення, а також де саме вони знаходяться. Проблема виявлення об'єкта є більш складною, ніж класифікація, яка також може розпізнавати об'єкти, але не вказує, де об’єкт розташований на зображенні. Крім того, класифікація не працює на зображеннях, що містять більше одного об’єкта.

YOLO популярний тим, що досягає високої точності, а також може працювати в режимі реального часу. Алгоритм "дивиться лише один раз" на зображення в тому сенсі, що для прогнозування йому потрібно лише один прохід через нейронну мережу. Після цього він виводить розпізнані об'єкти разом з обмежуючими полями.

З YOLO один CNN одночасно прогнозує кілька обмежувальних границь та ймовірності класу для кожної з них. Ця модель має ряд переваг перед іншими методами виявлення об'єктів:

- Працює надзвичайно швидко;
- Бачить весь образ під час тренувань та тестування, тому він неявно кодує контекстуальну інформацію про типи об'єктів, а також їх зовнішній вигляд;
- Вивчає узагальнюючі уявлення предметів, тому навчаючись на природних зображеннях і тестуючись на несправжніх зображеннях, алгоритм перевершує інші методи виявлення.

YOLO ділить вхідне зображення на сітку SxS. Якщо центр об’єкта потрапляє в комірку сітки, ця комірка сітки відповідає за виявлення цього об’єкта (ми призначаємо об'єкт до комірки сітки, де існує центр об’єкта).

YOLO запускає класифікацію та локалізацію для кожного з осередків сітки SxS одночасно. Оскільки нейромережа класифікації та локалізації може виявляти лише один об’єкт, це означає, що будь-яка комірка сітки може містити лише один об'єкт. Через цю ідею сітки YOLO стикається з деякими проблемами:

a. Оскільки використовується сітка SxS, і будь-яка сітка може виявити лише один об'єкт, максимальна кількість об'єктів, яку модель може виявити, становить SxS;

b. Якщо комірка сітки містить більше одного об'єкта, модель не зможе виявити їх усіх, це проблема виявлення близьких об'єктів, від якої страждає YOLO;

c. Об'єкт може знаходитись у більш ніж одній сітці, тому модель може виявити один об’єкт більше одного разу (у більш ніж одній сітці).

Усі SxS комірки сітки виявляються одночасно, і тому YOLO вважається дуже швидкою моделлю. Кожна з комірок сітки SxS прогнозує B обмежувальних границь і для кожної границі модель виводить результат точності. Ці результати оцінки відображають, наскільки модель впевнена, що поле містить предмет. Використовуючи цей показник, можна запобігти виявлення фону моделлю, тому якщо в комірці немає жодного об'єкта, результативність балів повинна бути нульовою.

На додаток до достовірності оцінки C модель видає 4 числа ((x, y), w, h) для відображення місця розташування та розмірів передбачуваного обмежувального поля.

Координати (x, y) являють собою центр вікна відносно меж комірки сітки. Ширина та висота прогнозуються відносно всього зображення, тому 0 <(x, y, w, h) <1.

YOLO використовує єдину згорнуту мережу для одночасного прогнозування декількох обмежувальних коробок та ймовірностей класів для цих ящиків. Ця мережа надихається моделлю GoogleNet для класифікації зображень, але замість початкових модулів, які використовує GoogLeNet, YOLO просто використовує 1 × 1 прошароки скорочення на 3 × 3 згорткових прошарків. У нього 24 згорткових прошарки, а потім 2 повністю з’єднаних прошарки.

Алгоритм вибору обмежувальних границь для кожного класу:

a. Відкинути всі поля зі значенням C <C-порогове (наприклад, C <0,5);

b. Відсортувати прогнози, починаючи з найвищого значення C;

c. Обрати поле з найвищим значенням C і вивести його як результат.

Модуль трекінгу реалізовано за наступним алгоритмом:

- Система проходить по кожному з визначених номерів за певний проміжок часу для того, щоб згрупувати однакові номерні знаки з різних камер;

- Використовується алгоритм визначення схожості тексту для групування неповних номерів (наприклад номерний знак «AA2345E» буде згруповано з «AA2345EK»);
- Для того, щоб вирішити проблему перекриття одного авто іншим, система вважає, що якщо один і той же номерний знак було знайдено з певним, невеликим проміжком часу – то цей номерний знак знаходився увесь цей проміжок часу у кадрі.

4.2 Опис архітектури веб-системи

Взаємодія між користувачем та системою відбувається шляхом запиту клієнта - відповіді сервера (request-response). Сервер має монолітну архітектуру MVT (Model-View-Template), що дозволяє швидко масштабувати весь функціонал одночасно.

Структура MBT має такі три частини:

1. Модель: Модель буде виконувати роль інтерфейсу даних. Вона відповідає за збереження даних. Це логічна структура, що стоїть за всією програмою і представлена базою даних (як правило, реляційними базами даних, такими як MySql, Postgres).

2. Перегляд: Перегляд - це інтерфейс користувача - те, що зображено у браузері під час використання веб-сайту. Він представлений файлами HTML / CSS / Javascript та Jinja.

3. Шаблон: Шаблон складається з статичних частин потрібного виводу HTML, а також спеціального синтаксису, що описує, як буде вставлено динамічний контент.

На рисунку 4.2.1 зображено детальну схему архітектури розробленого додатку.

На даній схемі зображено цикл опрацювання запитів та передавання відповідей між клієнтом та сервером:

1. Запит від клієнта надходить на веб-сервер.

2. Веб-сервер перенаправляє запит на найменш завантажений сервер додатку.

3. Сервер додатку використовує запущену копію системи, для того, щоб зіставити url запиту з необхідним view використовуючи url resolution.

4. View в разі необхідності інтеракції з базою даних використовує model.

5. Відбувається процес генерації результуючої веб-сторінки шляхом вставлення динамічного контенту в необхідний template.

6. Згенерована сторінка надсилається клієнту.

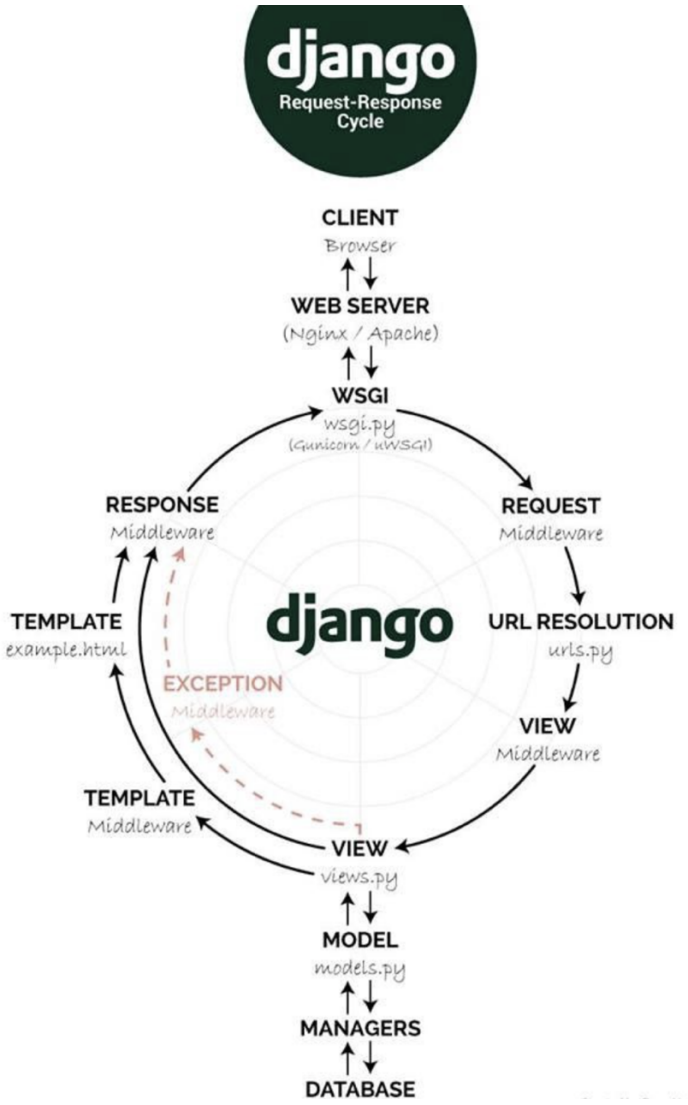

Рис. 4.2.1 – Архітектура веб-додатку

4.3 Взаємодія користувача з системою

Для візуалізації взаємодії користувача з розробленою системою було змодельовано діаграму прецедентів (рисунок 4.3.1).

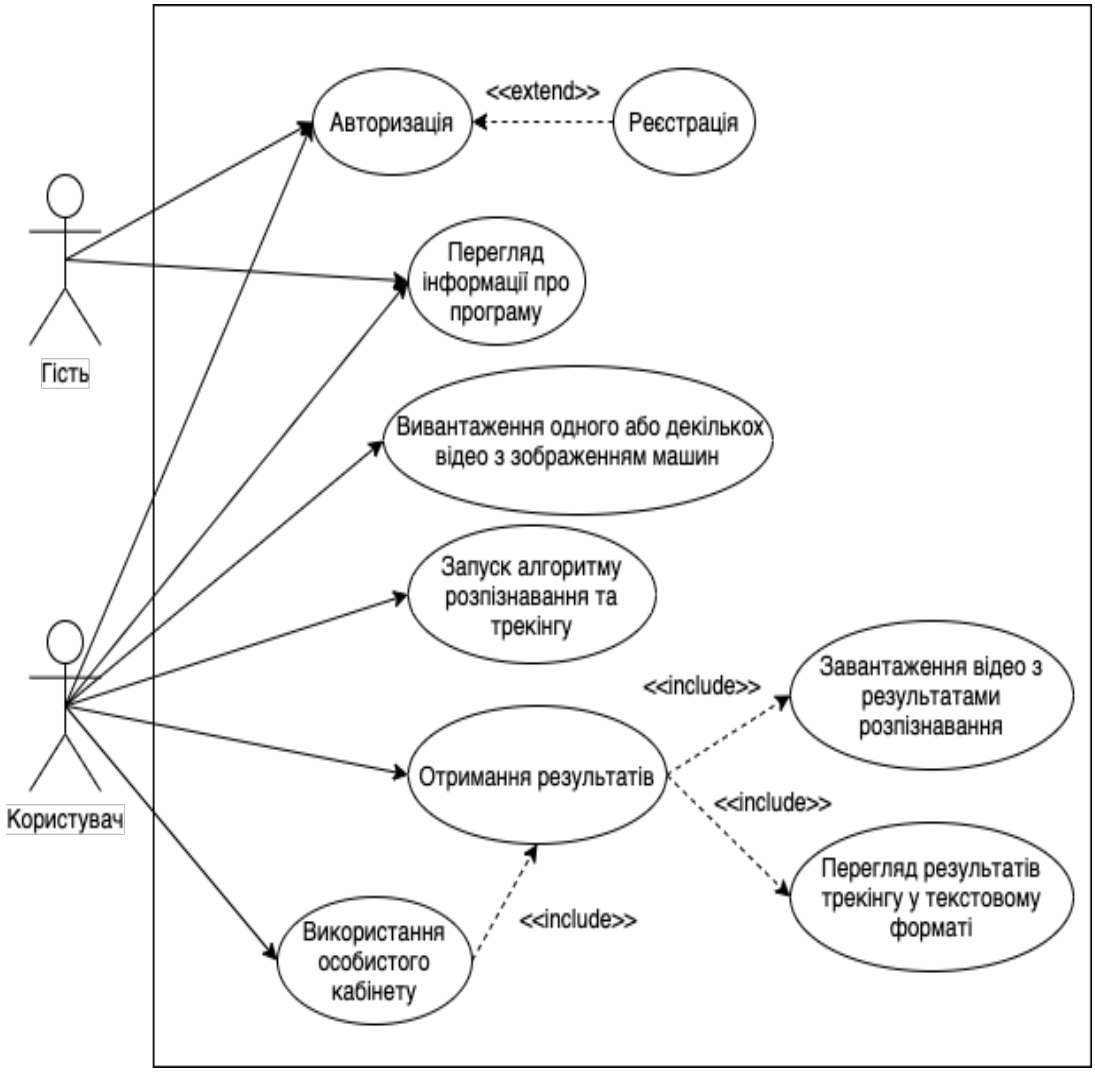

Рис 4.3.1 – Діаграма прецедентів

На діаграмі прецедентів зображено як різні види користувачів (актори) системи з нею взаємодіють та який рівень доступу має кожен з них.

4.4 Діаграма послідовності

Діаграми послідовності, які зазвичай використовуються розробниками, моделюють взаємодію між об'єктами в одному випадку використання. Вони ілюструють, як різні частини системи взаємодіють між собою, щоб виконувати функцію, і порядок, в якому взаємодії відбуваються при виконанні конкретного випадку використання.

Діаграма послідовностей показує різні частини роботи системи в "послідовності", щоб досягнути мети використання (рисунок 4.4.1).

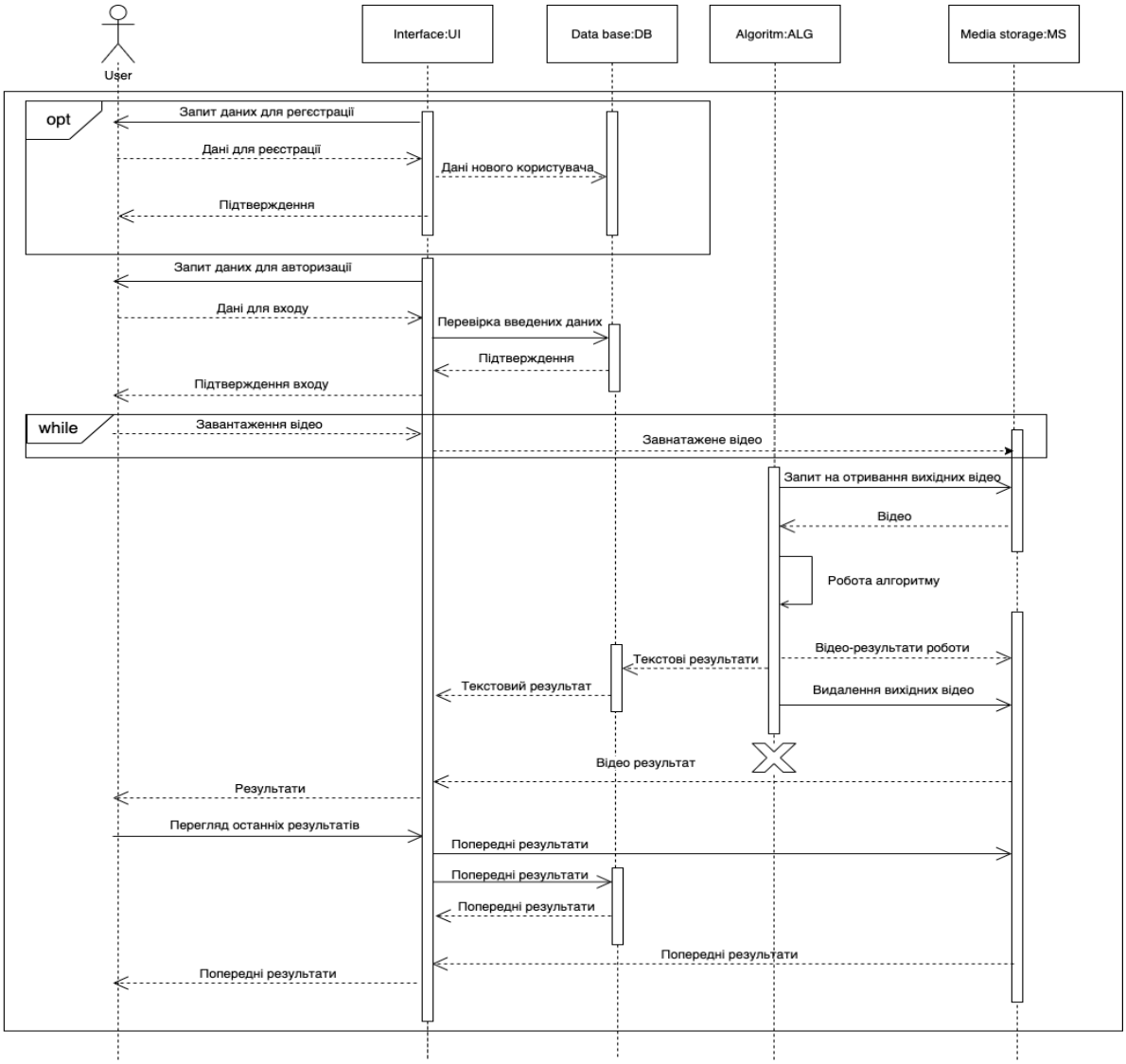

Рис 4.4.1 – Діаграма послідовностей

Висновки до розділу

У розділі 4 було розглянуто програмну реалізацію системи ідентифікації та трекінгу об’єктів за допомогою камер відеоспостереження. Наведено опис алгоритму, опис архітектури системи, діаграму прецедентів та діаграму послідовностей.

5. РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Програмна система розроблена за допомогою веб-технологій, тому підтримується багатьма браузерами.

5.1 Інсталяція та системні вимоги

Оскільки систему розроблено за допомогою сучасних веб-технологій, для її використання не потрібна інсталяція. Для взаємодії з системою необхідний веб-браузер з підтримкою актуальних веб-стандартів. Також для використання системи у користувача має бути стабільне підключення до інтернету.

5.2 Сценарій роботи з системою

Для роботи з системою користувачу необхідно створити обліковий запис. Для цього необхідно перейти на сторінку реєстрації з головного меню системи (рисунок 5.1). На рисунку 5.2 зображено форму реєстрації.

Рисунок 5.1 — Головне меню системи

Якщо користувач попередньо зареєстрований в системі, йому необхідно перейти на сторінку авторизації з головного меню системи (рисунок 5.1). На рисунку 5.3 зображено форму авторизації.

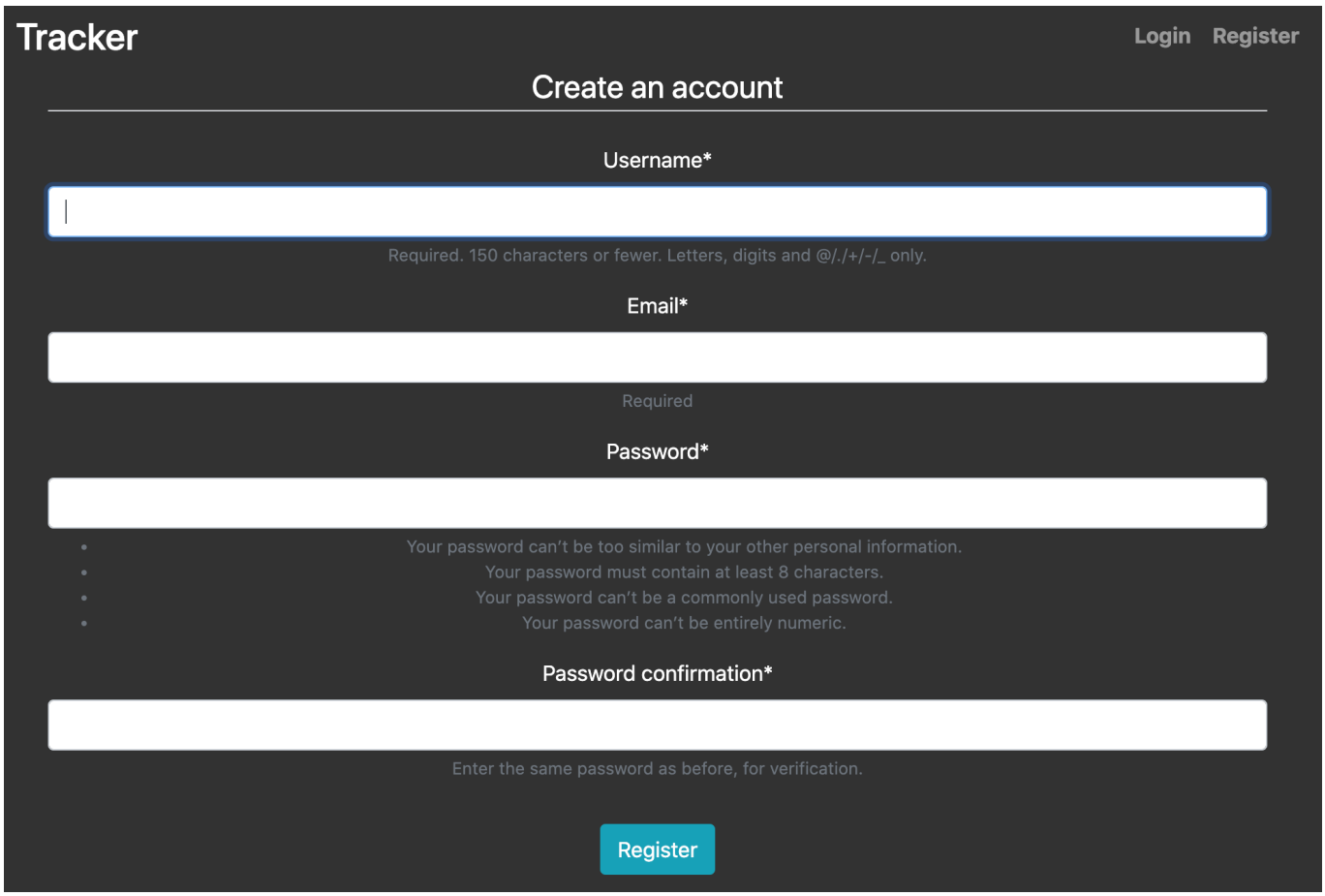

Рисунок 5.2 — Форма реєстрації

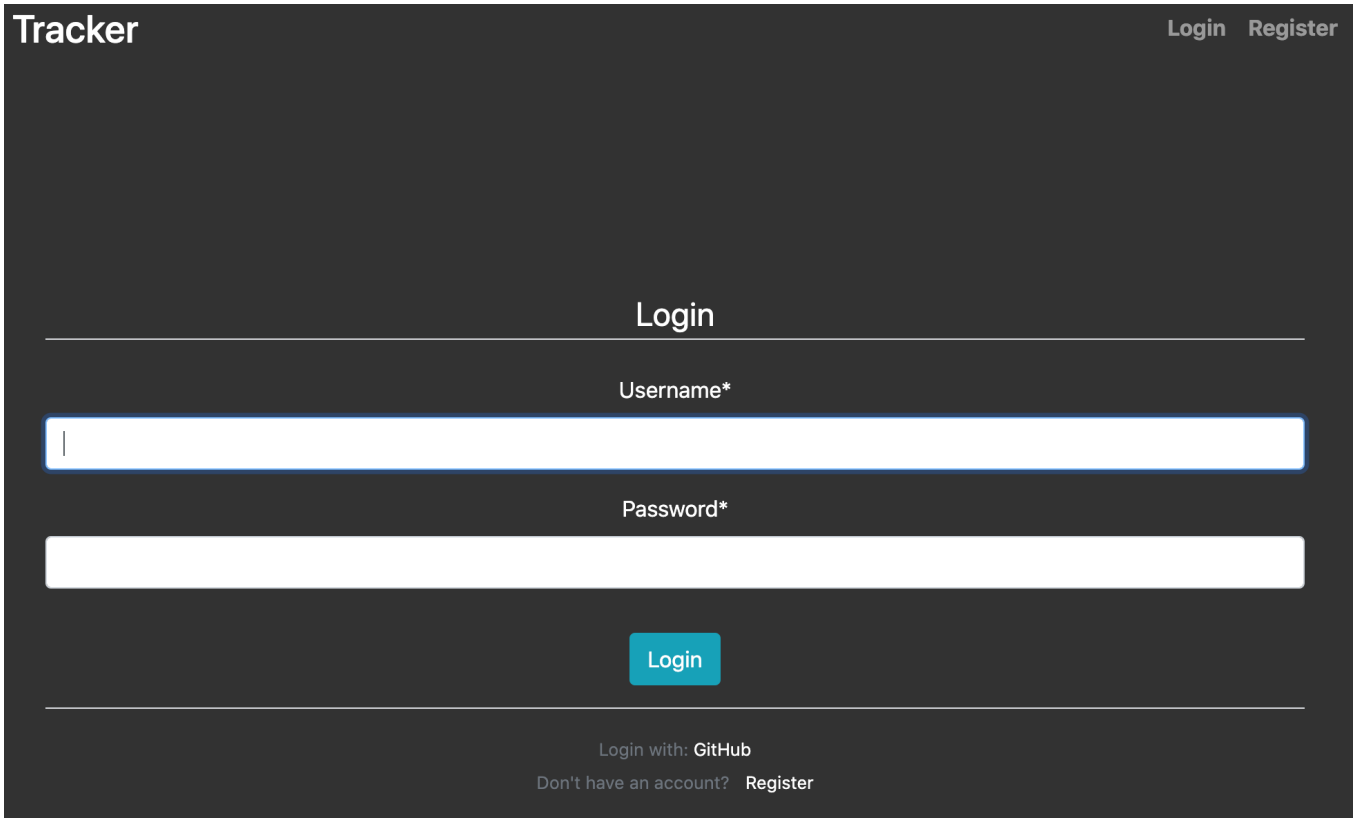

Рисунок 5.3 — Форма авторизації

У користувача також є можливість реєстрації та авторизації за допомогою соціальної мережі GitHub. Для цього користувачу необхідно перейти на сторінку авторизації, або реєстрації та обрати необхідну соціальну мережу GitHub (рисунок 5.4).

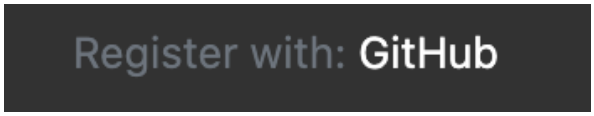

Рисунок 5.4 — Меню вибору реєстрації за допомогою GitHub

Після цього користувача буде перенаправлено на сайт обраної соціальної мережі. На рисунку 5.5 зображено реєстрацію користувача за допомогою платформі GitHub.

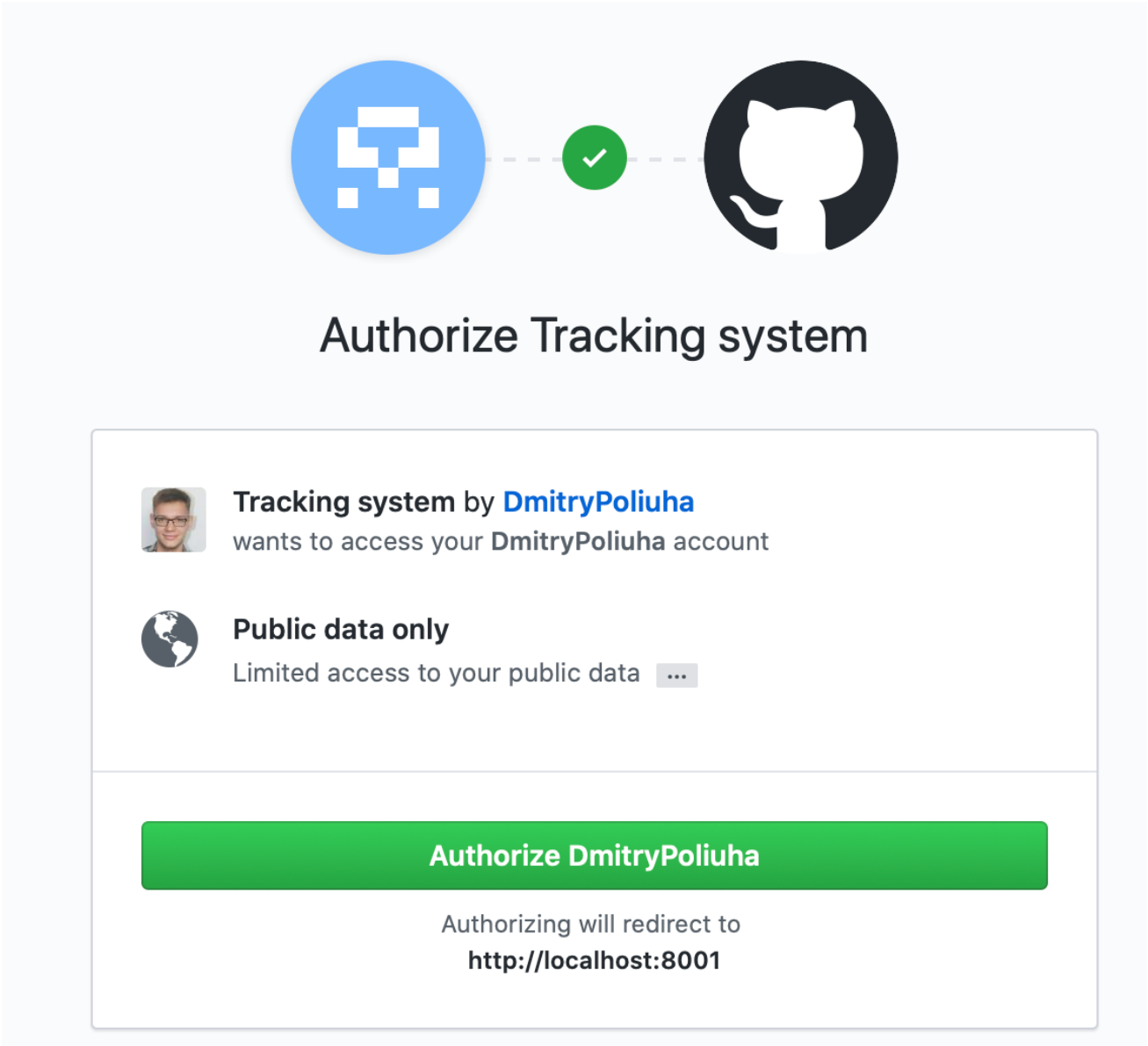

Рисунок 5.5 — Реєстрація за допомогою платформи GitHub

Після авторизації користувач отримує доступ до розширеного функціоналу системи. На рисунку 5.6 зображено оновлене меню користувача.

Рисунок 5.6 — Головне меню для авторизованого користувача

Користувач має змогу завантажити один, або декілька відео файлів у форматах «.MOV», або «.MP4» для початку роботи системи (рисунок 5.7). Після завантаження відео файлів користувач має змогу перевірити їх назву та тип (рисунок 5.8), а також видалити всі завантажені відео в разі помилки (рисунок 5.9).

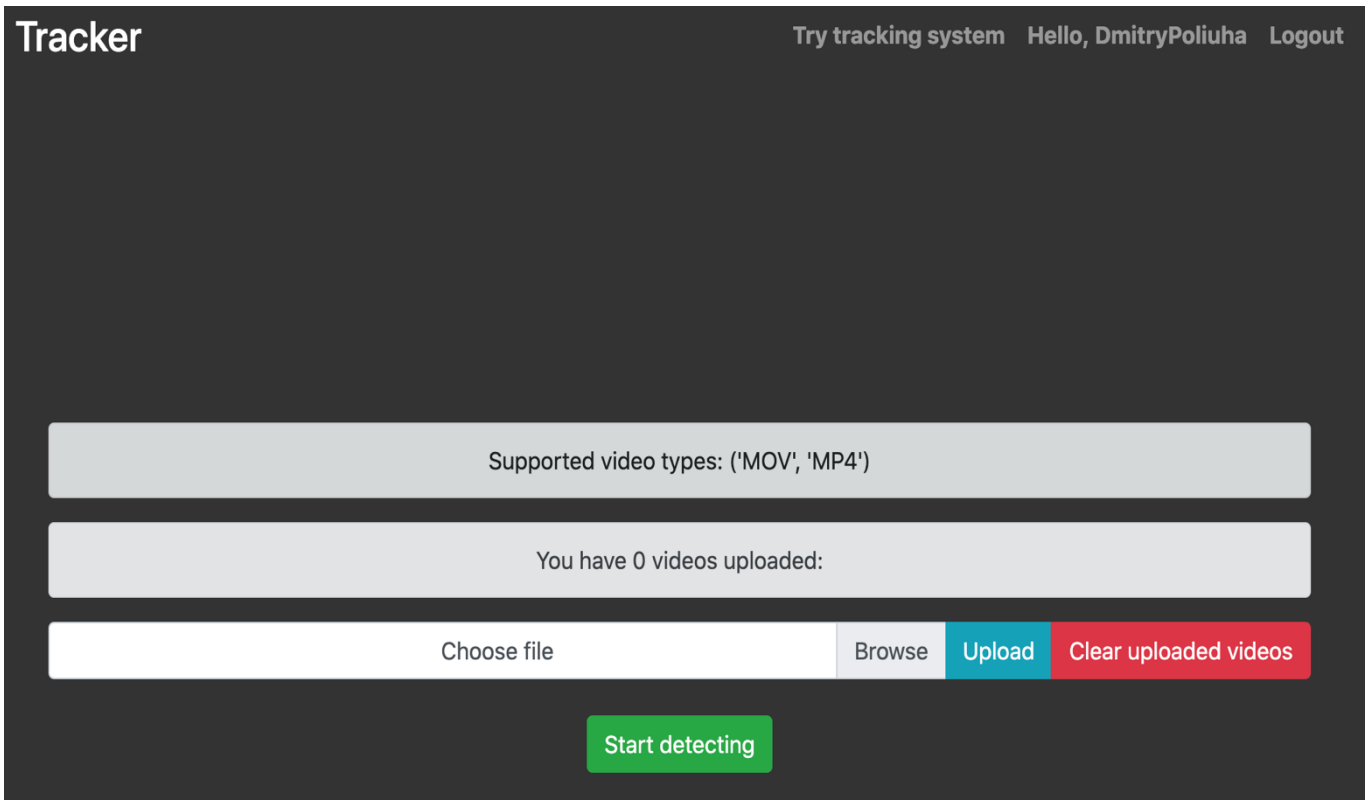

Рисунок 5.7 — Вікно завантаження файлів

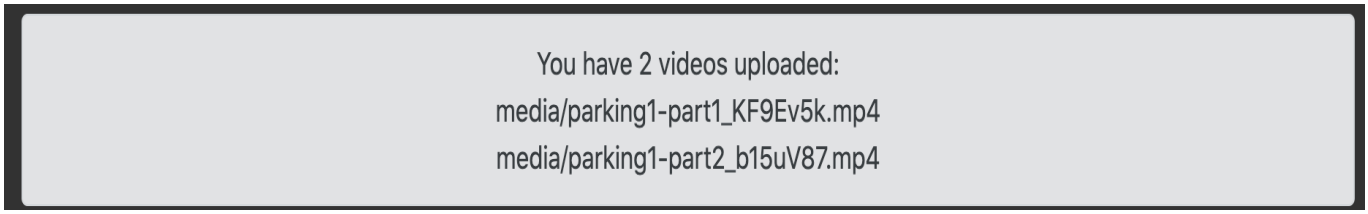

Рисунок 5.8 — Назва та тип завантажених файлів

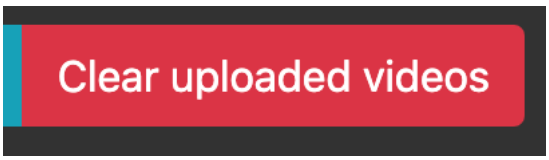

Рисунок 5.9 — Кнопка видалення всіх завантажених відео файлів

Перевіривши коректність завантаження вхідних файлів користувач має змогу розпочати обробку вхідних даних (рисунок 5.10).

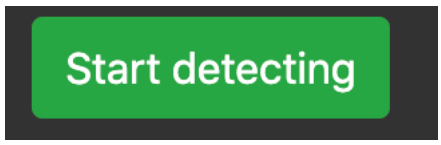

Рисунок 5.10 — Кнопка початку опрацювання всіх завантажених відео файлів

Під час процесу опрацювання користувач побачить анімацію опрацювання, що буде присутня до тих пір, поки система не опрацює кожен відео файл.

Спочатку завантажене відео розбивається на набір кадрів (рисунок 5.11, рисунок 5.12).

Рисунок 5.11 — Кадр із зображенням першого авто із завантаженого на аналіз відео

Рисунок 5.12 — Кадр із зображенням другого авто із завантаженого на аналіз відео

Після цього алгоритм пошуку номера на фото знаходить розташування номеру та алгоритм розпізнавання номеру відображає результат розпізнавання на фото (рисунок 5.13, рисунок 5.14).

При генерації результатів ідентифікації та трекінгу система генерує парні зображення авто — реєстраційний номер (рисунок 5.15) а також таблицю, в якій користувач має можливість переглянути повні результати ідентифікації та трекінгу для кожного з авто на відео (рисунок 5.16).

Рисунок 5.13 — Кадр із зображенням першого авто з розпізнаним номерним знаком

Рисунок 5.14 — Кадр із зображенням другого авто з розпізнаним номерним знаком

Рисунок 5.15 — Приклад результатів розпізнавання

Total results for camera

Plate	Seconds of video	Total seconds
CA4668GE	3.1	0.0
CAM688C	3.15	0.0
CA6668BE	3.2 3.25 3.3 3.35 3.4 3.45 3.5 3.55 3.6 3.65	0.44999999999999973
CA16480	4.35	0.0
CA8186BI	4.45 4.5 4.55 4.6 4.65 4.7	0.25
CA1708CA	6.55 6.6	0.04999999999999982
CA8700CA	6.65 6.7 6.75	0.09999999999999964
CA81348	10.95	0.0
CA4315AX	11.05	0.0
CA6115AX	11.15	0.0
CAAJ11AM	11.2	0.0
CA8115	11.25	0.0
CA2728HP	16.35 16.45 16.5 16.6 16.65 16.7	0.34999999999999787
CA7728HF	16.55	0.0

Рисунок 5.16 — Приклад результатів трекінгу

Висновки до розділу

У розділі Робота користувача з системою було розглянуто як наведено приклади роботи системи при різних діях користувача. Наведено зображення екрану ключових кроків роботи системи. Описано дії користувача для роботи з системою ідентифікації та трекінгу об’єктів за допомогою камер відеоспостереження.

6. ТЕСТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

6.1 Unit-тестування

Unit testing (юніт тестування або модульне тестування) - полягає в ізольованій перевірці кожного окремого елемента шляхом запуску тестів в штучному середовищі. Для цього необхідно використовувати драйвери і заглушки. Поелементне тестування – найперша можливість реалізувати вихідний код. Оцінюючи кожен елемент ізольовано і підтверджуючи коректність його роботи, точно встановити проблему значно простіше ніж, якби елемент був частиною системи.

Мета юніт-тестування - це ізоляція окремих частин програми і демонстрація того, що окремо ці частини працездатні.

Даний тип тестування в основному виконується програмістами.

Модульне тестування дозволяє проводити рефакторинг, будучи впевненим, що модуль коректно працює.

Юніт-тестування також допомагає усунути сумніви з приводу окремих модулів. Також може бути використаний спочатку для тестування окремих частин, а потім програми в цілому.

Модульний тест можна розглядати як «живий документ» для тестованого класу.

Django поставляється із власним тестовим набором у каталозі тестів бази коду. Це зроблено для того, щоб усі тести проходили постійно.

Всі тести Django використовують тестову інфраструктуру, яка постачається з Django для тестування програм.

Під час розробки програмного продукту для дипломної роботи було створенуо юніт тести на основі наявних у Django модулів. Для кожної підсистеми кожного модуля було створено набір тестів для забезпечення якості створеної системи.

6.1.1 Приклади тестів

Тестовий випадок 1 - перевірка типу формату вхідної інформації користувача. Для початку роботи системи користувач має завантажити один або декілька відеофайлів. Для уникнення непорозумінь та помилок система опрацьовує відео лише форматів .mp4 або .mov.

Вхідна інформація: 5 файлів hello.txt, one.mp4, 12_day.png, work.pdf, video.mov

Вихідна інформація: обробка файлів one.mp4 та video.mov коректна

Результат: тест пройдено.

Тестовий випадок 2 - перевірка правильності розбивання відеопотоку на кадри. Одним з частин алгоритму є модуль розбивки відеофайлу на певну кількість фото для спрощення подальшої роботи алгоритму.

Вхідна інформація: відеофайл формату .mp4 довжиною 10 секунд

Вихідна інформація: 20 кадрів у форматі .jpg

Результат: некоректна розбивка відеопоток, тест не пройдено (рисунок 6.1).

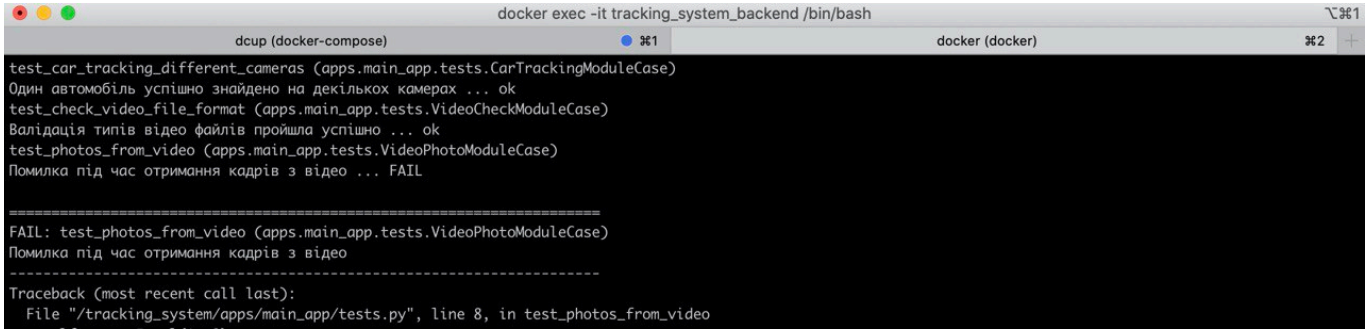

Рисунок 6.1 – Тест не пройдено

Після отримання негативного результату тесту було проведено налагодження модуля та оптимізація системи. Далі було зроблено регресійне тестування модуля системи.

Результат після налагодження: проведений повторно тест показав, що даний модуль системи функціонує коректно.

Тестовий випадок номер 3 - перевірка функціоналу трекінгу об’єкта. Однією з вимог до розроблюваного програмного забезпечення є розробка модуля трекінгу об’єкта. Наприклад, машина з номерним знаком AA0000TO була помічена на відео з камери 1 о 12:45:16, також машина з номером CA0000TE була помічена на цьому ж

відео о 12:47:09. На відео з камери 2 машина з номерним знаком АА0000ТО була помічена о 12:47:34. Система має ідентифікувати машину АА0000ТО на двох відео, та вивести інформацію про час зображення її на різних відео, та не включати машину зі схожим номером.

Вхідна інформація: два кадри з різних камер на яких зображений один і той самий об’єкт (машина АА0000ТО).

Вихідна інформація: номер авто АА0000ТО розпізнано та виділено на кадрах з обох камер, час перебування машини в кадрі виведено коректно.

Результат: тест пройдено.

6.2 Мануальне тестування

Тестування вручну, як правило, є частиною комплексного процесу тестування програмного забезпечення та проводиться до випуску програмного забезпечення. Тестери програмного забезпечення використовують та переглядають розроблене програмне забезпечення так, як ним користується кінцевий користувач. Процес тестування може бути систематичним процесом, який використовує формальні плани тестування та випадки дій.

Програмне забезпечення може бути протестовано з точки зору кількох ролей користувача, наприклад, адміністратора системи та користувача продукту. Проте ручне тестування передусім переглядає та тестує веб-сайт з точки зору кінцевого користувача, ручне тестування також може проводитися розробниками / тестерами програмного забезпечення, використовуючи свої знання та досвід для виявлення дефектів в межах програмного забезпечення.

6.2.1 Приклади тестів

Тестовий випадок 1 - використання системи гостем. У розробленій системі користувач має обов'язково зареєструватися, а потім авторизуватися.

Вхідна інформація: гість (неавторизований у системі користувач) намагається запустити алгоритм.

Вихідна інформація: система пренаправляє гостя на сторінку реєстрації чи авторизації (рисунок 6.2).

Результат: тест пройдено.

Тестовий випадок 2 - завантаження файлу недопустимого формату. У системі розпізнавання та трекінгу об’єктів користувач має змогу завантажити файли декількох форматів, а саме формати «.mp4» або «.mov.».

Вхідна інформація: користувач намагається завантажии файл формату .jpg.

Вихідна інформація: система виводить на інтерфейс помилку та інформацію про допустимі формати.

Результат: позитивний.

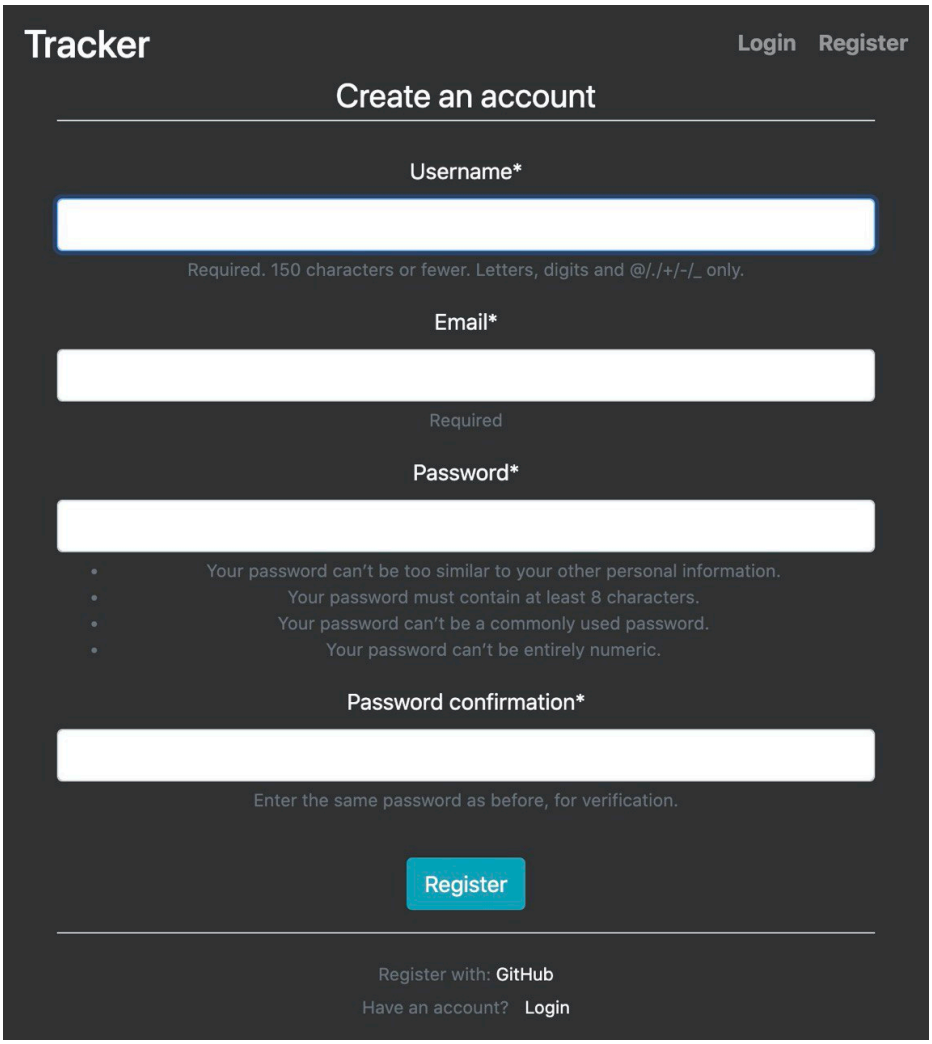

Рисунок 6.2 – Сторінка реєстрації

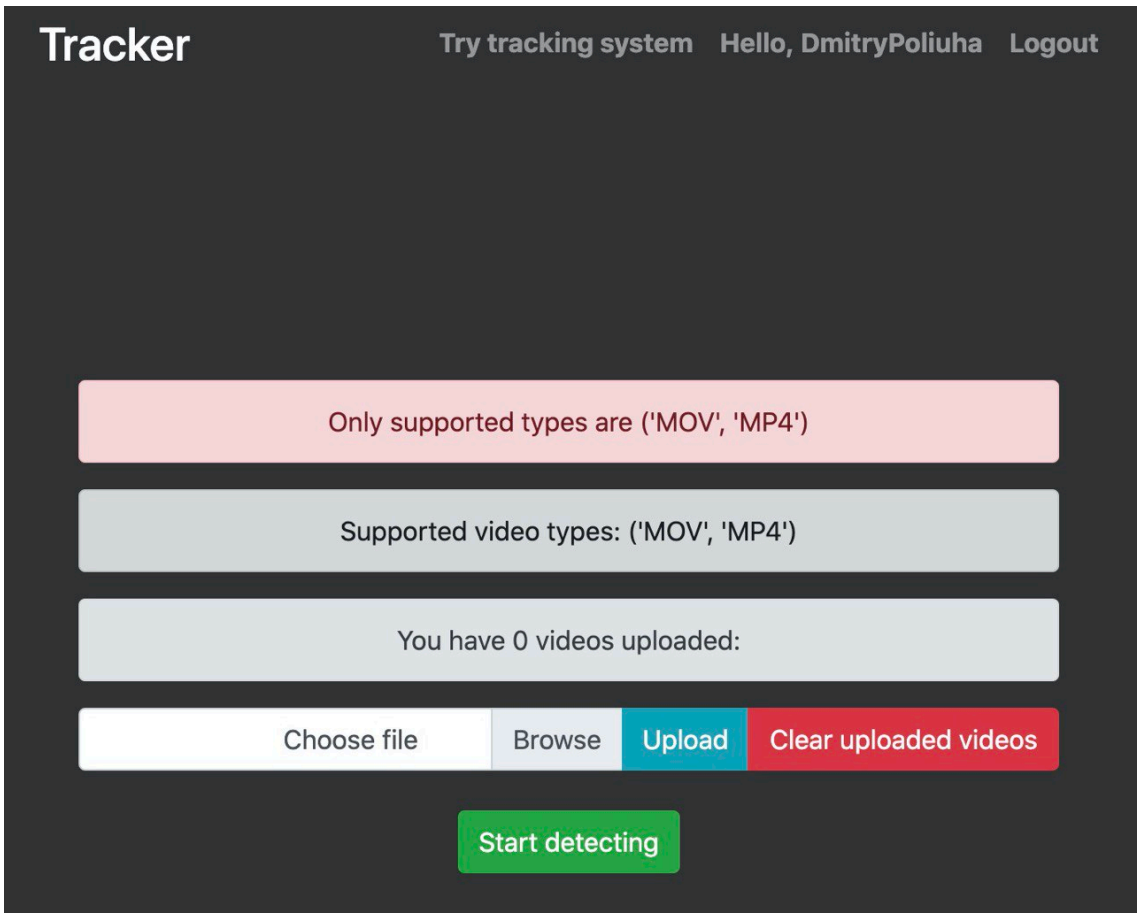

Рисунок 6.3 – Сповіщення про помилку

Висновки до розділу

Проведені тести, як юніт, так і мануальні показали, що розроблювана система працює коректно та відповідає вимогам якості, рис 6.4. За допомогою одного з юніт тестів було знайдено програмну помилку в системі, яка була виправлена.

Тестування допомогло виявити критичні помилки на ранніх етапах розробки, що призвело до правильного функціонування продукту. Майбутні користувачі можуть бути впевнені у якості.

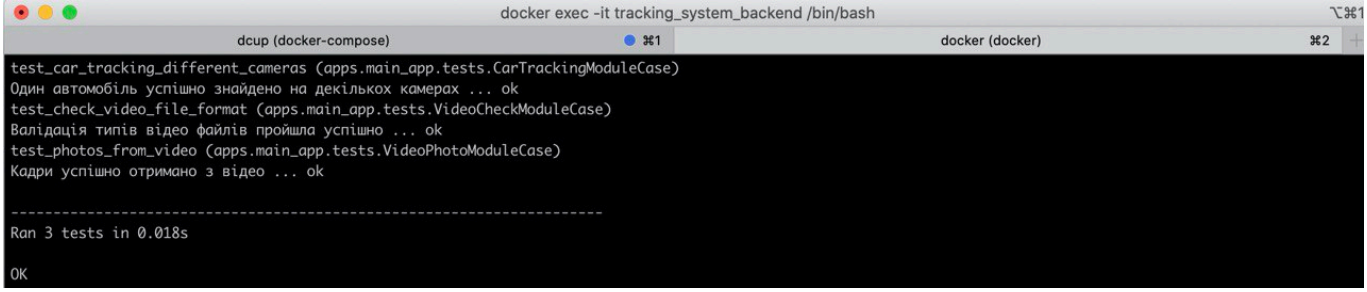

Рисунок 6.4 – Всі приведені вище тести виконано позитивно

ВИСНОВКИ

У результаті виконання дипломної роботи було розроблено систему розпізнавання та трекінгу об’єктів за допомогою камер відеоспостереження. Після вивчення предметної області та тестування декількох існуючих систем, об’єктом спостереження було обрано автомобіль. Спроектована система вирішує проблему ідентифікації та відстеження автомобілів за допомогою розпізнавання номерних знаків. Задача є актуальною, оскільки такі системи є необхідною частиною сучасного контролю трафіку, а існуючі програми не можуть задовольнити всі потреби користувачів.

В процесі дослідження було розглянуто існуючі системи, що використовують компоненти для розпізнавання тексту. Розроблено архітектуру модулів виходячи з аналізу проведених досліджень.

Основними перевагами спроектованої системи ідентифікації та трекінгу є:

a. Простота і зручність у використанні;

b. Практичність і функціональність при малій вартості апаратного забезпечення.

Було вирішено створити веб-систему з графічним користувацьким інтерфейсом для демонстрації розробленої системи ідентифікації номерних реєстраційних знаків авто та їх трекінгу. Розроблений продукт має наступні можливості:

- Завантаження необмеженої кількості відеопотоків;
- Розбиття відеопотоків на набір кадрів;
- Виділення областей з номерними знаками на фото;
- Розпізнавання номерних знаків на виділених областях;
- Аналіз опрацьованих даних з усіх відеопотоків;
- Вивід повних результатів роботи системи користувачам.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Kuo-Ming Hung, Ching-Tang Hsieh — A Real-Time Mobile Vehicle License Plate Detection and Recognition / Kuo-Ming Hung, Ching-Tang Hsieh // Tamkang Journal of Science and Engineering. — 2010.

2. Стругайло В.В. — Обзор методов фильтрации и сегментации цифровых изображений / В.В. Стругайло. — 2012.

3. Работа каскада Хаара в OpenCV: теория и практика [Електронний ресурс] — Режим доступу до ресурсу: https://habr.com/ru/company/recognitor/blog/228195/.

4. Shakhnarovish, Darrell, and Indyk — Nearest-Neighbour Methods in Learning and Vision / Shakhnarovish, Darrell, and Indyk // MIT Press. — 2005.

5. Арлазаров В.Л., Троянкер В.В., Котович Н.В. — Адаптивное распознавание Символов [Електронний ресурс] / В.Л. Арлазаров, В.В. Троянкер, Н.В. Котович. — 2000. — Режим доступу до ресурсу: http://ocrai.narod.ru/adaptive.html.

6. Котович Н.В., Славин О.А. — Распознавание скелетных образов [Електронний ресурс] / Н.В.Котович, О.А.Славин. — 2003. — Режим доступу до ресурсу: http://ocrai.narod.ru/skeletrecognize.html.

7. Ясницький Л. Н. — Введение в искусственный интеллект / Л. Н. Ясницький. — 2005.

8. Хайкин Саймон — Нейронные сети / Саймон Хайкин. — 2008.

9. Стадник А.В. — Использование искусственных нейронных сетей и вейвлет-анализа для повышения эффективности в задачах распознавания и классификации / А.В. Стадник // Ивановский Государственный университет — 2004.

10. Система распознавания автомобильных номеров – Автомаршал [Електронний ресурс] — Режим доступу до ресурсу: https://www.mallenom.ru/products/videokontrol-i-uchet-avtotransporta/avtomarshal.

11. SecurOS Auto [Електронний ресурс] — Режим доступу до ресурсу: https://www.syssoft.ru/Intelligent-Security-Systems/SecurOS-Auto/.

12. Система распознавания на базе решения ANPR Access [Електронний ресурс] — Режим доступу до ресурсу: http://www.tadviser.ru/index.php/%D0%9F%D1%80%D0%BE%D0%B4%D1%83%D0%BA%D1%82:ANPR_Access.

13. Распознавание автомобильных номеров — обзор 16 производителей [Електронний ресурс] — Режим доступу до ресурсу: http://securityrussia.com/blog/avto-nomera.html.

14. Распознавание номеров: от A до 9 [Електронний ресурс] — Режим доступу до ресурсу: https://habr.com/ru/company/recognitor/blog/221891/.

ДОДАТОК А

Система ідентифікації та трекінгу об’єктів за допомогою камер відеоспостереження

Специфікація

УКР.НТУУ «КПІ». ТІ6293_20Б

Аркушів 2

Київ 2020

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ«КПІ»_ТЕФ_АПЕПС_ТІ6293_20Б	ТІ62_Полюга_Записка.doc	Текстова частина дипломної роботи
Компоненти		
УКР.НТУУ«КПІ»_ТЕФ_АПЕПС_ТІ6293_20Б	main_app/views.py	Основний алгоритм
УКР.НТУУ«КПІ»_ТЕФ_АПЕПС_ТІ6293_20Б	settings.py	Налаштування системи
УКР.НТУУ«КПІ»_ТЕФ_АПЕПС_ТІ6293_20Б	main_app/base.html	Базовий шаблон додатку
УКР.НТУУ«КПІ»_ТЕФ_АПЕПС_ТІ6293_20Б	docker-compose.yaml	Конфігурація docker-compose
УКР.НТУУ«КПІ»_ТЕФ_АПЕПС_ТІ6293_20Б	Dockerfile	Конфігурація Docker
УКР.НТУУ«КПІ»_ТЕФ_АПЕПС_ТІ6293_20Б	accounts/views.py	Алгоритм роботи системи автентифікації
УКР.НТУУ«КПІ»_ТЕФ_АПЕПС_ТІ6293_20Б	accounts/urls.py	Конфігурація url для системи автентифікації

ДОДАТОК Б

Система ідентифікації та трекінгу об’єктів за допомогою камер відеоспостереження

Текст програми

УКР.НТУУ «КПІ». ТІ6293_20Б

Аркушів 17

Київ 2020

main_app/views.py

```
from collections import defaultdict
from dataclasses import dataclass

from django.shortcuts import render
from django.core.files.storage import FileSystemStorage
from django.http import JsonResponse
from django.contrib.auth.decorators import login_required
import logging
import json
import requests
import cv2
import time
import numpy as np
import os
from os.path import isfile, join
from difflib import SequenceMatcher
from dateutil import parser
import datetime
import subprocess
from collections import Counter

from PIL import Image, ImageFont, ImageDraw, ImageEnhance

regions = ['ua']  # Change to your country


@dataclass
class Result:
    id: int
    path: str
    processed_path: str
    tracking_plates_imgs_data: list
    camera_results: list


def about_view(request):
    context = {}
    return render(request, "main_app/about.html", context)


@login_required(login_url='/accounts/register/')
def index_view(request):
    user_videos_file = f"media/{request.user.username}.json"
    user_videos = {"videos": []}
    if isfile(user_videos_file):
        with open(user_videos_file) as f:
            user_videos = json.load(f)

    img_file = request.FILES.get('video_file')
```

```
    context = {}
    supported_types = ("MOV", "MP4")
    context["supported_types"] = supported_types
    context["uploaded_videos"] = len(user_videos["videos"])
    context["uploaded_videos_paths"] = user_videos["videos"]
    if request.method == "POST" and img_file:
        if img_file.name[-3:].upper() not in supported_types:
            context["err"] = f"Only supported types are {supported_types}"
            return render(request, "main_app/index.html", context)
        fs = FileSystemStorage()
        filename = fs.save(img_file.name, img_file)
        uploaded_file_url = fs.url(filename)
        uploaded_path = uploaded_file_url
        uploaded_path = uploaded_path[1:]
        user_videos["videos"].append(uploaded_path)
        context["uploaded_videos"] += 1
    with open(user_videos_file, 'w') as fp:
        json.dump(user_videos, fp)
    return render(request, "main_app/index.html", context)


@login_required(login_url='/accounts/register/')
def results_view(request):
    user_results_file = f"media/processed-test.json"
    # user_results_file = f"media/{request.user.username}-processed.json"
    results_file = {}
    if isfile(user_results_file):
        with open(user_results_file) as f:
            results_file = json.load(f)

    tracking_plates_image_data_raw = results_file.get("tracking_plates_image_data_raw", [])
    processed_paths = results_file.get("processed_paths", [])
    uploaded_videos_paths = results_file.get("uploaded_videos_paths", [])
    tracking_plates_list = []
    for camera_data in tracking_plates_image_data_raw:
        for found_frame in camera_data:
            tracking_plates_list += found_frame[0]
    tracking_plates_counter = Counter(tracking_plates_list)
    deleted_plates_mapping = {}
    for plate, count in tracking_plates_counter.items():
        comparison = []
        for plate_comp in tracking_plates_counter.keys():
            if len(plate) > 3 and len(plate_comp) > 3 and (plate != plate_comp):
                match = SequenceMatcher(None, plate, plate_comp).ratio()
                if plate in plate_comp:
                    comparison.append(plate_comp)
                elif match >= 0.8:
                    comparison.append(plate_comp)
        if comparison:
            max_count = tracking_plates_counter[plate]
            max_plate = plate
            for comp_plate in comparison:
```

```
            comp_plate_count = tracking_plates_counter[comp_plate]
            if comp_plate_count >= max_count or len(comp_plate) > len(max_plate):
                max_count = comp_plate_count
                max_plate = comp_plate
        if plate != max_plate:
            deleted_plates_mapping[plate] = max_plate
            original_plate_count = tracking_plates_counter[plate]
            tracking_plates_counter[plate] = 0
            tracking_plates_counter[max_plate] += original_plate_count

for cam_idx, cam_data in enumerate(tracking_plates_image_data_raw):
    for frame_idx, frame_data in enumerate(cam_data):
        for plate_idx, plate in enumerate(frame_data[0]):
            if plate in deleted_plates_mapping.keys():
                frame_data[0][plate_idx] = deleted_plates_mapping[plate]

tracking_plates_lists = []
for camera_data in tracking_plates_image_data_raw:
    tracking_plates_camera_list = []
    for found_frame in camera_data:
        tracking_plates_camera_list += found_frame[0]
    tracking_plates_lists.append(tracking_plates_camera_list)

cross_cameras_found_mapping = defaultdict(set)
for i in range(len(tracking_plates_lists) - 1):
    for j in range(i + 1, len(tracking_plates_lists)):
        for camera in tracking_plates_lists[i]:
            if camera in tracking_plates_lists[j]:
                cross_cameras_found_mapping[camera].update([i, j])
cross_cameras_found = []
for plate, cameras in cross_cameras_found_mapping.items():
    cross_cameras_found.append(f"Plate {plate} found on cameras {list(cameras)}")

cameras_plates_timecodes_list_of_dicts = []
for cam_idx, cam_data in enumerate(tracking_plates_image_data_raw):
    plates_timecodes = defaultdict(list)
    for frame_idx, frame_data in enumerate(cam_data):
        for plate_idx, plate in enumerate(frame_data[0]):
            plates_timecodes[plate].append(frame_data[2])
    cameras_plates_timecodes_list_of_dicts.append(plates_timecodes)
cameras_plates_timecodes = []
for camera in cameras_plates_timecodes_list_of_dicts:
    plates_timecodes = []
    for plate, timecodes in camera.items():
        # TODO if car twice on one camera
        plates_timecodes.append([plate, timecodes, max(timecodes) - min(timecodes)])
    cameras_plates_timecodes.append(plates_timecodes)

total_results = []
for cam_idx, cam_data in enumerate(tracking_plates_image_data_raw):
    total_results.append(
        Result(
```

```
                id=cam_idx,
                processed_path=processed_paths[cam_idx],
                path=uploaded_videos_paths[cam_idx],
                camera_results=cameras_plates_timecodes[cam_idx],
                tracking_plates_imgs_data=cam_data,
            )
        )

    context = {
        "results": total_results,
        "cross_cameras_found": cross_cameras_found
    }
    return render(request, "main_app/results.html", context)


def clear_videos_view(request):
    user_videos_file = f"media/{request.user.username}.json"
    if isfile(user_videos_file):
        with open(user_videos_file, "w") as f:
            json.dump({"videos": []}, f)
    return JsonResponse({})


def tracking_process_view(request):
    import time
    time.sleep(5)
    return JsonResponse({})
    user_videos_file = f"media/{request.user.username}.json"
    user_videos = {"videos": []}
    if isfile(user_videos_file):
        with open(user_videos_file) as f:
            user_videos = json.load(f)
    uploaded_paths = user_videos["videos"]
    processed_paths = []
    logging.warning(uploaded_paths)
    tracking_plates_image_data_raw = []

    try:
        for idx, uploaded_path in enumerate(uploaded_paths):
            tracking_plates_data_camera = []
            iphone = True if uploaded_path[-3:].upper() == "MOV" else False
            frameRate = 0.5  # //it will capture image in each 0.5 second
            # if iphone:
            #     scale = "1024:576"
            # else:
            #     scale = "576:1024"
            # output = subprocess.check_output(
            #     f"ffmpeg -y -noautorotate -i {uploaded_path} -f mp4 -vcodec libx264 -preset fast -profile:v
main -acodec aac -vf scale={scale} media/ua-auto-vid0-compressed.mp4",
            #     shell=True,
            #     stderr=subprocess.STDOUT
            # )
```

```
# output = str(output)
# start = output.index("creation_time") + len("creation_time")
# creation_time = output[start:start+24]
# creation_time = creation_time.lstrip(" :")
#
# try:
#     creation_time = parser.parse(creation_time)
# except Exception:
#     creation_time = datetime.datetime.now()
creation_time = datetime.datetime.now()

# vidcap = cv2.VideoCapture('media/ua-auto-vid0-compressed.mp4')
vidcap = cv2.VideoCapture(uploaded_path)
width = int(vidcap.get(3))  # float
height = int(vidcap.get(4))
if iphone:
    size = (height, width)
else:
    size = (width, height)
out_name = f'media/video{idx}.avi'
processed_paths.append(out_name)
out = cv2.VideoWriter(out_name, cv2.VideoWriter_fourcc(*'DIVX'), 2, size)

def getFrame(sec):
    vidcap.set(cv2.CAP_PROP_POS_MSEC, sec * 1000)
    hasFrames, image = vidcap.read()
    plate_numbers_on_frame = []

    if hasFrames:

        uploaded_path = "media/videoframes/temp.jpg"
        if iphone:
            rotated = cv2.transpose(image)
            image = cv2.flip(rotated, flipCode=1)
        cv2.imwrite(uploaded_path, image)
        with open(uploaded_path, 'rb') as fp:
            results = PlatePredictor(fp)
        source_img_t = Image.open(uploaded_path)
        source_img = Image.open(uploaded_path)
        plates_on_img = []
        img_plates_list = []
        for idx_plate, result_plate in enumerate(results):
            plate_img_name = f"plate-img-cam-{idx}-sec-{sec}-plate-{idx_plate}.jpg"
            plate_number = result_plate.get("plate").upper()
            box = result_plate.get("box")
            x_min = box['xmin']
            y_min = box['ymin']
            x_max = box['xmax']
            y_max = box['ymax']
            im_crop = source_img_t.crop((x_min, y_min, x_max, y_max))
            im_crop.save('media/plates/' + plate_img_name, "JPEG")
            font_size = y_max - y_min - 2
```

```
                draw = ImageDraw.Draw(source_img)
                draw.rectangle(((x_min, y_min), (x_max, y_max)), fill="black")
                plate_numbers_on_frame.append(plate_number)
                draw.text((x_min, y_min), plate_number, font=ImageFont.truetype("media/arial.ttf",
font_size))
                plates_on_img.append(plate_number)
                img_plates_list.append(plate_img_name)
            if plates_on_img:
                raw_img_name = f"raw-img-cam-{idx}-sec-{sec}-plate.jpg"
                source_img_t.save('media/plates/' + raw_img_name, "JPEG")
                img_plates_list = [raw_img_name] + img_plates_list
                tracking_plates_data_camera.append([plates_on_img, img_plates_list, sec])

            source_img.save('media/videoframes/temp-out.jpg', "JPEG")

            out.write(cv2.imread('media/videoframes/temp-out.jpg'))
        return hasFrames, plate_numbers_on_frame

    sec = 0
    count = 1
    success, plate_numbers = getFrame(sec)
    while success:
        count = count + 1
        sec = sec + frameRate
        sec = round(sec, 2)
        success, plate_numbers = getFrame(sec)
    out.release()
    tracking_plates_image_data_raw.append(tracking_plates_data_camera)

  user_videos_file = f"media/{request.user.username}.json"
  user_videos = {"videos": []}
  if isfile(user_videos_file):
    with open(user_videos_file) as f:
      user_videos = json.load(f)
  context = {
    "tracking_plates_image_data_raw": tracking_plates_image_data_raw,
    "processed_paths": processed_paths,
    "uploaded_videos_paths": user_videos["videos"],
  }
  user_results_file = f"media/{request.user.username}-processed.json"
  with open(user_results_file, 'w') as fp:
    json.dump(context, fp)
 except Exception as e:
  context = {"err": str(e)}
 return JsonResponse(context)
```

settings.py

```
"""
Django settings for tracking_system project.
```

```
Generated by 'django-admin startproject' using Django 3.0.6.

For more information on this file, see
https://docs.djangoproject.com/en/3.0/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/3.0/ref/settings/
"""

import logging.config
import os
from pathlib import Path
from django.utils.log import DEFAULT_LOGGING

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = os.getenv("SECRET_KEY")

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = [
    "0.0.0.0",
    "127.0.0.1",
    "localhost",
]


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    "crispy_forms",
    "social_django",

    "apps.main_app.apps.MainAppConfig",
    "apps.accounts.apps.AccountConfig"
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
```

```
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
    "social_django.middleware.SocialAuthExceptionMiddleware",
]

ROOT_URLCONF = 'tracking_system.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        "DIRS": [os.path.join(BASE_DIR, "tracking_system/templates")],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',

                'social_django.context_processors.backends',  # <--
                'social_django.context_processors.login_redirect', # <--
            ],
        },
    },
]

WSGI_APPLICATION = 'tracking_system.wsgi.application'


# Database
# https://docs.djangoproject.com/en/3.0/ref/settings/#databases

DATABASES = {
    "default": {
        "ENGINE": "django.db.backends.postgresql_psycopg2",
        "NAME": "tracking_system",
        "USER": "postgres",
        "PASSWORD": "postgres",
        "HOST": "db",
        "PORT": 5432,
    }
}


AUTHENTICATION_BACKENDS = (
    'social_core.backends.github.GithubOAuth2',
    'social_core.backends.facebook.FacebookOAuth2',
```

```
    'django.contrib.auth.backends.ModelBackend',
)

# Password validation
# https://docs.djangoproject.com/en/3.0/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },

    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },

    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },

    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/3.0/topics/i18n/

LANGUAGE_CODE = 'en'

TIME_ZONE = "Europe/Kiev"

USE_I18N = True

USE_L10N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.0/howto/static-files/

STATIC_URL = "/static/"
STATIC_ROOT = os.path.join(BASE_DIR, "static")

STATICFILES_DIRS = [
    os.path.join(BASE_DIR, "tracking_system/static"),
]
CRISPY_TEMPLATE_PACK = "bootstrap4"

MEDIA_URL = "/media/"
MEDIA_ROOT = os.path.join(BASE_DIR, "media")
```

```
# SMTP
EMAIL_HOST = 'smtp'  # Mailhog Host
EMAIL_PORT = '1025'

# redirects
LOGIN_REDIRECT_URL = "main_app:index"
LOGOUT_REDIRECT_URL = "main_app:index"
LOGIN_URL = "login"
LOGOUT_URL = "logout"

SOCIAL_AUTH_POSTGRES_JSONFIELD = True
SOCIAL_AUTH_GITHUB_KEY = "50df6eef351d78a60a50"
SOCIAL_AUTH_GITHUB_SECRET = "2dddbfdeb58593e3334ab5a3ac52e6ae654f37b0"

# Logging
GLOBAL_LOGLEVEL = os.environ.get("GLOBAL_LOGLEVEL", "info").upper()
CONSOLE_LOGLEVEL = os.environ.get("CONSOLE_LOGLEVEL", "info").upper()
FILE_LOGLEVEL = os.environ.get("FILE_LOGLEVEL", "debug").upper()
LOGGING_CONFIG = None
LOGGING_PATH = Path("logs")
LOGGING_PATH.mkdir(parents=True, exist_ok=True)
logging.config.dictConfig(
    {
        "version": 1,
        "disable_existing_loggers": False,
        "formatters": {
            "simple": {"format": "%(asctime)s %(name)-12s %(levelname)-8s %(message)s",},
            "django.server": DEFAULT_LOGGING["formatters"]["django.server"],
        },
        "handlers": {
            "console": {"level": CONSOLE_LOGLEVEL, "class": "logging.StreamHandler", "formatter":
"simple",},
            "file": {
                "level": FILE_LOGLEVEL,
                "class": "logging.handlers.TimedRotatingFileHandler",
                "backupCount": 10,
                "when": "midnight",
                "formatter": "simple",
                "filename": LOGGING_PATH / "debug.log",
            },
            "django.server": DEFAULT_LOGGING["handlers"]["django.server"],
        },
        "loggers": {
            "": {"level": GLOBAL_LOGLEVEL, "handlers": ["console", "file"],},

            "django.server": DEFAULT_LOGGING["handlers"]["django.server"],

        },
    }
)
```

main_app/base.html

```
{% load static %}
<!doctype html>
<html lang="en">
<head>
   <!-- Required meta tags -->
   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

   <link rel="shortcut icon" href="{% static "favicon.ico" %}" type="image/x-icon">

   <link rel="icon" href="{% static "favicon.ico" %}" type="image/x-icon">

   <link rel="stylesheet" type="text/css" href="{% static "css/main.css" %}">
   <!-- Bootstrap CSS -->
   <link rel="stylesheet" href="{% static "css/bootstrap.min.css" %}">

   <link rel="stylesheet" href="{% static "css/cover.css" %}">

   {% block stylesheets %}{% endblock %}

   <title>{% block title %}Tracking System{% endblock %}</title>
</head>

<body class="text-center" data-gr-c-s-loaded="true">

   <div class="cover-container d-flex h-100 p-3 mx-auto flex-column">
    <header class="masthead mb-auto">

      <div class="inner">

        <a href="{% url "main_app:index" %}"><h3 class="masthead-brand">Tracker</h3></a>

       <nav class="nav nav-masthead justify-content-center">
              {% if user.is_authenticated %}
         <a class="nav-link" href="{% url "main_app:try" %}">Try tracking system</a>
              {% endif %}
        {% if user.is_authenticated %}
         <a class="nav-link" href="{% url "profile" %}">Hello, {{ user }}</a>
         <a class="nav-link" href="{% url "logout" %}">Logout</a>
        {% else %}
          <a class="nav-link" href="{% url "login" %}">Login</a>
          <a class="nav-link" href="{% url "register" %}">Register</a>
        {% endif %}
       </nav>
      </div>
    </header>

    <main role="main" class="inner cover">
```

```
    {% if messages %}
            {%  for message in messages %}
               <div class="alert alert-{{ message.tags }}">{{ message }}</div>
            {% endfor %}
          {% endif %}
                {% block content %}{% endblock %}

      <p class="lead">
      </p>
    </main>

    <footer class="mastfoot mt-auto">
      <div class="inner">
{#        <p>Tracker, by <a href="{% url "main_app:index" %}">Dmitry Poliuha</a></p>#}
      </div>
    </footer>
  </div>


   <!-- Bootstrap core JavaScript
   ================================================== -->
   <!-- Placed at the end of the document so the pages load faster -->

{#    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-
KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
crossorigin="anonymous"></script>#}
{#    <script src="../../assets/js/vendor/popper.min.js"></script>#}
{#    <script src="../../dist/js/bootstrap.min.js"></script>#}


   <!-- jQuery first, then Popper.js, then Bootstrap JS -->
   <script src="{% static "js/jquery-3.4.1.min.js" %}"></script>
   <script src="{% static "js/popper.min.js" %}"></script>
   <script src="{% static "js/bootstrap.min.js" %}"></script>

   <script src="{% static "main_app/js/main_script.js" %}"></script>
   <script src="{% static "main_app/js/loader.js" %}"></script>

<script type="text/javascript">
  //init: automatic monitoring ajax events
  var loading = $.loading();

  //enable and disable listening ajax events
  loading.ajax(true);//enable
  </script>
    {% block script %}{% endblock %}

</body>
</html>
```

docker-compose.yml

```
version: "3.3"

services:
  db:
    container_name: tracking_system_db_postgres
    image: postgres:latest
    volumes:
      - postgres_data:/tracking_system/data
    environment:
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=postgres
      - POSTGRES_DB=tracking_system
    ports:
      - 5432:5432
    networks:
      - backend_network

  backend_wsgi:
    restart: always
    container_name: tracking_system_backend
    build: .
    command: gunicorn tracking_system.wsgi:application --bind=:8000 --workers=2 --reload --timeout
600
    env_file:
      - .env
    volumes:
      - .:/tracking_system
    ports:
      - 8000:8000
    depends_on:
      - db
      - smtp
    networks:
      - backend_network

  smtp:
    image: mailhog/mailhog:latest
    container_name: tracking_system_mailhog
    expose:
      - 25
      - 8025
    ports:
      - 8025:8025
    networks:
      - backend_network

volumes:
  postgres_data:

networks:
```

```
  backend_network:
    driver: bridge
```

Dockerfile

```
FROM python:3.8-slim-buster

ARG PROJECT_ROOT=/tracking_system

EXPOSE 8000

ENV PYTHONDONTWRITEBYTECODE 1
ENV PYTHONPATH $PYTHONPATH:$PROJECT_ROOT

WORKDIR $PROJECT_ROOT

RUN apt-get update \
  && apt-get install -y --no-install-recommends gcc libc-dev libsm6 libxext6 libxrender-dev libgtk2.0-
dev\
  && rm -rf /var/lib/apt/lists/* \
  && pip install --no-cache-dir psycopg2-binary regex numpy Pillow \
  && apt-get purge -y --auto-remove gcc libc-dev

RUN apt update && apt install -y ffmpeg

COPY requirements.txt $PROJECT_ROOT/requirements.txt
RUN pip install --no-cache-dir --upgrade pip && pip install --no-cache-dir -r requirements.txt

COPY tracking_system $PROJECT_ROOT
```

accounts/views.py

```
from django.contrib import messages
from django.contrib.auth.decorators import login_required
from django.shortcuts import redirect, render

from .forms import SignupForm, UserUpdateForm


@login_required
def profile_view(request):
  if request.method == "POST":
    u_form = UserUpdateForm(request.POST, instance=request.user)

    if u_form.is_valid():

      u_form.save()

      messages.success(request, f"Your account has been updated!")
```

```
            return redirect("profile")

    else:
        u_form = UserUpdateForm(instance=request.user)
    context = {"u_form": u_form}
    return render(request, "accounts/profile.html", context)


def register_view(request):
    if request.user.is_authenticated:
        return redirect("main_app:index")
    if request.method == "POST":
        form = SignupForm(request.POST)
        if form.is_valid():
            user = form.save()
            messages.success(
                request, f"{user}, you have been successfully registered",
            )
        return redirect("main_app:index")
    else:
        form = SignupForm()
    return render(request, "accounts/register.html", {"form": form})
```

accounts/urls.py

```
from django.contrib.auth import views as auth_views
from django.urls import path

from . import views

urlpatterns = [

    path("register/", views.register_view, name="register"),

    path("profile/", views.profile_view, name="profile"),

    path(
        "login/",
        auth_views.LoginView.as_view(template_name="accounts/login.html",
redirect_authenticated_user=True),
        name="login",
    ),
    path("logout/", auth_views.LogoutView.as_view(), name="logout"),
    path(
        "password_reset/",
        auth_views.PasswordResetView.as_view(template_name="accounts/password_reset.html"),
        name="password_reset",
    ),

    path(
```

```
        "password_reset/done/",

auth_views.PasswordResetDoneView.as_view(template_name="accounts/password_reset_done.html"),
        name="password_reset_done",
    ),

    path(
        "password_reset/confirm/<uidb64>/<token>/",

auth_views.PasswordResetConfirmView.as_view(template_name="accounts/password_reset_confirm.ht
ml",),
        name="password_reset_confirm",
    ),
    path(
        "password_reset/complete/",

auth_views.PasswordResetCompleteView.as_view(template_name="accounts/password_reset_complete.
html"),
        name="password_reset_complete",
    ),
]
```

ДОДАТОК В

Система ідентифікації та трекінгу об’єктів за допомогою камер відеоспостереження

Опис програми

УКР.НТУУ «КПІ». ТІ6293_20Б

Аркушів 10

Київ 2020

АНОТАЦІЯ

Додаток містить опис основних програмних компонентів системи ідентифікації та трекінгу об’єктів за допомогою камер відеоспостереження, а саме:

- Алгоритмів ідентифікації та трекінгу об’єктів;
- Основний шаблон системи;
- Конфігурація системи;
- Формування результатів та їх вивід користувачу;
- Алгоритм роботи системи автентифікації.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ

В даному додатку міститься опис основних програмних компонентів системи ідентифікації та трекінгу об’єктів за допомогою камер відеоспостереження. Програмний код відповідних компонентів міститься у додатку Б.

Веб-застосунок створено за допомогою мови програмування Python, фреймворку Django для створення серверної частини системи та мови програмування JavaScript для створення інтерфейсу користувача.

Для ізоляції програмного середовища використано технологію віртуалізації Docker.

Алгоритм ідентифікації та трекінгу створено за допомогою мови програмування та бібліотек NumPy та PyTorch.

Для зберігання даних використано систему управління базами даних PostgreSQL.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Система надає функціонал для ідентифікації та трекінгу об’єктів за допомогою камер відеоспостереження, а саме:

- Функціонал створення нового користувача, а також авторизації за допомогою GitHub;
- Графічний інтерфейс користувача для завантаження відеофайлів різних форматів;
- Завантаження опрацьованого результуючого відео на пристрій користувача для подальшого перегляду;
- Перегляд результатів ідентифікації об’єктів на кожному кадрі відеофайлу;
- Перегляд результатів ідентифікації та трекінгу об’єктів з усіх відеофайлів у вигляді таблиці;
- Зберігання результатів ідентифікації та трекінгу об’єктів в кабінеті користувача.

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Програмний комплекс складається з двох модулів згідно з архітектурою:

- Алгоритм ідентифікації та трекінгу об’єктів за допомогою камер відеоспостереження;
- Веб-додаток для демонстрації роботи алгоритму.

Веб-система складається з компонентів користувацького інтерфейсу та компонентів взаємодії користувацького інтерфейсу, конфігурації системи та алгоритму ідентифікації та трекінгу об’єктів.

ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Для використання веб-системи користувач повинен мати персональний комп’ютер з браузером, а також підключення до мережі інтернет.

Для запуску системи користувач повинен мати персональний комп’ютер зі встановленими Docker та Docker-compose.

ВИКЛИК І ЗАВАНТАЖЕННЯ

Система не потребує інсталяції завдяки тому, що було створено веб-додаток для демонстрації роботи алгоритму. Додаток доступний за наступним посиланням http://0.0.0.0:8000.

ВХІДНІ ДАНІ

Вхідна інформація веб-системи для інтерактивної роботи з 3D-об’єктами:

- Інформація про використовуваний апаратний пристрій – браузер користувача;
- Один, або декілька відеофайлів певного типу, що будуть завантажені в систему.

ВИХІДНІ ДАНІ

Вихідна інформація системи ідентифікації та трекінгу об’єктів за допомогою камер відеоспостереження:

- Інформація про розроблену систему;
- Кабінет користувача;
- Інформація про завантажені відеофайли;
- Результати ідентифікації об’єктів на кожному кадрі відеофайлу;
- Результатів ідентифікації та трекінгу об’єктів з усіх відеофайлів у вигляді таблиці;
- Зберігання результатів ідентифікації та трекінгу об’єктів в кабінеті.