МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інтелектуальних інформаційних систем

_____ Євген СІДЕНКО

«____»_____ 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

СИСТЕМА РОЗПІЗНАВАННЯ АВТОМОБІЛЬНИХ НОМЕРІВ «СВІЙ/ЧУЖИЙ»

Спеціальність 122 Комп'ютерні науки

Освітня програма «Комп'ютерні науки»

Здобувач _____Олег ЄМЕЛЬЯНОВ

«___»_____ 2026 р.

Керівник ст. викладачка _____Світлана БОРОВЛЬОВА

«___»_____ 2026 р.

Миколаїв – 2026

Бланк завдання на кваліфікаційну роботу

Чорноморський національний університет імені Петра Могили
(повне найменування закладу вищої освіти)

Факультет	Комп’ютерних наук
Кафедра	Інтелектуальних інформаційних систем
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступень	Бакалавр
Спеціальність	122 Комп’ютерні науки
Освітня програма	Комп’ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем

_____ Євген СІДЕНКО

«____» _____ 20___ р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувача

_____Ємельянова Олега Віталійовича_____
(прізвище, ім’я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Система розпізнавання автомобільних номерів «Свій/Чужий»

Керівник роботи: Боровльова Світлана Юріївна, старша викладачка
(прізвище, ім’я, по батькові, посада, науковий ступінь, вчене звання)

Затверджена наказом ЧНУ ім. Петра Могили від «25» 12_ 2026 р. № 353.

2. Строк представлення кваліфікаційної роботи «__» ___20__ р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні: розроблена вебплатформа відповідно до теми роботи.

Система автоматичного розпізнавання автомобільних номерів із функцією перевірки «Свій/Чужий», вебінтерфейс для адміністрування бази дозволених транспортних засобів та перегляду журналу подій доступу.

4. Перелік питань, що підлягають розробці:
– огляд предметної сфери контролю доступу транспортних засобів;
– огляд наявних аналогів систем розпізнавання автомобільних номерних знаків;
– аналіз методів та технологій для розробки системи; – вибір технічного та програмного забезпечення для системи розпізнавання автомобільних номерів;
– програмна реалізація системи розпізнавання автомобільних номерів «Свій/Чужий» на базі Django, Python та PostgreSQL .

5. Перелік графічних матеріалів: презентація

Здобувач _____ Олег ЄМЕЛЬЯНОВ
_____
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Керівник роботи _____ Світлана БОРОВЛЬОВА
_____
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата видачі завдання «21» грудня 2025 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема:_ «Система розпізнавання автомобільних номерів «Свій/Чужий»»

№	Найменування роботи	Початок	Закінчення	Примітки
1	Дослідження предметної сфери розпізнавання автомобільних номерів	14.10.2025	18.11.2025	Виконано
2	Огляд наявних аналогів систем розпізнавання номерних знаків	19.11.2025	27.11.2025	Виконано
3	Вибір технічного забезпечення	28.11.2025	04.12.2025	Виконано
4	Вибір програмного забезпечення (Python, Django, OpenCV, YOLO)	05.12.2025	09.12.2025	Виконано
5	Розробка та навчання нейромережі для розпізнавання номерних знаків	10.12.2025	19.12.2025	Виконано
6	Проєктування бази даних PostgreSQL та моделей Django	20.12.2025	26.12.2025	Виконано
7	Розробка серверної частини вебзастосунку (Django REST API)	27.12.2025	30.04.2025	Виконано
8	Інтеграція нейромережі з вебзастосунком та реалізація логіки «Свій/Чужий»	05.02.2026	26.02.2026	Виконано
9	Тестування системи та збір статистики	27.02.2026	28.03.2026	Виконано
10	Аналіз результатів та оптимізація точності розпізнавання	01.04.2026	06.04.2026	Виконано
11	Оформлення пояснювальної записки та підготовка до захисту	07.04.2026	02.05.2026	Виконано
12	Подання КР, її електронної копії та інших документів (відгуку, рецензії) до захисту	02.05.2026	24.05.2026	Виконано
13	Захист КР перед ЕК	22.06.2026	22.06.2026	Виконано

Розробив студент ___Ємельянов Олег Віталійович _____
(прізвище, ім'я, по батькові студента) (*підпис*)

Керівник роботи старша викладачка Боровльова Світлана Юріївна
(*ступень, звання, прізвище, ім'я, по батькові)*
_____
(підпис)

АНОТАЦІЯ

до кваліфікаційної роботи

студента 402 групи ЧНУ ім. Петра Могили

Ємельянова Олега Віталійовича

на тему: «Система розпізнавання автомобільних номерів «Свій/Чужий»»

Керівник: старша викладачка Боровльова Світлана Юріївна.

Об'єкт роботи – процес автоматизації контролю доступу транспортних засобів на закриту територію.

Предмет роботи – використання технологій комп'ютерного зору та нейронних мереж для автоматизації ідентифікації транспортних засобів за державним реєстраційним номером.

Мета – розробка програмного забезпечення для контролю доступу транспортних засобів шляхом розробки системи автоматичного розпізнавання автомобільних номерів із функцією перевірки «Свій/Чужий».

У роботі проведено аналіз предметної області контролю доступу та огляд існуючих аналогів, обґрунтовано вибір технологій, спроєктовано архітектуру системи. Здійснено програмну реалізацію на базі Django та PostgreSQL із модулем розпізнавання на основі YOLO та EasyOCR. Проведено тестування точності розпізнавання та логіки перевірки доступу.

Сторінок - 74, таблиць - 13, рисунків - 17, посилань - 32, додатків - 2.

Ключові слова: *розпізнавання номерних знаків, комп'ютерний зір, OpenCV, EasyOCR, Django, PostgreSQL, контроль доступу.*

ABSTRACT

for diploma work

of student of 402 group at Petro Mohyla Black Sea National University

Yemelianov Oleh Vitaliyovych

on the topic: «Vehicle License Plate Recognition System «Friend or Foe»»

Supervisor: senior teacher Borovlova Svitlana Yuriivna.

The object of work is the processes of automating access control of vehicles to closed territory. The subject of work is the use of computer vision and neural network technologies for automating vehicle identification by state registration plate number.

The goal is to develop software for vehicle access control by creating an automatic license plate recognition system with a "Friend or Foe" verification function.The work includes an analysis of the subject area and existing analogues, justification of the technology stack, system architecture. Software implementation was carried out using Django and PostgreSQL with a recognition module based on YOLO and EasyOCR. Testing of recognition accuracy and access verification logic was performed.

Pages - 74, tables - 13, figures - 17, references - 32, appendices - 2.

Keywords: *license plate recognition, computer vision, OpenCV, EasyOCR, Django, PostgreSQL, access control.*

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

КПП — контрольно-пропускний пункт

ПЗ — програмне забезпечення

ПК — персональний комп'ютер

СКБД — система керування базами даних

ALPR — Automatic License Plate Recognition, автоматичне розпізнавання номерних знаків

ANPR — Automatic Number Plate Recognition, автоматичне розпізнавання номерних знаків

API — Application Programming Interface, програмний інтерфейс застосунку

CNN — Convolutional Neural Network, згорткова нейронна мережа

CRAFT — Character Region Awareness for Text Detection, метод детекції тексту на рівні символів

CRNN — Convolutional Recurrent Neural Network, згортково-рекурентна нейронна мережа

CRUD — Create, Read, Update, Delete — основні операції роботи з даними

CSRF — Cross-Site Request Forgery, міжсайтова підробка запитів

CSS — Cascading Style Sheets, каскадні таблиці стилів

CTC — Connectionist Temporal Classification, метод декодування послідовностей

FPN — Feature Pyramid Network, мережа піраміди ознак

HTML — HyperText Markup Language, мова розмітки гіпертексту

HTTP/HTTPS — HyperText Transfer Protocol (Secure), протокол передачі гіпертексту (захищений)

JSON — JavaScript Object Notation, формат обміну даними

LSTM — Long Short-Term Memory, довга короткочасна пам'ять

MVT — Model-View-Template, архітектурний шаблон фреймворку Django

OCR — Optical Character Recognition, оптичне розпізнавання символів

ORM — Object-Relational Mapping, об'єктно-реляційне відображення

REST — Representational State Transfer, архітектурний стиль взаємодії компонентів

RFID — Radio-Frequency Identification, радіочастотна ідентифікація

SQL — Structured Query Language, мова структурованих запитів

UML — Unified Modeling Language, уніфікована мова моделювання

URL — Uniform Resource Locator, уніфікований покажчик ресурсу

XSS — Cross-Site Scripting, міжсайтовий скриптинг

YOLO — You Only Look Once, одноетапний детектор об'єктів

ВСТУП

Сучасні житлові комплекси, бізнес-центри та закриті паркінги щодня стикаються із необхідністю контролю доступу транспортних засобів на свою територію. Традиційні методи, такі як шлагбауми з пультами, картки доступу або ручна перевірка охоронцем, мають суттєві недоліки: низьку швидкість пропуску, залежність від людського фактору та можливість несанкціонованого проникнення.

Технології комп'ютерного зору та нейронних мереж дозволяють автоматизувати процес ідентифікації транспортних засобів за їхніми державними реєстраційними номерами. Автоматична система розпізнавання не потребує фізичних носіїв доступу, працює цілодобово без участі людини та забезпечує високу точність ідентифікації навіть за складних умов освітлення чи забруднення номерного знаку. Важливою перевагою є можливість ведення журналу всіх подій доступу, що підвищує загальний рівень безпеки об'єкта.

Мета кваліфікаційної роботи – розробка вебзастосунку для автоматизації доступу транспортних засобів на закриту територію шляхом розробки та реалізації системи автоматичного розпізнавання автомобільних номерів із функцією перевірки «Свій/Чужий».

Об'єкт кваліфікаційної роботи - процес контролю доступу транспортних засобів на закриту територію.

Предмет кваліфікаційної роботи - використання технологій комп'ютерного зору та нейронних мереж для автоматизації процесу ідентифікації транспортних засобів за державним реєстраційним номером.

Для досягнення мети необхідно розв'язати наступні задачі:

– аналіз предметної області систем розпізнавання автомобільних номерних знаків;

– вибір та навчання нейромережевої моделі для детекції та розпізнавання символів номерного знаку;

– проєктування та реалізація вебзастосунку на базі Django та PostgreSQL із логікою перевірки «Свій/Чужий»;

– тестування системи на реальних зображеннях та аналіз точності розпізнавання.

Для розв'язання задач використано мову програмування Python, фреймворк Django, СКБД PostgreSQL, бібліотеки OpenCV та EasyOCR для обробки зображень та оптичного розпізнавання символів, а також модель нейронної мережі YOLO для детекції області номерного знаку на зображенні.

Практична цінність отриманих результатів полягає у створенні доступної та гнучкої системи контролю доступу, яка не потребує дорогого спеціалізованого обладнання, працює з будь-якою камерою з роздільною здатністю від 1280×720 пікселів та повністю адаптована до українських державних реєстраційних номерів. Система має відкритий вихідний код, що дозволяє модифікувати та розширювати її функціональність відповідно до потреб конкретного об'єкта.

Декларація про використання ШІ. Під час підготовки наукової роботи (академічного тексту) було використано інструмент генеративного ШІ Claude Sonnet 4.6. Відповідно до таксономії GAIDeT (2025), наведені нижче завдання були делеговані інструментам генеративного ШІ за повного людського нагляду:

– Пошук і систематизація літератури
– Написання огляду літератури
– Вичитування та редагування
– Формулювання висновків

Повну відповідальність за фінальний рукопис несе автор

Інструменти генеративного ШІ не зазначаються як автори та не несуть відповідальності за кінцеві результати.

Декларацію подав: Ємельянов Олег

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

У даному розділі розглядається предметна область контролю доступу транспортних засобів на закриту територію. Проводиться аналіз існуючих методів ідентифікації автомобілів, описується процес діяльності та функціональна модель розроблюваної системи. Також здійснюється огляд наявних аналогів систем розпізнавання номерних знаків та формулюється постановка задачі з визначенням функціональних, програмних та технічних вимог до системи.

1.1 Опис предметної сфери

З організацією контролю доступу транспортних засобів на закриту територію пов'язано чимало практичних складнощів. Перш за все необхідно забезпечити надійну ідентифікацію кожного автомобіля, який прибуває до КПП, та прийняти рішення про надання або відмову у доступі за мінімальний проміжок часу.

Протягом розвитку інфраструктури безпеки було створено різноманітні методи контролю доступу транспорту, такі як:

– **особиста перевірка охоронцем -** співробітник на КПП візуально звіряє державний реєстраційний номер автомобіля зі списком дозволених транспортних засобів. Метод є найпростішим у впровадженні, але повністю залежить від уважності та кваліфікації персоналу;

– **системи на основі магнітних карток або брелоків доступу -** водій використовує фізичний носій (RFID-картку, магнітну картку або радіобрелок), який зчитується автоматичним пристроєм біля шлагбаума. Такий підхід широко використовується у житлових комплексах та паркінгах;

– **системи з дистанційним керуванням -** оператор спостерігає за камерою відеоспостереження та дистанційно відкриває шлагбаум, приймаючи рішення на основі візуальної оцінки;

– **автоматичні системи розпізнавання номерних знаків -** спеціалізовані програмно-апаратні комплекси, які за допомогою камер та

алгоритмів комп'ютерного зору автоматично зчитують державний номер і порівнюють його з базою даних дозволених транспортних засобів.

Найбільшого поширення на сьогоднішній день набули системи на основі карток та брелоків доступу. Таке широке розповсюдження ці системи здобули завдяки ряду переваг:

– **простота впровадження -** система потребує лише зчитувач та набір карток, не вимагає складного програмного забезпечення або потужного обладнання для обробки зображень;

– **автоматизація -** процес перевірки відбувається без участі людини — водій прикладає картку, система перевіряє її ідентифікатор і автоматично піднімає шлагбаум, що значно скорочує час пропуску;

– **надійність ідентифікації -** кожна картка має унікальний ідентифікатор, що виключає помилки, пов'язані з людським фактором при візуальному розпізнаванні номерів;

– **рівність умов -** усі користувачі проходять єдину процедуру ідентифікації за однаковими правилами, що забезпечує прозорість роботи системи.

Не зважаючи на значну кількість переваг, системи на основі фізичних носіїв мають суттєві вади, які спонукають до пошуку альтернативних рішень. Важливими недоліками даних систем є:

– **залежність від фізичного носія -** водій повинен завжди мати при собі картку або брелок. У разі втрати, пошкодження або забуття носія доступ стає неможливим, що створює незручності та потребує залучення додаткового персоналу для вирішення ситуації;

– **можливість несанкціонованого доступу -** фізичний носій може бути переданий іншій особі, загублений або скопійований. Система не має можливості перевірити, чи саме авторизований користувач використовує картку, що створює загрозу безпеці об'єкта;

– **витрати на обслуговування -** кожному новому користувачу необхідно виготовити персональну картку, а у разі втрати — **випустити дублікат та**

деактивувати попередню. Для великих об'єктів із сотнями користувачів ці витрати стають суттєвими;

– **відсутність візуального журналу -** традиційні карткові системи фіксують лише факт використання певного ідентифікатора, але не зберігають візуального підтвердження того, який саме автомобіль проїхав через контрольний пункт, що ускладнює розслідування інцидентів;

– **незручність у використанні -** водій повинен зупинити автомобіль, опустити вікно та піднести картку до зчитувача, що уповільнює процес проїзду, особливо в години пік або за несприятливих погодних умов;

– **неможливість ідентифікації незареєстрованого транспорту -** якщо автомобіль не має картки, система не може зафіксувати жодної інформації про нього, тоді як система на основі розпізнавання номерів здатна зберегти номер будь-якого транспортного засобу незалежно від його статусу в базі даних.

1.2 Опис процесу діяльності

Система розпізнавання автомобільних номерів «Свій/Чужий» — це програмний продукт, який дозволяє автоматично ідентифікувати транспортний засіб за його державним реєстраційним номером та прийняти рішення про надання або відмову у доступі на закриту територію на основі порівняння розпізнаного номера із записами у базі даних.

Система побудована за клієнт-серверною архітектурою. Серверна частина реалізована на базі вебфреймворку Django із використанням СКБД PostgreSQL для зберігання реєстру дозволених номерних знаків та журналу подій доступу. Модуль розпізнавання номерів використовує нейромережеву модель для детекції номерного знаку на зображенні та оптичного розпізнавання символів, що дозволяє перетворити візуальну інформацію у текстовий рядок номера.

Процес роботи системи побудований за наступним принципом. Камера, встановлена на КПП, фіксує зображення автомобіля, що наближається. Це зображення передається до серверної частини системи, де модуль розпізнавання

виконує послідовну обробку: спочатку на зображенні виявляється область номерного знаку, після чого виконується розпізнавання окремих символів. Отриманий текстовий рядок номера порівнюється із записами у базі даних PostgreSQL.

Якщо розпізнаний номер знайдено у базі даних дозволених транспортних засобів, система повертає результат «Доступ дозволено» та фіксує подію успішного проїзду у журналі. Якщо номер відсутній у базі даних, система повертає результат «Доступ заборонено» та також зберігає запис про спробу доступу з невідомим номером. Усі результати відображаються у вебінтерфейсі системи в режимі реального часу.

Використання системи автоматичного розпізнавання номерних знаків «Свій/Чужий» має свої переваги як для користувачів, так і для адміністраторів об'єкта:

Для користувачів (водіїв):

– відсутність необхідності мати при собі будь-які фізичні носії доступу - картки, брелоки чи пульти;

– швидкий проїзд через КПП без необхідності зупинятись та взаємодіяти зі зчитувачем або охоронцем.

Для адміністраторів об'єкта:

– повна автоматизація процесу перевірки без залучення додаткового персоналу на КПП;

– ведення детального журналу подій із зафіксованим зображенням, розпізнаним номером, датою та часом кожної спроби доступу;

– можливість оперативного додавання або видалення номерних знаків із бази дозволених транспортних засобів через вебінтерфейс;

– зниження витрат на виготовлення та обслуговування фізичних носіїв доступу;

– підвищення рівня безпеки завдяки фіксації усіх транспортних засобів, у тому числі тих, що не мають дозволу на проїзд.

1.3 Огляд наявних аналогів систем

Системи автоматичного розпізнавання номерних знаків ANPR почали активно розвиватися з середини 2000-х років разом із прогресом у сфері комп'ютерного зору та машинного навчання. Оскільки такі рішення потребують інтеграції апаратної та програмної складових, більшість продуктів на ринку розробляються спеціалізованими компаніями у сфері безпеки та відеоспостереження. Серед них можна виділити як комерційні програмно-апаратні комплекси, так і відкриті програмні рішення.

Одним із найвідоміших комерційних рішень є система Hikvision ANPR від китайського виробника обладнання для відеоспостереження Hikvision. Система являє собою програмно-апаратний комплекс, що включає спеціалізовані камери з інфрачервоним підсвічуванням та вбудованим модулем розпізнавання номерних знаків. Камери здатні розпізнавати номери транспортних засобів на швидкості до 120 км/год та передавати результати на сервер управління. Система підтримує інтеграцію з шлагбаумами та воротами, що дозволяє повністю автоматизувати КПП.

Рисунок 1.1 – Приклад системи розпізнавання автономерів від Hikvision

Загалом система отримує високі оцінки від користувачів завдяки надійності та точності розпізнавання. Проте серед недоліків відзначають високу вартість обладнання, закритість програмного забезпечення та складність налаштування для розпізнавання номерних знаків нестандартних форматів, зокрема українських.

Іншим відомим комерційним рішенням є Dahua ANPR, розроблений компанією Dahua Technology. Подібно до Hikvision, Dahua пропонує інтегрований комплекс із камер та програмного забезпечення для управління доступом. Система підтримує розпізнавання номерів у складних умовах освітлення завдяки алгоритмам адаптивної експозиції та вбудованому інфрачервоному підсвічуванню. Dahua також надає можливість формування чорних та білих списків номерів для автоматичного управління шлагбаумом.

Рисунок 1.2 – Камера Dahua

Переважна більшість користувачів високо оцінюють якість зображення та швидкість розпізнавання. Серед негативних аспектів відзначають обмежену кількість підтримуваних форматів номерних знаків, необхідність придбання ліцензій на додатковий функціонал та відсутність гнучкого API для інтеграції з власними програмними рішеннями.

Серед відкритих програмних рішень необхідно відзначити проект OpenALPR (нині Rekor Recognition Systems). OpenALPR — це бібліотека з відкритим вихідним кодом для розпізнавання номерних знаків, написана мовами C++ та Python. Бібліотека використовує алгоритми комп'ютерного зору на базі OpenCV для детекції області номерного знаку та механізм OCR (Tesseract) для розпізнавання символів. OpenALPR підтримує номерні знаки різних країн та може бути інтегрована у власні програмні продукти через API.

Рисунок 1.3 – Система розпізнавання номерів OpenALPR

Переважна більшість розробників позитивно оцінюють гнучкість та можливості налаштування бібліотеки. Серед недоліків відзначають зниження точності розпізнавання за поганих умов освітлення або при значному забрудненні номерного знаку, а також необхідність додаткового навчання моделі для коректної роботи з номерами окремих країн, у тому числі України.

Також слід згадати рішення PlateRecognizer — хмарний сервіс розпізнавання номерних знаків, який надає REST API для інтеграції у будь-який програмний продукт [18]. Розробник завантажує зображення через HTTP-запит та отримує у відповідь розпізнаний номер із вказанням рівня впевненості. PlateRecognizer підтримує номерні знаки понад 100 країн, включаючи Україну, та пропонує як хмарну, так і локальну версію для розгортання на власному сервері [18].

Сервіс забезпечує точність розпізнавання на рівні 97% для номерних знаків стандартного формату та надає додаткові можливості, такі як визначення типу транспортного засобу, його кольору та напрямку руху. PlateRecognizer також пропонує SDK для мов програмування Python, JavaScript та Java, що спрощує інтеграцію у існуючі програмні продукти.

Рисунок 1.4 – Розпізнавання номерів від PlateRecognizer

Більшість користувачів відзначають високу точність розпізнавання та зручність інтеграції через API. Серед негативних аспектів виділяють залежність хмарної версії від стабільного інтернет-з'єднання, обмеження безкоштовного тарифного плану та відсутність повного контролю над обробкою даних при використанні хмарного варіанту.

Проведений аналіз показує, що існуючі комерційні рішення здебільшого є дорогими програмно-апаратними комплексами із закритим програмним забезпеченням, а відкриті бібліотеки потребують значних зусиль з інтеграції та налаштування. Розробка власної системи на базі Django, PostgreSQL та сучасних нейромережевих моделей дозволяє створити доступне, гнучке та легко розширюване рішення, адаптоване саме до українських номерних знаків, із повним контролем над усіма компонентами системи.

Для наочного порівняння розглянутих аналогів було проведено порівняльний аналіз їх ключових переваг та недоліків (див. табл. 1.1).

Таблиця 1.1 – Порівняльний аналіз популярних систем розпізнавання

Аналог	**Переваги**	**Недоліки**
Hikvision ANPR	Висока точність розпізнавання, інтеграція з шлагбаумами, робота на швидкості до 120 км/год	Висока вартість, закрите ПЗ, обмежена підтримка українських номерів
Dahua ANPR	Адаптивна експозиція, ІЧ-підсвічування, формування чорних/білих списків	Висока вартість, потребує фірмових камер, платні ліцензії на додатковий функціонал
PlateRecognizer	Підтримка 100+ країн, зручний REST API, хмарна та локальна версії	Залежність від інтернету (хмарна версія), обмеження безкоштовного тарифу, відсутність повного контролю над даними
OpenALPR	Відкритий вихідний код, безкоштовність, інтеграція через API	Низька точність для українських номерів, відсутність веб-інтерфейсу, потребує додаткового навчання моделі

1.4 Постановка задачі

Загальні відомості

Повна назва - система розпізнавання автомобільних номерів «Свій/Чужий».

Передумови для проведення робіт

Житлові комплекси, бізнес-центри та закриті паркінги щодня стикаються з необхідністю контролю доступу транспортних засобів на свою територію. Традиційні методи контролю, такі як ручна перевірка охоронцем або використання карток доступу, мають суттєві обмеження у вигляді залежності від людського фактора, витрат на обслуговування фізичних носіїв та можливості

несанкціонованого проникнення.

Стрімке зростання кількості транспортних засобів в Україні та розбудова житлової інфраструктури створюють попит на доступні та надійні автоматизовані рішення контролю доступу. Розвиток технологій комп'ютерного зору та нейронних мереж дозволяє створити систему, яка автоматично розпізнає державний реєстраційний номер автомобіля та приймає рішення про допуск без участі людини.

Характеристика об’єкту проєктування

Дослідженню підлягає процес контролю доступу транспортних засобів на закриту територію шляхом автоматичного розпізнавання державних реєстраційних номерів.

Вимоги до функціональних характеристик

Програмний продукт повинен володіти наступними функціональними характеристиками:

– автоматичне розпізнавання державного реєстраційного номера автомобіля із завантаженого зображення;

– перевірка розпізнаного номера за базою даних дозволених транспортних засобів із видачею результату «Доступ дозволено» або «Доступ заборонено»;

– ведення журналу подій доступу із зазначенням розпізнаного номера, дати, часу та результату перевірки;

– вебінтерфейс для адміністрування бази дозволених номерних знаків (додавання, видалення, перегляд);

– відображення результатів розпізнавання у вебінтерфейсі.

Вимоги до програмного забезпечення

Серверна частина системи повинна бути реалізована на вебфреймворку Django мовою програмування Python. Для зберігання даних повинна використовуватись СКБД PostgreSQL. Модуль розпізнавання номерних знаків повинен використовувати бібліотеку OpenCV для попередньої обробки зображень та

нейромережеву модель для детекції та оптичного розпізнавання символів номерного знаку.

Система повинна функціонувати на операційних системах Windows 10/11 або Linux (Ubuntu 20.04 і вище). Для розгортання серверної частини необхідний інтерпретатор Python версії 3.10 або вище та встановлена СКБД PostgreSQL версії 14 або вище.

Вимоги до технічного забезпечення

Мінімальна апаратна конфігурація сервера, на якому може бути запущене середовище:

- процесор: Intel Core i3 10-го покоління / AMD Ryzen 3 3100 або кращий;
- оперативна пам'ять: від 4 ГБ;
- вільний простір на диску: від 10 ГБ;
- мережеве підключення: Ethernet 100 Мбіт/с або вище.

Додаткові технічні засоби:

- IP-камера або веб-камера з роздільною здатністю не менше 1280×720 пікселів для захоплення зображень номерних знаків;
- стабільне мережеве з'єднання між камерою та сервером.

Висновки до розділу 1

В даному розділі було охарактеризовано основні методи контролю доступу транспортних засобів на закриту територію, виділено їх головні переваги та недоліки. Особливу увагу приділено критичному аналізу систем, що базуються на радіочастотних мітках (RFID) та магнітних картках, у порівнянні з безконтактними оптичними методами фіксації. Також було описано процес діяльності та функціональну модель системи розпізнавання автомобільних номерів «Свій/Чужий», її клієнт-серверну архітектуру на базі Django та PostgreSQL, а також принцип автоматичної ідентифікації транспортного засобу за державним реєстраційним номером із використанням нейромережевої моделі. Обґрунтовано вибір паттерну MVC (Model-View-Controller) для побудови логіки вебзастосунку,

що забезпечує чітке розділення процесів обробки даних нейромережею, збереження інформації у реляційній базі даних та відображення динамічного контенту користувачу в режимі реального часу. Для пересвідчення доцільності розробки власного рішення було проведено огляд існуючих комерційних та відкритих аналогів систем розпізнавання номерних знаків, оцінено їх переваги та обмеження щодо вартості, гнучкості та адаптації до українських номерних знаків.

У процесі компаративного аналізу встановлено, що більшість закордонних open-source бібліотек мають високий рівень похибки при зчитуванні специфічних кириличних літерних серій вітчизняних стандартів ДСТУ. Наприкінці було розроблено технічні та функціональні вимоги до системи, які визначають мінімальну апаратну конфігурацію сервера, перелік необхідного програмного забезпечення та функціональні можливості, що повинен забезпечувати кінцевий програмний продукт.

Сформульовано вимоги до основних модулів системи: авторизації, завантаження зображень, розпізнавання номерних знаків, перевірки доступу, управління базою дозволених транспортних засобів та перегляду журналу подій. Додатково визначено критерії безпеки та збереження конфіденційності персональних даних власників автотранспорту відповідно до чинного законодавства.

Сформований комплекс вимог став надійним підґрунтям для подальшого математичного моделювання процесів комп'ютерного зору. Результати аналізу підтвердили, що інтеграція сучасних засобів глибокого навчання (Deep Learning) здатна нівелювати більшість виявлених деструктивних факторів навколишнього середовища.

2 ОГЛЯД ТЕХНОЛОГІЙ ТА СПЕЦИФІКАЦІЯ ВИМОГ

У даному розділі проводиться аналіз та обґрунтування вибору методів і технологій для розробки системи розпізнавання автомобільних номерів «Свій/Чужий». Розглядаються основні інструменти для реалізації серверної частини, обробки зображень та розпізнавання символів. Також розробляється детальна специфікація вимог до програмного забезпечення, що включає функціональні вимоги, характеристики користувачів, вимоги до інформаційного, технічного та програмного забезпечення системи.

2.1 Аналіз методів та технологій

Перед розробкою будь-якого програмного забезпечення спершу треба визначитись з інструментарієм, який буде використовуватись для створення застосунку, а саме які бібліотеки, фреймворки та інструменти, що надають можливість ефективно працювати із серверною частиною, обробляти зображення, зберігати дані та надавати користувачеві зручний вебінтерфейс, треба використовувати. Також при виборі потрібно враховувати специфікацію створюваного застосунку та функціонал, який буде реалізовуватись.

Для розробки серверної частини використовується фремворк Django, який є одним із найпопулярніших вебфреймворків мови програмування Python. Django побудований за архітектурним шаблоном MVT, який забезпечує чітке розділення логіки застосунку на три рівні - модель відповідає за роботу з даними, представлення (view) обробляє бізнес-логіку та запити користувача, а шаблон (template) формує HTML-сторінки для відображення у браузері. Django надає вбудовану адміністративну панель, систему автентифікації, ORM для роботи з базами даних та механізм маршрутизації URL-адрес, що значно прискорює процес розробки.

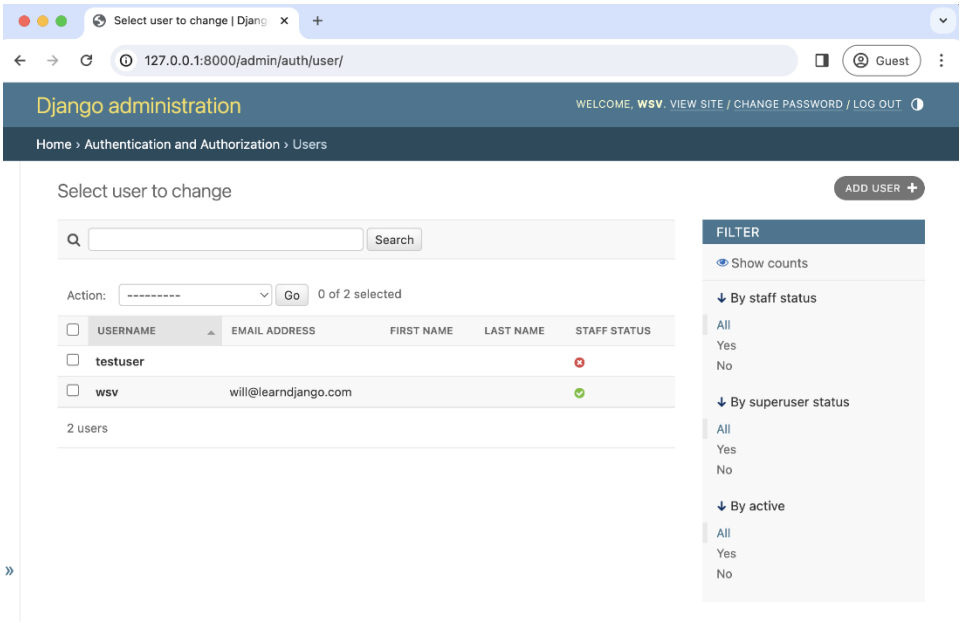

Рисунок 2.1 – Стандартна адміністративна панель у Django

PostgreSQL використано як основну систему управління базами даних у розроблюваній системі завдяки її високій продуктивності та надійності. PostgreSQL являє собою потужну реляційну СКБД з відкритим вихідним кодом, яка підтримує принципи ACID, що є гарантією цілісності даних навіть в умовах великих навантажень та паралельної роботи з великою кількістю транзакцій. PostgreSQL підтримує складні типи даних, повнотекстовий пошук та має широкі можливості індексування, що є важливим для швидкого пошуку номерних знаків у базі дозволених транспортних засобів.

Django ORM використано як інструмент об'єктно-реляційного відображення, що дозволяє працювати з базою даних PostgreSQL з використанням об'єктно-орієнтованого підходу мовою Python. Django ORM дозволяє описувати моделі таблиць бази даних у вигляді Python-класів, що забезпечує чітку структуру коду та захист від SQL-ін'єкцій. Використовуючи Django ORM можна реалізувати будь-які реляційні з'єднання між таблицями бази даних, такі як один до одного, один до багатьох або багато до багатьох. Також Django ORM автоматично створює та застосовує міграції бази даних, що спрощує процес розгортання та оновлення системи.

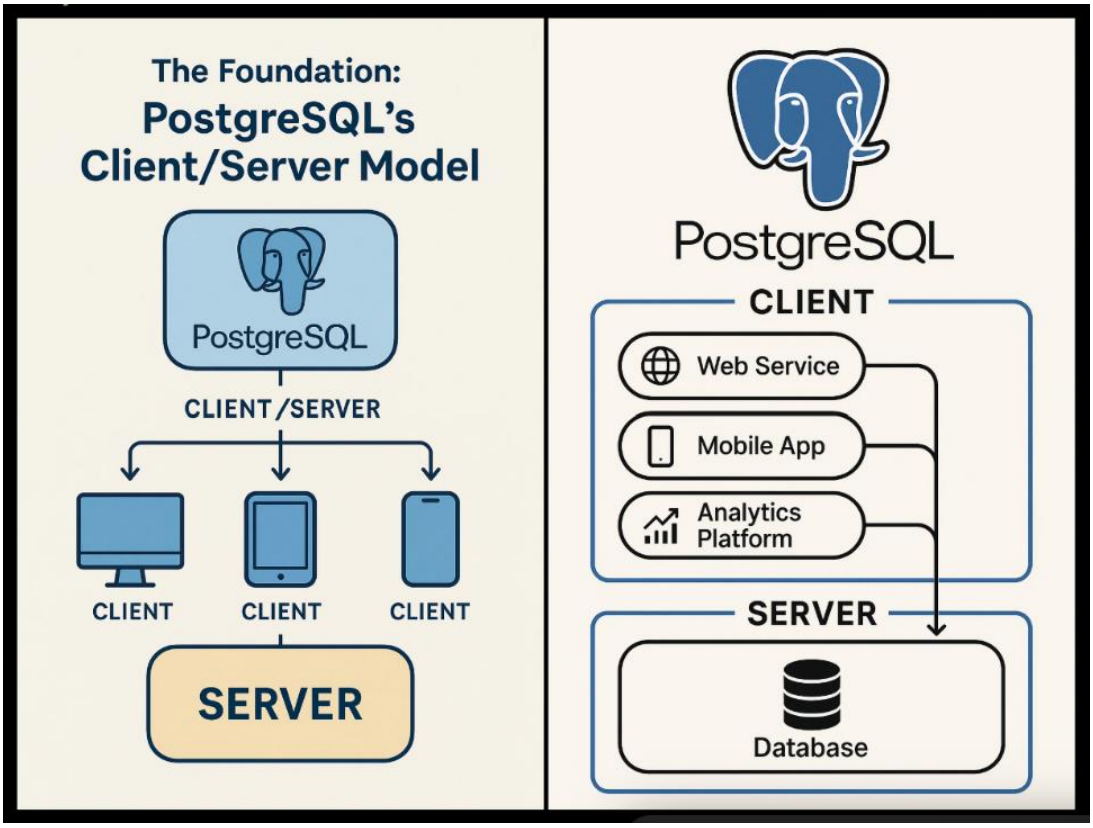

Рисунок 2.2 – Модель роботи PostgreSQL

Для реалізації модуля розпізнавання номерних знаків використано бібліотеку OpenCV — відкриту бібліотеку комп'ютерного зору, написану мовами C++ та Python. OpenCV надає широкий набір функцій для обробки зображень, включаючи перетворення кольорових просторів, фільтрацію, бінаризацію, виявлення контурів та геометричні трансформації. Бібліотека оптимізована для роботи в реальному часі та підтримує апаратне прискорення, що є важливим для швидкої обробки зображень з камери.

Для детекції області номерного знаку на зображенні використано нейромережеву модель. Модель приймає на вхід зображення автомобіля та повертає координати обмежувальної рамки навколо номерного знаку, що дозволяє вирізати саме ту частину зображення, яка містить номер, для подальшого розпізнавання символів.

Рисунок 2.3 – Приклад розпізнавання об’єктів із OpenCV

Для оптичного розпізнавання символів на виділеній області номерного знаку використано бібліотеку EasyOCR — інструмент для розпізнавання тексту на зображеннях з підтримкою понад 80 мов, включаючи українську та англійську [10]. EasyOCR побудований на базі глибоких нейронних мереж та забезпечує високу точність розпізнавання навіть при складних умовах зйомки, таких як нерівномірне освітлення, нахил номерного знаку або часткове забруднення. Внутрішня архітектура EasyOCR складається з двох ключових компонентів: детектора тексту CRAFT [4], який визначає розташування символів на зображенні, та розпізнавача CRNN [5], який перетворює виявлені символи у текстовий рядок. Бібліотека має відкритий вихідний код, не потребує комерційної ліцензії та легко інтегрується у Python-проєкти через менеджер пакетів pip.

Для детекції області номерного знаку на зображенні використано модель нейронної мережі сімейства YOLO [3], яка належить до класу одноетапних детекторів об'єктів. На відміну від двоетапних архітектур, YOLO формулює задачу детекції як єдину задачу регресії та виконує прогнозування координат обмежувальних рамок за один прохід нейромережі, що забезпечує високу швидкість обробки зображень у реальному часі. Модель було попередньо навчено на наборі даних автомобільних номерних знаків та збережено у форматі .pt для подальшого завантаження через фреймворк Ultralytics [11].

Для створення шаблонів вебінтерфейсу використано вбудований механізм шаблонів Django Templates у поєднанні з фреймворком Tailwind CSS [15], який забезпечує адаптивний дизайн та сучасний зовнішній вигляд сторінок без необхідності написання складного CSS-коду. Tailwind CSS використовує утилітарний підхід до стилізації, де кожен клас відповідає за одну CSS-властивість, що дозволяє швидко створювати складні інтерфейси безпосередньо у HTML-розмітці шаблонів.

2.2 Специфікація вимог до програмного забезпечення

2.2.1 Призначення та межі проєкту

Система, що розробляється призначена для автоматичного розпізнавання державних реєстраційних номерів автомобілів із зображень та прийняття рішення про надання або відмову у доступі на закриту територію шляхом порівняння розпізнаного номера із базою даних дозволених транспортних засобів.

2.2.2 Погодження, що ухвалені в програмній документації

Визначено перелік основних функцій, які будуть реалізовані на першому етапі розробки.

Використання сучасних вебтехнологій для забезпечення кросплатформеної роботи.

Система працює із зображеннями у форматах JPEG та PNG.

2.2.3 Межі проєкту ПЗ

Проєкт включає в себе розробку серверної частини на Django із вебінтерфейсом та модулем розпізнавання номерних знаків.

Система орієнтована на розпізнавання українських державних реєстраційних номерів.

Не включається розробка апаратної частини (камери, шлагбауми), але передбачається можливість інтеграції з ними через API.

2.2.4 Загальний опис

Систему можна застосовувати у різних галузях: у житлових комплексах для контролю доступу мешканців на територію, у бізнес-центрах для управління паркуванням співробітників та відвідувачів, на закритих паркінгах для автоматизації в'їзду та виїзду, а також на підприємствах для обмеження доступу стороннього транспорту на виробничу територію.

2.2.5 Характеристики користувачів

Адміністратор - це уповноважена особа, яка керує базою дозволених транспортних засобів. Адміністратор має змогу додавати нові номерні знаки до бази, видаляти існуючі записи, переглядати журнал подій доступу та здійснювати пошук за номером або датою.

Оператор - це користувач системи, який завантажує зображення автомобіля для розпізнавання номерного знаку та отримує результат перевірки доступу. У перспективі роль оператора може бути замінена автоматичною подачею зображень із камери відеоспостереження.

Рівнем підготовки адміністратора є базові навички системного адміністрування, оператора – базові навички роботи з вебзастосунками.

2.2.6 Загальна структура і склад системи

Серверна частина побудована на основі фреймворку Django мовою Python, з базою даних PostgreSQL. Модуль розпізнавання - використано бібліотеки OpenCV та EasyOCR для обробки зображень та розпізнавання символів. Вебінтерфейс - побудовано за допомогою Django Templates та Tailwind.

Загальні обмеження

Робота системи потребує стабільного мережевого з'єднання між камерою та сервером. Система орієнтована на розпізнавання номерних знаків українського зразка. Точність розпізнавання залежить від якості вхідного зображення та умов зйомки.

Функції системи

Дозволяє адміністраторам та операторам входити до системи за допомогою логіну та паролю для доступу до функціоналу управління.

Вхідна і вихідна інформація:

– вхідна: логін, пароль;

– вихідна: сесія користувача, перенаправлення на головну сторінку системи.

Функціональні вимоги:

– шифрування паролів за допомогою вбудованого механізму хешування Django (PBKDF2);

– валідація введених даних;

– обмеження кількості невдалих спроб входу.

2.2.7 Завантаження зображення для розпізнавання

Дозволяє оператору завантажити зображення автомобіля через вебінтерфейс для автоматичного розпізнавання номерного знаку.

Вхідна і вихідна інформація:

– вхідна: файл зображення номерного знаку автомобіля;

– вихідна: розпізнаний номерний знак, результат перевірки доступу («Доступ дозволено» або «Доступ заборонено»), збережене зображення.

Функціональні вимоги:

– перевірка формату та розміру завантаженого файлу;

– автоматичний запуск модуля розпізнавання після завантаження;

– збереження зображення на сервері для журналу подій.

Зміна паролю

Система автоматично обробляє завантажене зображення, виявляє область номерного знаку та розпізнає текст номера.

Функціональні вимоги:

– попередня обробка зображення;

– детекція області номерного знаку;

– розпізнавання символів за допомогою EasyOCR;

– нормалізація розпізнаного тексту.

2.2.8 Перевірка доступу

Система порівнює розпізнаний номер із базою дозволених транспортних засобів та формує результат («Доступ дозволено» або «Доступ заборонено»).

Вхідна і вихідна інформація:

– вхідна - розпізнаний номерний знак;

– вихідна - результат перевірки, запис у журналі подій.

Функціональні вимоги:

– пошук номера у базі даних;

– формування запису у журналі подій;

– відображення результату у вебінтерфейсі.

2.2.9 Управління базою транспортних засобів

Адміністратор (оператор) системи додає, редагує та видаляє записи дозволених номерних знаків.

Вхідна і вихідна інформація:

– вхідна: номерний знак, ім'я власника, опис;

– вихідна: підтвердження операції.

Функціональні вимоги:

– валідація формату номерного знаку;

– перевірка на унікальність;

– пошук та фільтрація записів.

2.2.10 Перегляд журналу подій

Адміністратор переглядає журнал подій доступу з можливістю фільтрації.

Вхідна і вихідна інформація:

– вхідна: параметри фільтрації (дата, номер, результат);

– вихідна: список подій із зображеннями.

Функціональні вимоги:

– хронологічне відображення подій;

– фільтрація та пошук;

– перегляд зображення для кожної події.

2.2.11 Зміна паролю

Дозволяє змінити пароль облікового запису.

Вхідна і вихідна інформація:

– вхідна: старий пароль, новий пароль;

– вихідна: підтвердження зміни.

Функціональні вимоги:

– перевірка старого паролю;

– мінімальна довжина нового паролю – 8 символів;

– хешування перед збереженням.

2.2.12 Вимоги до інформаційного забезпечення

Джерела і зміст вхідної інформації:

– зображення автомобілів із камери або завантажені вручну;

– дані про дозволені транспортні засоби (номер, власник, опис);

– облікові дані адміністраторів та операторів.

Нормативно-довідкова інформація

Формати українських державних реєстраційних номерів.

Класифікатор результатів перевірки доступу.

Вимоги до способів організації, збереження та ведення інформації

Зберігання у реляційній базі даних PostgreSQL.

Зберігання зображень у файловій системі сервера з посиланнями у базі даних.

Захищене зберігання паролів за допомогою вбудованого механізму хешування Django.

2.2.13 Вимоги до технічного забезпечення

Сервер із процесором Intel Core i3 або кращим та оперативною пам'яттю від 4 ГБ.

IP-камера або веб-камера з роздільною здатністю не менше 1280×720.

Стабільне мережеве з'єднання між камерою та сервером.

Клієнтські пристрої: комп'ютери з сучасним веб-браузером.

2.2.14 Вимоги до програмного забезпечення

Архітектура програмної системи

Монолітна архітектура на базі фреймворку Django з чітким розділенням на модулі.

Взаємодія між клієнтом і сервером через стандартні HTTP-запити.

Системне ПЗ

ОС сервера: Linux (Ubuntu 20.04 або новіше) або Windows 10/11.

Інтерпретатор Python версії 3.10 або вище.

Мережне ПЗ

Протокол HTTPS із сертифікатом SSL/TLS для захисту даних.

Підтримка стандартних HTTP-методів (GET, POST).

ПЗ ведення інформаційної бази

СКБД: PostgreSQL версії 14 або вище.

Інструменти для резервного копіювання: pg_dump.

Мова і технологія розробки

Backend: Python, Django, Django ORM.

Модуль розпізнавання: OpenCV, EasyOCR.

Frontend: Django Templates, Tailwind CSS.

База даних: PostgreSQL.

інструменти: Git для контролю версій, pip для управління залежностями.

2.2.15 Вимоги до зовнішніх інтерфейсів

Інтерфейс користувача

Адаптивний дизайн на базі Tailwind, сумісний із роздільною здатністю від 1024×768 до 2560×1440.

Інтуїтивно зрозумілий інтерфейс із відображенням результатів розпізнавання у реальному часі.

Апаратний інтерфейс

IP-камера або веб-камера для захоплення зображень автомобілів.

Стандартний ПЗ для доступу до вебінтерфейсу.

Програмний інтерфейс

Внутрішній API модуля розпізнавання для обробки зображень.

Django Admin API для управління даними через адміністративну панель.

Комунікаційний протокол

HTTPS для всіх запитів між клієнтом та сервером.

2.2.16 Властивості програмного забезпечення

Доступність

Система повинна бути доступною цілодобово для забезпечення безперервного контролю доступу транспортних засобів.

Супроводжуваність

Модульна структура коду для спрощення оновлень та додавання нового функціоналу.

Документація коду та API.

Переносимість

Можливість розгортання на будь-якому сервері з підтримкою Python та PostgreSQL, включаючи хмарні платформи.

Продуктивність

Час розпізнавання одного номерного знаку – не більше 3 секунд.

Одночасна підтримка до 10 операторів у вебінтерфейсі.

Надійність

Регулярне резервне копіювання бази даних.

Збереження зображень навіть у разі помилки розпізнавання для можливості ручної перевірки.

Безпека

Захист від XSS, CSRF та SQL-ін'єкцій засобами Django.

Хешування паролів користувачів.

Обмеження доступу до функцій управління лише для авторизованих користувачів.

2.2.17 Інші вимоги

Можливість розширення системи для підтримки номерних знаків інших

країн.

Можливість інтеграції з апаратними засобами контролю доступу (шлагбауми, ворота) через API.

Ведення статистики розпізнавання для аналізу точності роботи системи.

Висновки до розділу 2

В даному розділі було проведено аналіз та обґрунтування вибору технологій для розробки системи розпізнавання автомобільних номерів «Свій/Чужий», а саме фреймворку Django, СКБД PostgreSQL, бібліотек OpenCV та EasyOCR для обробки зображень та розпізнавання символів.

Також було розроблено детальну специфікацію вимог до програмного забезпечення, що включає опис призначення системи, характеристики користувачів, функціональні вимоги до кожного модуля, вимоги до інформаційного, технічного та програмного забезпечення, а також вимоги до зовнішніх інтерфейсів та властивостей системи.

Визначено архітектуру системи як монолітний Django-застосунок із модульною структурою, описано ролі користувачів, сформульовано вимоги до продуктивності, безпеки та надійності.

Розроблена специфікація є необхідною основою для подальшого етапу проєктування структури бази даних, моделювання архітектури програмного забезпечення та безпосередньої програмної реалізації системи розпізнавання автомобільних номерів «Свій/Чужий».

3 СТРУКТУРА СИСТЕМИ

У даному розділі описується структура розробленої системи розпізнавання автомобільних номерів «Свій/Чужий». Розглядається проєктування бази даних PostgreSQL, архітектура програмного забезпечення та алгоритм функціонування модулів системи, включаючи модуль комп'ютерного зору на базі YOLO та EasyOCR. Також проводиться обґрунтування вибору нейромережевих архітектур для задач детекції та розпізнавання номерних знаків та виконується об'єктно-орієнтоване моделювання системи із використанням мови UML.

3.1 Архітектура програмного забезпечення

Розроблене ПЗ системи розпізнавання автомобільних номерів «Свій/Чужий» має модульну архітектуру, побудовану на основі розподілу обов’язків між серверною частиною, клієнтським інтерфейсом та ізольованим модулем комп’ютерного зору й машинного навчання.

Основою серверної архітектури є вебфреймворк Django з використанням розширення Django REST Framework, що дозволяє реалізувати архітектурний стиль REST API для гнучкої взаємодії між компонентами. Клієнтська частина надсилає HTTP-запити до ендпоінтів сервера, який здійснює автентифікацію, валідацію даних, взаємодію з базою даних PostgreSQL та керування потоком комп’ютерного зору.

3.1.1 Взаємодія об’єктів на рівні ORM та бізнес-логіки

Для забезпечення автоматизації процесів перевірки статусу транспортного засобу («свій» чи «чужий») на рівні бізнес-логіки вебфреймворку Django застосовано об’єктно-реляційне відображення та вбудований механізм сигналів. Завдяки цьому архітектура системи позбавлена жорсткої зв'язності компонентів, а перевірка прав доступу відбувається автоматично під час транзакцій з базою даних.

Основним тригером для запуску бізнес-логіки розпізнавання є збереження нового об'єкта у таблиці сканувань. При кожній фіксації автомобільного номера

модулем комп'ютерного зору та спробі запису даних у модель PlateScan, автоматично генерується сигнал post_save.

Цей сигнал викликає внутрішній сервісний метод check_access(), яка виконує таку послідовність операцій:

– зчитує новостворений об'єкт сканування та виділяє з нього текстовий рядок розпізнаного державного номера;

– виконує запит до бази даних через ORM Django, здійснюючи пошук точного співпадіння (SQL-еквівалент оператора SELECT ... WHERE) у таблиці зареєстрованих номерів AutoNumbers;

– якщо ідентичний номер знайдено в базі даних і його поточний статус є активним (дозвіл не протерміновано), система динамічно змінює атрибут доступу транзакції, встановлюючи прапор дозволу на проїзд (access_allowed = True);

– якщо номер відсутній у базі даних або його прапор доступу деактивовано адміністратором, об'єкту сканування присвоюється статус «Чужий» або «Заблокований» (access_allowed = False). У цей же момент система ініціює алгоритм генерації сповіщення для оператора КПП.

Валідація вхідних даних та взаємодія клієнтської частини з серверною логікою реалізована за допомогою стандартного механізму представлень (Django Views) та форм (Django Forms). Під час завантаження кадру або потокового зображення з камери, відповідне представлення обробляє HTTP POST-запит, перевіряє коректність отриманого файлу через request.FILES та передає його обробнику комп'ютерного зору. Після завершення обробки та відпрацювання ORM-сигналів, Django View повертає оновлені дані на інтерфейс користувача у вигляді згенерованого HTML-шаблону (з використанням вбудованого двигуна шаблонів Django Template) або через об'єкт JsonResponse для динамічного оновлення елементів сторінки без її повного перезавантаження.

3.1.2 Алгоритм функціонування модуля комп'ютерного зору

Ключем елементом системи є потоковий або кадрований аналіз графічної інформації, що надходить із засобів фіксації (відеокамер). Процес обробки кожного кадру та детекції номерного знака складається з кількох послідовних етапів, алгоритм яких наведено нижче:

Етап 1. Попередня обробка зображення.

Вхідний кольоровий кадр зчитується за допомогою бібліотеки OpenCV і конвертується з простору BGR у простір відтінків сірого за формулою:

$$Y = 0.299 * R + 0.587 * G + 0.114 * B. \quad (3.1)$$

де Y– значення яскравості пікселя;

R– значення червоного каналу;

G – значення зеленого каналу;

B – значення синього каналу зображення.

Для усунення високочастотних шумів застосовується гаусове розмиття з ядром фільтра (5 на 5)

Етап 2. Локалізація та детекція номерного знака. Оброблений кадр передається на вхід нейромережевої моделі YOLO, яка була попередньо навчена на наборі даних автомобільних номерних знаків. Мережа повертає координати обмежувального рамкового контейнера у форматі крайніх точок та рівень впевненості моделі. Якщо коефіцієнт впевненості перевищує встановлений поріг, виконується операція кадрування - вирізання знайденої області номерного знака з оригінального зображення для подальшого аналізу.

Етап 3. Сегментація та оптичне розпізнавання символів. Вирізана область номерного знака піддається додатковому бінарному аналізу за методом Отсу для максимального підвищення контрасту між символами та фоном пластини. Очищена область передається в OCR-модуль. Модуль використовує глибоку нейронну мережу, що складається з детектору тексту CRAFT та розпізнавальної архітектури CRNN, яка поєднує згорткові шари для виділення ознак, рекурентні

шари LSTM для аналізу послідовностей та шар декодування CTC. На виході формується текстовий рядок розпізнаних символів.

Етап 4. Пост-обробка тексту та нормалізація. Отриманий текст очищується від випадкових спецсимволів, розділових знаків та пробілів за допомогою регулярних виразів. Здійснюється перевірка на відповідність шаблонам українських автомобільних номерів (наприклад, поєднання двох літер регіону, чотирьох цифр та двох літер серії типу AA1234BB). Також на цьому етапі реалізовано евристичний механізм заміни схожих за накресленням символів кирилиці та латиниці (наприклад, заміна латинської літери O на цифру 0, або кириличної В на латинську B, залежно від позиції символу в шаблоні відповідно до стандарту державного номера).

3.2. Обґрунтування вибору та аналіз архітектур нейронних мереж

Автоматичне розпізнавання автомобільних номерних знаків є комплексною двоетапною задачею комп'ютерного зору, яка вимагає послідовного застосування різних типів архітектур глибоких нейронних мереж. Перший етап полягає у просторовій локалізації об'єкта (номерної пластини) на зображенні довільного масштабу, другий — у розпізнаванні послідовності символів на вирізаній області. Для побудови високоефективної системи, що здатна працювати в умовах реального часу на обмежених обчислювальних потужностях, було обрано комбінацію моделі детекції YOLO та спеціалізованого OCR-модуля EasyOCR.

3.2.1 Архітектурні особливості нейромережі YOLO для детекції об'єктів

Для детекції зони розташування номерного знака в системі використано архітектуру сімейства YOLO. Вибір цієї моделі зумовлений її фундаментальною відмінністю від класичних двоетапних детекторів (на зразок Faster R-CNN), які спочатку генерують потенційні регіони інтересу, а потім класифікують їх. YOLO формулює задачу детекції як єдину задачу регресії, де координати обмежувальної

рамки та ймовірності класів прогнозуються безпосередньо з повного кадру за один прохід нейромережі (single forward pass).

Архітектурно нейромережа складається з трьох ключових компонентів:

– backbone (Магістральна мережа) - використовує модифіковану згорткову архітектуру для екстракції просторових ознак із вхідного зображення. Вона поступово зменшує роздільну здатність кадру, одночасно збільшуючи глибину каналів ознак, що дозволяє фіксувати як дрібні текстурні елементи (межі номера), так і глобальні контекстуальні зв'язки (контури автомобіля);

– neck (Шийка мережі) - призначена для об'єднання ознак, отриманих з різних шарів магістральної мережі. Тут застосовуються структури типу FPN або PANet. Вони забезпечують передачу низькорівневої геометрії на верхні шари та високорівневої семантики на нижні, завдяки чому мережа однаково точно детектує номерні знаки як на автомобілях, що під'їхали впритул до камери, так і на тих, що перебувають на відстані;

– head (Голова мережі) - відповідає за фінальне прогнозування. Вона генерує тривимірні тензори, які містять координати центрів обмежувальних рамок, їхні ширину та висоту, а також коефіцієнт впевненості моделі в тому, що всередині рамки дійсно міститься номерний знак.

Використання YOLO у системі «Свій/Чужий» забезпечує стійкість до суворих умов експлуатації: зміна ракурсу зйомки, нахил автомобіля, а також робота в темну пору доби за наявності засвічування від фар головного світла.

3.2.2 Компоненти глибокого навчання в архітектурі EasyOCR

Після того як YOLO локалізує та виріже прямокутну область номерного знака, вона передається на розпізнавання тексту. Замість застарілих методів посимвольної сегментації, в системі застосовано сучасну бібліотеку EasyOCR, внутрішня архітектура якої базується на поєднанні двох потужних нейромережевих концепцій: детектора тексту CRAFT та розпізнавача CRNN.

Детектор тексту CRAFT. Цей модуль призначений для точного виявлення символьних областей на вирізаній пластині номера. На відміну від стандартних детекторів об'єктів, CRAFT фокусується на двох аспектних картах:

– region Score (Оцінка регіону) - показує ймовірність того, що конкретний піксель є центром певного символу;

– affinity Score (Оцінка спорідненості) - визначає ймовірність того, що два послідовні символи є частиною одного слова (або єдиного номерного знака).

Такий підхід дозволяє EasyOCR успішно локалізувати символи навіть на пошкоджених, деформованих або забруднених номерних пластинах, чітко відокремлюючи корисні літери та цифри від фонового бруду чи кріпильних елементів.

Розпізнавач CRNN. Це гібридна модель, яка безпосередньо трансформує графічну область з символами у текстовий рядок. Вона складається з трьох послідовних рівнів:

– згорткові шари - приймають бінаризоване зображення номерного знака та виділяють з нього послідовність карт ознак. Зображення розрізається на вертикальні смуги (слайси), кожна з яких описується вектором ознак;

– рекурентні шари - складаються з двонаправлених блоків довгострокової короткочасної пам'яті Вони аналізують отриману послідовність векторів ознак зліва направо та справа наліво. Це дозволяє моделі враховувати контекст: літери та цифри в автомобільних номерах зазвичай чергуються за певними стандартами, і LSTM-шари допомагають мережі «додумувати» розмиті чи частково перекриті символи на основі сусідніх;

– шар декодування CTC - скільки символи на номерній пластині можуть мати різну ширину та пробіли між собою, мережа під час сканування може продублювати один і той самий символ кілька разів. Шар CTC аналізує вихідні ймовірності рекурентного рівня, об'єднує однакові послідовні символи, видаляє спеціальні «порожні» символи та формує фінальний очищений текстовий рядок номера.

Завдяки такій трирівневій структурі, EasyOCR забезпечує високу гнучкість розпізнавання без необхідності ручного налаштування кроку сегментації літер, що суттєво підвищує загальну надійність системи контролю доступу.

3.3 Об'єктно-орієнтоване моделювання системи

Для формалізації вимог, візуалізації структури застосунку та опису динаміки взаємодії між компонентами системи розпізнавання автомобільних номерів «Свій/Чужий» застосовано методологію об'єктно-орієнтованого аналізу та моделювання за допомогою уніфікованої мови моделювання UML.

Моделювання дозволяє абстрагуватися від конкретної програмної реалізації на початкових етапах і створити чіткий каркас системи, який описує поведінку користувачів, життєвий цикл інформаційних об'єктів та послідовність виконання бізнес-логіки. В межах проєкту розроблено та детально проаналізовано три базові типи моделей: діаграму прецедентів діаграму послідовності та діаграму станів.

3.3.1 Сценарії взаємодії та діаграма прецедентів системи

Діаграма прецедентів визначає функціональні межі системи та описує взаємодію між зовнішніми суб'єктами (акторами) і внутрішніми сервісами (прецедентами). В розроблюваній системі виділено дві основні ролі користувачів, які мають чітко розмежовані права доступу та сценарії взаємодії із вебінтерфейсом застосунку:

– адміністратор системи - уповноважена особа, яка володіє повним набором прав для керування конфігурацією та інформаційним наповненням бази даних. До ключових прецедентів адміністратора належать: аутентифікація в системі, керування обліковими записами операторів, адміністрування білого списку дозволених номерів (додавання, редагування, видалення записів), створення та налаштування категорій транспортних засобів, перегляд розширеного журналу подій доступу та вивантаження статистичних звітів;

– оператор КПП - користувач із обмеженими правами, основним завданням якого є безпосередній контроль потоку транспортних засобів у реальному часі. До прецедентів оператора належать: авторизація, завантаження графічних або відеофайлів для розпізнавання, моніторинг результатів детекції нейромережею та візуальне підтвердження проїзду автомобіля у разі виникнення позаштатних ситуацій;

Крім людських факторів, у моделі виділено системного актора - Модуль комп'ютерного зору, який взаємодіє з прецедентами обробки медіафайлів та детекції обмежувальних рамок. Зв'язки між прецедентами розширено за допомогою базових відношень UML. Зокрема, прецедент «Завантажити файл для розпізнавання» містить обов'язкове відношення зв'язку «include» з прецедентом «Автоматична детекція області номера нейромережею YOLO», оскільки процес аналізу не може бути запущений без наявності вхідного графічного потоку. В свою чергу, прецедент «Фіксація події доступу» має умовне відношення розширення «extend» з прецедентом «Надсилання сповіщення про спробу несанкціонованого доступу», яке спрацьовує виключно у разі, якщо розпізнаний номер набуває статусу «Чужий».

Таким чином, система передбачає чітке розмежування відповідальності між ролями: адміністратор відповідає за конфігурацію та наповнення бази даних дозволених номерів, тоді як оператор зосереджений на безпосередній роботі із потоком транспортних засобів. Модуль комп'ютерного зору функціонує як автономний системний актор, який не потребує ручного втручання для виконання детекції та розпізнавання номерних знаків. Взаємодія між прецедентами побудована за принципом послідовного ланцюга: завантаження файлу обов'язково ініціює детекцію нейромережею, яка в свою чергу автоматично фіксує подію доступу у журналі (див. рис. 3.1).

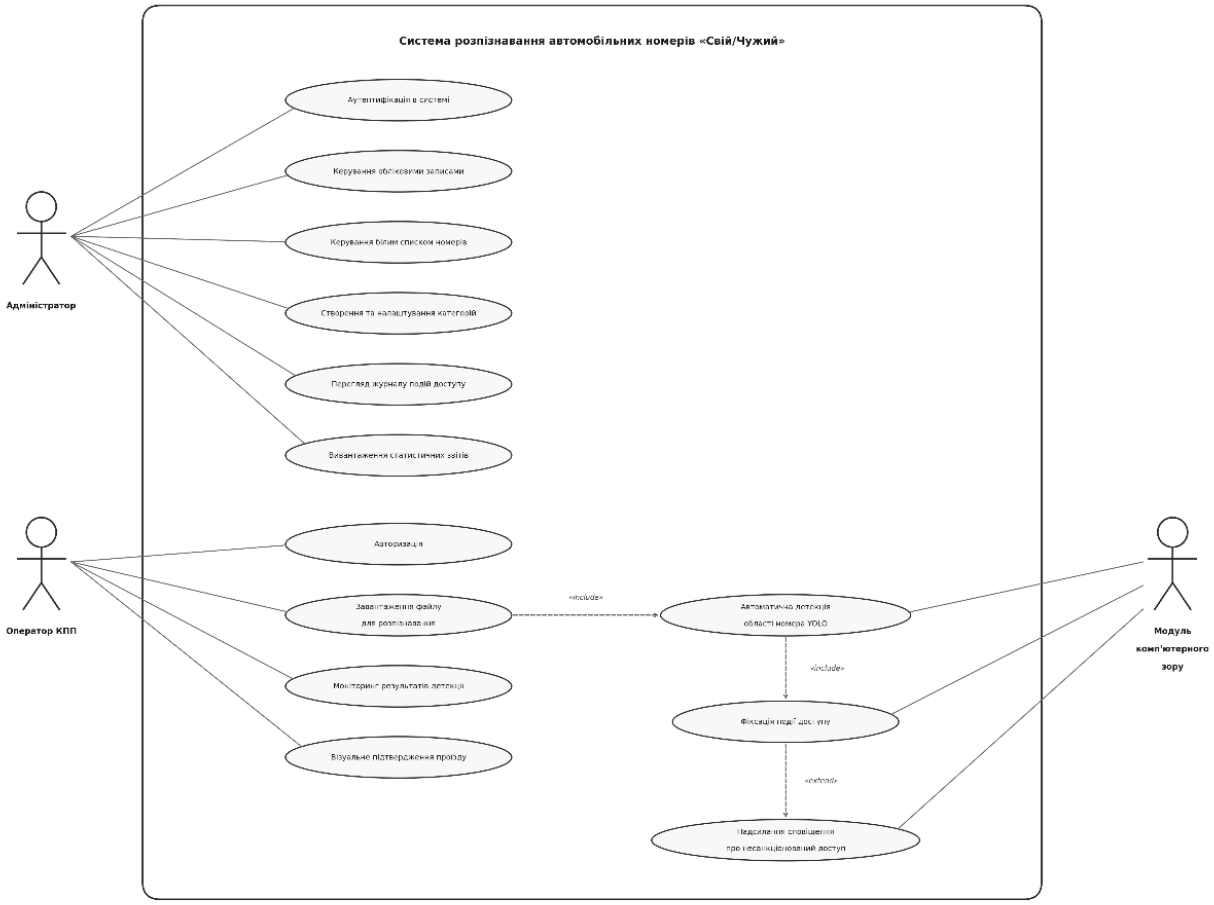

Рисунок 3.1 – Діаграма прецедентів роботи застосунку

Побудована діаграма прецедентів повністю охоплює функціональні можливості системи та демонструє чітке розмежування прав доступу між адміністратором, оператором КПП та модулем комп'ютерного зору, що забезпечує структуровану основу для подальшого проєктування динамічних моделей взаємодії об'єктів.

3.3.2 Реалізація моделей бази даних

Моделі бази даних реалізовано у файлі main/models.py з використанням Django ORM. Кожна модель є Python-класом, що наслідує клас django.db.models.Model та описує структуру відповідної таблиці бази даних.

Модель UserProfile розширює стандартну модель користувача Django та зберігає додаткову інформацію про профіль. Зв'язок із моделлю User реалізовано через поле OneToOneField, що забезпечує відношення «один до одного». Поле avatar використовує тип ImageField із вказанням директорії завантаження avatars/ та значенням за замовчуванням avatars/user.png. Поле updated_at автоматично оновлюється при кожному збереженні об'єкта завдяки параметру auto_now=True.

Модель AutoNumbers є основною таблицею системи контролю доступу та реалізує логіку «Свій/Чужий». Принцип роботи системи побудовано на простому правилі: якщо розпізнаний номерний знак присутній у таблиці AutoNumbers — транспортний засіб вважається «своїм» і отримує дозвіл на проїзд; якщо номер відсутній у базі даних — автомобіль вважається «чужим» і доступ забороняється. Поле numbers має тип CharField з обмеженням максимальної довжини у 8 символів, що відповідає формату українського державного реєстраційного номера.

Модель Category реалізує категоризацію номерних знаків за країною реєстрації. Поле category використовує параметр choices із переліком допустимих значень: Ukraine, Europe, USA, Japan. Зв'язок «багато до багатьох» із моделлю AutoNumbers реалізовано через поле ManyToManyField, яке автоматично створює проміжну таблицю main_category_numbers у базі даних.

Модель PlateScan зберігає результати кожного сканування. Для зберігання масиву розпізнаних номерів використовується спеціалізоване поле ArrayField із базовим типом CharField. Координати обмежувальних рамок від детектора YOLO зберігаються у полі raw_bboxes типу JSONField. Поля image та video є необов'язковими (blank=True, null=True), що дозволяє створювати сканування як із зображенням, так і з відеофайлом (див. рис. 3.1).

Рисунок 3.2 – ER-Діаграма бази даних

3.3.3 Текстовий аналіз та моделювання динаміки взаємодії об'єктів

Діаграма послідовності відображає часову послідовність взаємодії об'єктів у межах виконання конкретного функціонального сценарію. Для системи «Свій/Чужий» найважливішим процесом, що визначає ефективність її функціонування, є сценарій завантаження кадру, його обробки та прийняття рішення про допуск автомобіля. Текстовий опис цього процесу на рівні об'єктів моделі виглядає наступним чином:

– **ініціація запиту -** актор (Оператор КПП) через графічний інтерфейс користувача ініціює POST-запит, прикріплюючи файл зображення транспортного засобу;

– **маршрутизація та перевірка -** вебсервер Django приймає запит, перевіряє сесію користувача на наявність прав доступу і передає керування об'єкту представлення;

– **збереження та верифікація файлу -** представлення взаємодіє з модулем файлової системи викликаючи метод збереження. Об'єкт сховища повертає унікальний шлях до збереженого медіаресурсу та передає його об'єкту MimeTypes для автоматичного визначення формату (зображення чи відео);

– **інференс нейромережевого модуля -** перевірений шлях до файлу передається об'єкту ANPR Detector. Цей об'єкт послідовно викликає методи обробника OpenCV для конвертації кадру у Grayscale та усунення шумів. Після цього трансформована матриця передається об'єкту YOLO Model, яка повертає координати обмежувальної рамки. Вирізаний фрагмент передається об'єкту EasyOCR Engine, який повертає сирий розпізнаний текст;

– **нормалізація та транзакція ORM -** об'єкт представлення очищує текст за допомогою вбудованого об'єкта регулярних виразів і формує SQL-запит через Django ORM до об'єкта моделі AutoNumbers;

– **прийняття рішення -** об'єкт бази даних виконує пошук і повертає булеве значення: True (якщо номер знайдено у білому списку) або False (якщо номер відсутній). На основі цього об'єкт представлення генерує новий запис для журналу подій у моделі PlateScan;

– **фінальна відповідь -** об'єкт представлення повертає сформований HTTP-код відповіді та рендерить оновлений HTML-шаблон, передаючи актору (Оператору) візуальний статус транзакції («Доступ дозволено» зеленим кольором або «Доступ заборонено» червоним кольором).

Цей ланцюжок взаємодії демонструє асинхронну стійкість системи та чіткий поділ обов'язків між системними компонентами, що виключає перевантаження сервера при високій інтенсивності запитів.

3.3.4 Моделювання життєвого циклу об'єктів транзакцій доступу

Діаграма станів описує динамічну поведінку окремого об'єкта протягом його життєвого циклу, фіксуючи всі можливі стани та події (тригери), що викликають переходи між ними. Головним об'єктом аналізу в межах системи є Транзакція

обробки події доступу (запис у PlateScan), життєвий цикл якої складається з таких станів:

– стан «Створення» - об'єкт переходить у цей стан у момент успішного завершення HTTP POST-запиту та збереження файлу у тимчасову директорію сервера. На цьому етапі дані про номерний знак ще відсутні;

– стан «Попередня обробка» - тригером переходу є виклик методів OpenCV. Об'єкт перебуває в цьому стані під час виконання операцій фільтрації, розмиття за Гаусом та бінаризації зображення;

– стан «Аналіз детекції» - перехід відбувається в момент передачі кадру в модель YOLO. Об'єкт очікує на повернення матриці координат bounding box. Якщо модель повертає відсутність об'єктів або рівень впевненості є критично низьким, об'єкт миттєво переходить у фінальний стан «Помилка розпізнавання»;

– стан «Розпізнавання тексту» - активується під час роботи EasyOCR. Об'єкт утримує цей стан, поки згорткові та рекурентні шари нейромережі трансформують графічні ознаки літер у символьний рядок;

– стан «Валідація та верифікація» - об'єкт переходить у цей стан після отримання тексту. Тут виконується нормалізація через регулярні вирази та безпосередній пошук зрізу номера в таблиці AutoNumbers через Django ORM;

– стан «Свій» / Стан «Чужий» - залежно від результату логічної перевірки в базі даних, об'єкт розгалужується в один із цих двох стаціонарних станів, що супроводжується фіксацією відповідного прапора в базі даних PostgreSQL;

– стан «Архівовано» - фінальний стан об'єкта події. Перехід здійснюється автоматично після успішного збереження всієї транзакції комп'ютерного зору в базу даних та оновлення інтерфейсу оператора. Об'єкт залишається в цьому стані для подальшого перегляду в журналі історії та аналізу статистики;

Розроблені UML-моделі дозволили повністю формалізувати логічну структуру системи розпізнавання автомобільних номерів, оптимізувати черговість виконання сервісних методів та забезпечити надійну основу для написання чистого програмного коду, виключивши появу архітектурних помилок на етапі кодування.

Висновки до розділу 3

У третьому розділі кваліфікаційної роботи було виконано комплексне проєктування, математичне обґрунтування та об'єктно-орієнтоване моделювання системи автоматичного розпізнавання автомобільних номерних знаків «Свій/Чужий». На основі проведеного аналізу технологій, що включає чистий вебфреймворк Django та реляційну СКБД PostgreSQL, було розширено специфічними структурами даних (такими як масиви ArrayField та об'єкти JSONField), що дозволило адаптувати схему даних під збереження результатів роботи алгоритмів комп'ютерного зору.

Було детально досліджено та теоретично обґрунтовано вибір сучасних архітектур глибокого навчання для побудови ANPR-модуля. Використання одноетапного детектора YOLO дозволило досягти високої швидкості та точності локалізації номерних пластин у складних зовнішніх умовах експлуатації, а інтеграція OCR-модуля на базі архітектур CRAFT та CRNN забезпечила стійке посимвольне розпізнавання тексту з урахуванням контексту послідовностей символів за допомогою двонаправлених рекурентних шарів LSTM та декодування CTC.

За допомогою уніфікованої мови моделювання UML було побудовано функціональні та динамічні моделі системи. Опис прецедентів користувачів, об'єктних взаємодій на рівні діаграм послідовності та аналіз життєвого циклу транзакцій доступу дозволили повністю формалізувати логіку роботи вебзастосунку та підготувати надійне підґрунтя для безпосереднього етапу програмної реалізації, тестування та впровадження системи на об'єктах контролю доступу.

Діаграма прецедентів визначила два основних актори системи — адміністратора та оператора, а також повний набір функціональних можливостей, доступних кожному з них: від авторизації та завантаження зображень до управління білим списком номерних знаків та перегляду журналу подій.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

У даному розділі описується безпосередній процес програмної реалізації системи розпізнавання автомобільних номерів «Свій/Чужий». Розглядається структура Django-проєкту, реалізація моделей бази даних, представлень, шаблонів вебінтерфейсу, модуля розпізнавання номерних знаків, а також наводяться результати тестування системи на реальних зображеннях автомобілів.

4.1 Структура Django-проєкту

Програмний проєкт системи реалізовано з використанням стандартної структури Django-додатку. Кореневий каталог проєкту містить конфігураційний пакет із налаштуваннями маршрутизацією URL та точкою входу WSGI/ASGI, а також основний Django-додаток «main», який об'єднує всю бізнес-логіку системи (див. табл. 4.1).

Таблиця 4.1 – Основні робочі модулі проєкту

Файл/Каталог	**Призначення**
config/settings.py	Конфігурація проєкту: підключення до БД PostgreSQL, налаштування медіа-файлів, статичних файлів, встановлених додатків
config/urls.py	Кореневий маршрутизатор URL-адрес
main/models.py	Визначення ORM моделей: UserProfile, AutoNumbers, Category, PlateScan
main/views.py	Представлення (Views) для обробки HTTP-запитів
main/forms.py	Django-форми для валідації вхідних даних
main/urls.py	Маршрутизація URL-адрес додатку main
main/admin.py	Реєстрація моделей у адміністративній панелі Django
main/signals.py	Обробники сигналів post_save для автоматичної перевірки доступу
main/services/detector.py	Модуль детекції номерного знаку (YOLO)

Кінець таблиці 4.1

Файл/Каталог	**Призначення**
main/services/ocr.py	Модуль оптичного розпізнавання символів (EasyOCR)
main/services/preprocessing.py	Функції попередньої обробки зображень (OpenCV)
main/services/normalizer.py	Функції нормалізації розпізнаного тексту
templates/	HTML-шаблони вебінтерфейсу
static/	Статичні файли (CSS, JavaScript, зображення)
media/	Завантажені користувачами файли (аватари, сканування)
requirements.txt	Перелік Python-залежностей проєкту
manage.py	Утиліта керування Django-проєктом

Проєкт організовано за стандартною структурою Django-застосунку із чітким розділенням на конфігураційний пакет, модуль розпізнавання номерних знаків та основний додаток бізнес-логіки.

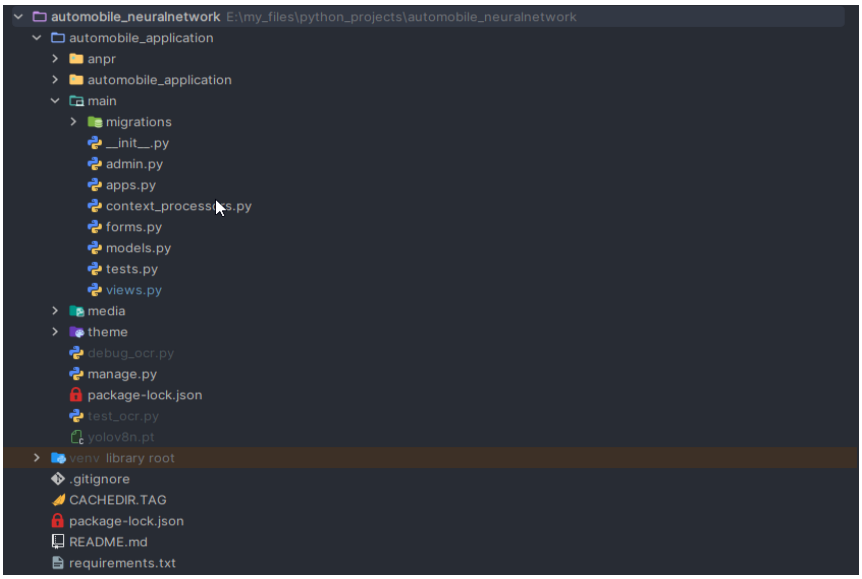

Рисунок 4.1 – Структура робочої директорії

Для підключення до бази даних PostgreSQL у файлі settings.py налаштовано параметри з'єднання: ім'я бази даних, хост, порт, ім'я користувача та пароль. Конфіденційні дані винесено у змінні середовища через бібліотеку python-decouple, що забезпечує безпечне зберігання облікових даних.

Для забезпечення роботи системи використовується набір спеціалізованих Python-бібліотек, кожна з яких відповідає за окремий функціональний аспект: від серверної логіки та взаємодії з базою даних до обробки зображень та розпізнавання номерних знаків. Перелік основних залежностей проєкту із зазначенням версій та призначення наведено у таблиці 4.2 (див. табл. 4.2).

Таблиця 4.2 – Використані бібліотеки та версії у застосунку

Бібліотека	**Версія**	**Призначення**
Django	5.0+	Вебфреймворк серверної частини
psycopg2-binary	2.9+	Драйвер підключення до PostgreSQL
opencv-python	4.9+	Бібліотека комп'ютерного зору для обробки зображень
easyocr	1.7+	Оптичне розпізнавання символів
ultralytics	8.1+	Фреймворк для моделей YOLO
Pillow	10.0+	Обробка зображень у Python
python-decouple	3.8+	Управління змінними середовища
numpy	1.26+	Робота з масивами та матрицями

Після створення моделей було згенеровано та застосовано міграції бази даних за допомогою команд Django: python manage.py makemigrations для створення файлів міграцій та python manage.py migrate для їх застосування до бази даних PostgreSQL.

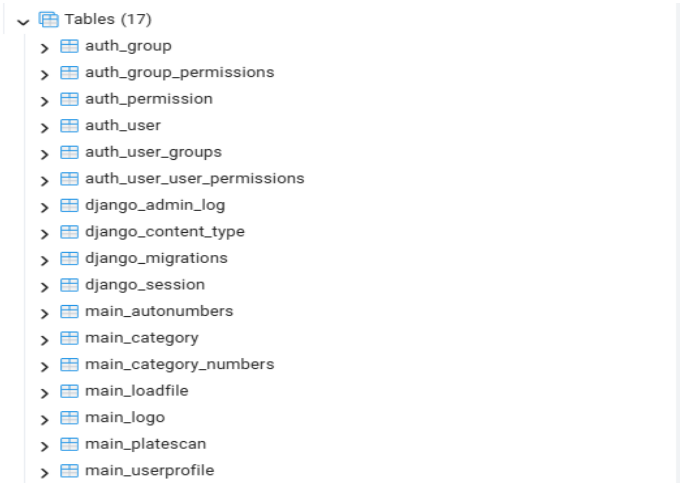

Рисунок 4.2 – Структура таблиць у СКБД «PgAdmin4»

4.2 Структура бази даних

Для зберігання даних системи використовується СКБД PostgreSQL. Структура бази даних складається з трьох основних таблиць, які забезпечують зберігання інформації про дозволені транспортні засоби, журнал подій доступу та облікові записи користувачів системи (див. табл. 4.3).

Таблиця 4.3– Схема профілю користувача у базі даних

Назва поля	**Тип даних**	**Обмеження**	**Опис**
id	SERIAL	PRIMARY KEY	Унікальний ідентифікатор профілю
user_id	INTEGER	FOREIGN KEY (auth_user.id), UNIQUE	Зовнішній ключ на таблицю користувачів
avatar	VARCHAR(100)	DEFAULT 'avatars/user.png'	Шлях до зображення аватара користувача
updated_at	TIMESTAMP	AUTO NOW	Дата та час останнього оновлення профілю

Основною таблицею системи є таблиця номерних знаків, у якій зберігаються всі додані користувачем номери із зазначенням статусу доступу (див. табл. 4.4).

Таблиця 4.4 – Схема номерних знаків у базі даних

Назва поля	Тип даних	Обмеження	Опис
id	SERIAL	PRIMARY KEY	Унікальний ідентифікатор запису
user_id	INTEGER	FOREIGN KEY (auth_user.id), NOT NULL	Зовнішній ключ на власника запису
numbers	VARCHAR(8)	NOT NULL	Рядок номерного знаку
is_allowed	BOOLEAN	NOT NULL	Прапор дозволу доступу (TRUE – дозволено, FALSE – заборонено)
time_create	TIMESTAMP	AUTO NOW ADD	Дата та час створення запису

Для категоризації номерних знаків за країною реєстрації використовується таблиця категорій (див. табл 4.5).

Таблиця 4.5 – Схема категоризації номерних знаків

Назва поля	Тип даних	Обмеження	Опис
id	SERIAL	PRIMARY KEY	Унікальний ідентифікатор категорії
user_id	INTEGER	FOREIGN KEY (auth_user.id), NOT NULL	Зовнішній ключ на власника категорії
category	VARCHAR(20)	NOT NULL, CHOICES	Країна реєстрації (Ukraine, Europe, USA, Japan)

Зв'язок між категоріями та номерними знаками реалізовано через проміжну таблицю типу «багато до багатьох», що дозволяє одному номерному знаку належати до декількох категорій, а одній категорії — містити декілька номерних знаків (див. табл. 4.6)

Таблиця 4.6 – Схема зав’язків між таблицями бази даних

Назва поля	**Тип даних**	**Обмеження**	**Опис**
id	SERIAL	PRIMARY KEY	Унікальний ідентифікатор зв'язку
category_id	INTEGER	FOREIGN KEY (main_category.id), NOT NULL	Зовнішній ключ на категорію
autonumbers_id	INTEGER	FOREIGN KEY (main_autonumbers.id), NOT NULL	Зовнішній ключ на номерний знак

Ключовою таблицею для модуля розпізнавання є таблиця результатів сканування, яка зберігає завантажені зображення або відео, розпізнані номерні знаки та координати їх розташування на зображенні (див. табл. 4.7).

Таблиця 4.7 – Схема з результатами сканування

Назва поля	**Тип даних**	**Обмеження**	**Опис**
id	SERIAL	PRIMARY KEY	Унікальний ідентифікатор сканування
owner_id	INTEGER	FOREIGN KEY (auth_user.id), NOT NULL	Зовнішній ключ на користувача,
Name	VARCHAR(120)	NOT NULL	Назва сканування
image	ImageField	NULL, BLANK	Шлях до завантаженого зображення (scans/images/)
video	FileField	NULL, BLANK	Шлях до завантаженого відео (scans/videos/)

Кінець таблиці 4.7

Назва поля	Тип даних	Обмеження	Опис
plate_texts	ArrayField(VARCHAR)	NOT NULL	Масив рядків із розпізнаними номерними знаками
is_ukrainian	BOOLEAN	NOT NULL	Прапор належності номера до українського формату
raw_bboxes	JSONField	NOT NULL	JSON із координатами обмежувальних рамок від детектора
created_at	TIMESTAMP	AUTO NOW ADD	Дата та час створення запису

Зв'язки між таблицями організовано наступним чином. Таблиця main_userprofile має зв'язок «один до одного» з таблицею auth_user. Таблиці main_autonumbers, main_category та main_platescan мають зв'язок «багато до одного» з таблицею auth_user - кожен користувач може мати необмежену кількість номерних знаків, категорій та сканувань. Таблиці main_category та main_autonumbers з'єднані зв'язком «багато до багатьох» через проміжну таблицю main_category_numbers

4.3 Реалізація модуля розпізнавання номерних знаків

Модуль розпізнавання номерних знаків реалізовано у окремому каталозі anpr/, який містить основний файл логіки detector.py та файли попередньо навчених

моделей нейронних мереж: best.pt (модель YOLO для детекції номерних знаків) та license_plate_detector.pt (додаткова модель детектора). Такий підхід дозволяє ізолювати компонент комп'ютерного зору від бізнес-логіки вебзастосунку та замінювати моделі без необхідності модифікації коду серверної частини.

Файл detector.py об'єднує всі етапи обробки зображення в єдиному модулі: попередню обробку засобами бібліотеки OpenCV, детекцію області номерного знаку за допомогою моделі YOLO та оптичне розпізнавання символів через EasyOCR. При ініціалізації модуля відбувається одноразове завантаження моделі YOLO з файлу best.pt через фреймворк Ultralytics та створення об'єкта EasyOCR Reader з підтримкою мов «en» та «uk» для розпізнавання латинських та кириличних символів. Об'єкти моделей зберігаються у пам'яті сервера для прискорення подальших запитів.

Процес обробки зображення у detector.py виконується послідовно. Спочатку вхідне зображення зчитується функцією cv2.imread() та перетворюється у градації сірого за допомогою cv2.cvtColor() із параметром cv2.COLOR_BGR2GRAY. Для зменшення шуму застосовується гаусівський фільтр cv2.GaussianBlur() з ядром розміром 5×5. Підготовлене зображення передається до моделі YOLO, яка повертає список обмежувальних рамок із координатами у форматі (x1, y1, x2, y2) та рівнем впевненості. Рамки з рівнем впевненості нижче порогу 0.5 відкидаються як ненадійні. Для кожної знайденої рамки виконується вирізання відповідного фрагмента зображення та його передача у метод readtext() бібліотеки EasyOCR. Результатом є текстовий рядок розпізнаних символів.

Після розпізнавання виконується нормалізація тексту: видалення пробілів та спеціальних символів, приведення усіх літер до верхнього регістру та заміна візуально подібних символів кирилиці та латиниці. Нормалізований рядок повертається до представлення у main/views.py, де відбувається перевірка за базою даних та збереження результатів сканування у модель PlateScan (див. табл. 4.8).

Таблиця 4.8 – Модулі нейромережі у проєкті

Файл	Призначення
__init__.py	Ініціалізація Python-пакету
detector.py	Основна логіка детекції та розпізнавання: попередня обробка (OpenCV), детекція (YOLO), OCR (EasyOCR), нормалізація тексту
best.pt	Файл ваг навченої моделі YOLO для детекції номерних знаків
license_plate_detector.pt	Додатковий файл ваг детектора номерних пластин

4.4 Реалізація вебінтерфейсу користувача

Вебінтерфейс системи розпізнавання автомобільних номерів «Свій/Чужий» побудовано з використанням шаблонізатора Django Templates та CSS-фреймворку Tailwind CSS. Структура шаблонів організована за принципом наслідування: базовий шаблон base.html містить загальну розмітку сторінки, навігаційне меню, підключення стилів та скриптів, а дочірні шаблони розширюють його через тег {% extends 'base.html' %}.

Сторінка авторизації є точкою входу до системи. Вона містить форму з полями для введення логіну та паролю, кнопку входу та посилання на реєстрацію нового облікового запису. При введенні некоректних даних під формою відображається повідомлення про помилку. Після успішної авторизації користувач перенаправляється на головну сторінку системи (див. рис. 4.3).

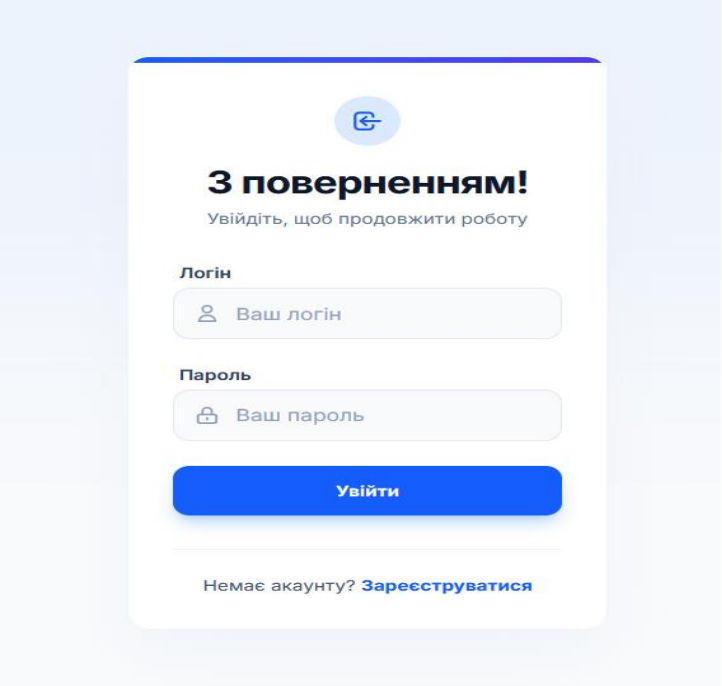

Рисунок 4.3 – Сторінка авторизації

Головна сторінка системи поєднує інформаційну та навігаційну функції. У верхній частині розташовано навігаційну панель із логотипом «Свій/Чужий», кнопкою перемикання теми оформлення та інформацією про авторизованого користувача.

Особливістю головної сторінки є інформаційний блок із оперативними даними про втрати ворога у війні проти України, який отримує дані в реальному часі через зовнішній API. Блок відображає ключові категорії втрат (особовий склад, танки, ББМ, артилерійські системи, літаки, БПЛА тощо) у вигляді наочних лічильників.

Нижче розташовано секцію-герой із заголовком «Інтелектуальна система контролю доступу» та описом основних можливостей системи. Три інформаційні картки представляють ключові переваги: швидкість роботи, висока точність розпізнавання та надійність і безпека. Кнопка «Розпочати роботу» перенаправляє користувача на сторінку сканування (див. рис. 4.4).

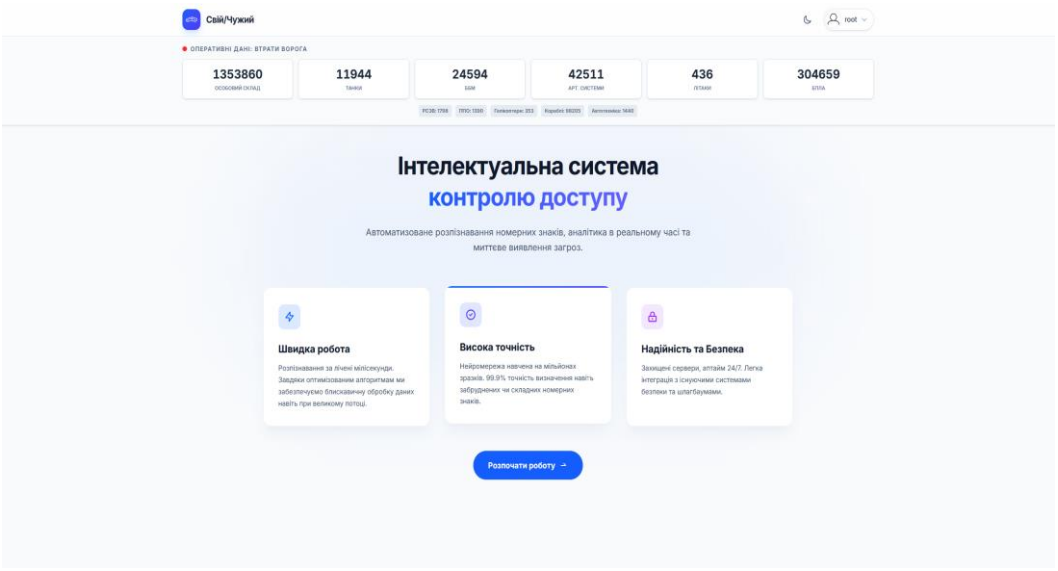

Рисунок 4.4 – Головна сторінка застосунку

Інтерфейс системи підтримує дві теми оформлення: світлу та темну. Перемикання здійснюється натисканням на іконку у навігаційній панелі. Обрана тема зберігається у локальному сховищі браузера та автоматично застосовується при повторному відкритті системи. Темна тема забезпечує комфортну роботу в умовах зниженого освітлення, що є важливим для операторів КПП, які працюють у нічний час (див. рис. 4.5).

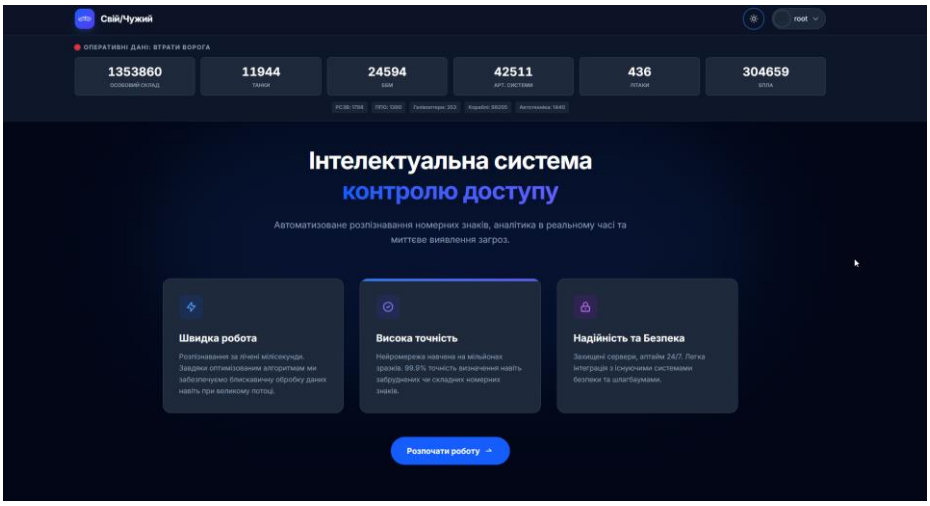

Рисунок 4.5 – Зовнішній вигляд застосунку при темній темі

Сторінка сканування є основною робочою сторінкою системи, на якій виконується завантаження зображення або відео автомобіля та запуск процесу

розпізнавання номерного знаку. Сторінка містить форму з полем для введення назви сканування та зоною завантаження файлу. Після натискання кнопки розпізнавання файл відправляється на сервер, де обробляється модулем anpr/detector.py. Результати відображаються на цій же сторінці: зображення з виділеними обмежувальними рамками, розпізнані номери та результат перевірки за базою даних із кольоровою індикацією статусу «Свій» або «Чужий». Для кожного розпізнаного номера відображається позначка наявності у білому списку, що дозволяє оператору миттєво оцінити результат без необхідності ручної перевірки (див. рис. 4.6).

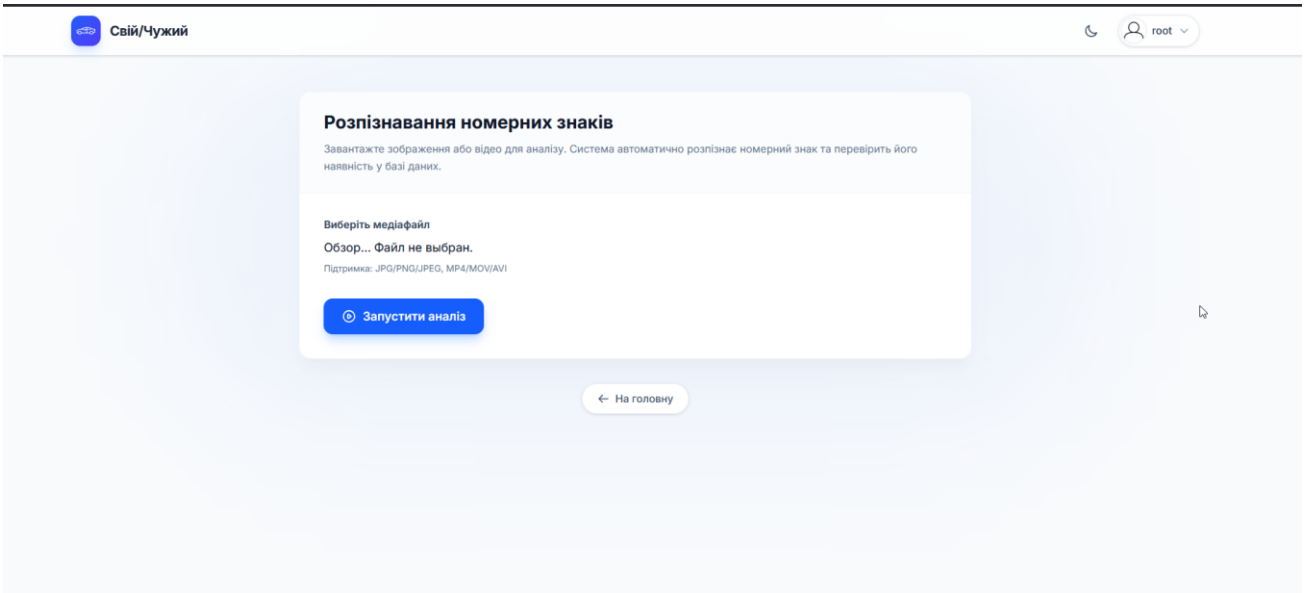

Рисунок 4.6 – Сторінка сканування

Після завантаження зображення або відео та натискання кнопки «Запустити аналіз» система передає файл на сервер, де модуль anpr/detector.py виконує послідовну обробку: попередню фільтрацію засобами OpenCV, детекцію області номерного знаку моделлю YOLO та розпізнавання символів бібліотекою EasyOCR. Результати відображаються на цій же сторінці у блоці «Результат аналізу»: ліворуч — завантажене зображення з попереднім переглядом, праворуч — загальний статус доступу з кольоровою індикацією та список розпізнаних номерів із позначкою наявності у базі даних. На рисунку 4.7 наведено приклад успішного розпізнавання

номерного знаку AC7621EI, який було знайдено у білому списку, внаслідок чого система повернула статус «Доступ дозволено» (див. рис. 4.7).

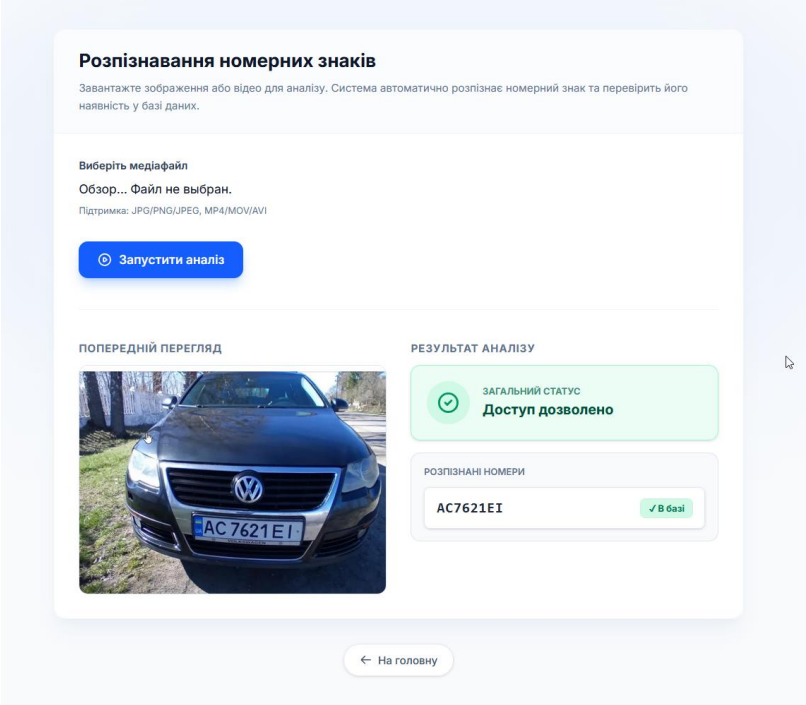

Рисунок 4.7 – Результат розпізнавання знаку

Сторінка профілю відображає інформацію про поточного користувача: аватар, ім'я, форми зміни паролю та аватару (див. рис. 4.8).

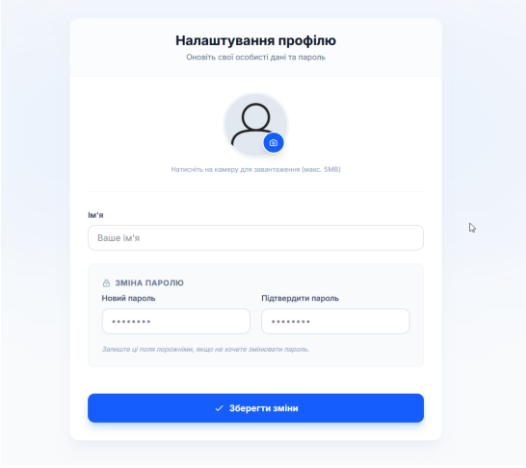

Рисунок 4.8 – Сторінка профілю користувача

Механізм автоматичної перевірки доступу реалізує ключову логіку системи «Свій/Чужий». Після того як модуль anpr/detector.py повертає розпізнаний

текстовий рядок номерного знаку, у представленні main/views.py виконується пошук цього номера у таблиці AutoNumbers бази даних. Принцип роботи є простим: якщо номер знайдено у базі — автомобіль вважається «своїм» і отримує дозвіл на проїзд, якщо номер відсутній — автомобіль вважається «чужим» і доступ забороняється (див. табл. 4.9)

Таблиця 4.9 – Кроки розпізнавання номеру

Крок	**Дія**	**Умова переходу**
1	Отримання масиву plate_texts від модуля detector.py	Якщо масив порожній — статус «Не розпізнано»
2	Нормалізація кожного рядка (верхній регістр, заміна подібних символів)	Перехід до кроку 3
3	Пошук у таблиці AutoNumbers через ORM-запит objects.filter(numbers=plate_text)	Якщо знайдено — крок 4, якщо ні — крок 5
4	Номер знайдено у базі	Статус «Свій — Доступ дозволено»
5	Номер відсутній у базі	Статус «Чужий — Доступ заборонено»
6	Збереження результатів у модель PlateScan	Відображення результату у вебінтерфейсі

Результати сканування зберігаються у моделі PlateScan із зазначенням масиву розпізнаних номерів, координат обмежувальних, прапора належності до українського формату та шляху до завантаженого зображення або відео. Це забезпечує повний журнал усіх спроб доступу для подальшого аналізу та перегляду адміністратором.

Для перевірки коректності роботи системи було проведено комплексне тестування, яке складалося з трьох етапів: тестування функціональності вебзастосунку, тестування точності модуля розпізнавання номерних знаків та тестування логіки перевірки доступу «Свій/Чужий».

Для оцінки точності модуля розпізнавання було підготовлено тестовий набір із 50 зображень автомобілів з українськими державними реєстраційними номерами, знятих за різних умов (див. табл. 4.10).

Таблиця 4.10 – Тестування точності розпізнавання

Категорія	**Кількість зображень**	**Опис умов**
Денне освітлення, фронтальний ракурс	20	Ясна погода, камера на рівні номерного знаку
Денне освітлення, бічний ракурс	10	Ясна погода, кут відхилення до 30°
Вечірнє/нічне освітлення	10	Знижена освітленість, штучне підсвічування
Часткове забруднення номера	5	Бруд, сніг або інші перешкоди
Дощова погода	5	Краплі води на номері або об'єктиві

Результати тестування точності розпізнавання за кожною категорією (див. табл. 4.11).

Таблиця 4.11 - Результати тестування точності розпізнавання

Категорія	**Кількість**	**Правильно**	**Частково**	**Не розпізнано**	**Точність, %**
Денне освітлення, фронтальний	20	19	1	0	95,0
Денне освітлення, бічний	10	8	1	1	80,0
Вечірнє/нічне освітлення	10	7	2	1	70,0
Часткове забруднення	5	3	1	1	60,0
Дощова погода	5	3	1	1	60,0

Під «правильно розпізнано» розуміється повний збіг усіх символів розпізнаного номера з дійсним номером транспортного засобу. «Частково розпізнано» означає помилку в одному або двох символах. «Не розпізнано»

означає, що модуль не зміг виявити область номерного знаку або допустив помилку більш ніж у двох символах. Найвища точність (95%) досягається за оптимальних умов зйомки — денне освітлення та фронтальний ракурс камери відносно номерного знаку.

Окремо було протестовано логіку прийняття рішення «Свій/Чужий» для різних сценаріїв взаємодії розпізнаного номера з базою даних (див. табл. 4.12).

Таблиця 4.12 – Результати тесту логіки прийняття рішень

Тестовий сценарій	**Кількість тестів**	**Очікуваний результат**	**Успішно**	**Неуспішно**
Номер присутній у базі, розпізнано правильно	20	«Свій» - доступ дозволено	20	0
Номер відсутній у базі, розпізнано правильно	10	«Чужий» - доступ заборонено	10	0
Номер присутній у базі, розпізнано частково	6	«Чужий» - доступ заборонено	6	0
Номер не розпізнано	4	Повідомлення про помилку	4	0
Всього	40		40	0

Тестування підтвердило, що система коректно обробляє всі можливі сценарії. Номери, що присутні у базі даних, правильно отримують статус «Свій» із дозволом на проїзд. Номери, відсутні у базі, отримують статус «Чужий» із відмовою у доступі. При частковому розпізнаванні (помилка в 1–2 символах) система не допускає хибного надання доступу — такий номер не збігається з жодним записом у базі і отримує статус «Чужий». Загалом усі 40 тестових сценаріїв було пройдено успішно з коректністю 100%, що підтверджує надійність реалізованого алгоритму перевірки. Жодного випадку хибного спрацювання зафіксовано не було, що є критично важливим з точки зору безпеки (див. рис. 4.9).

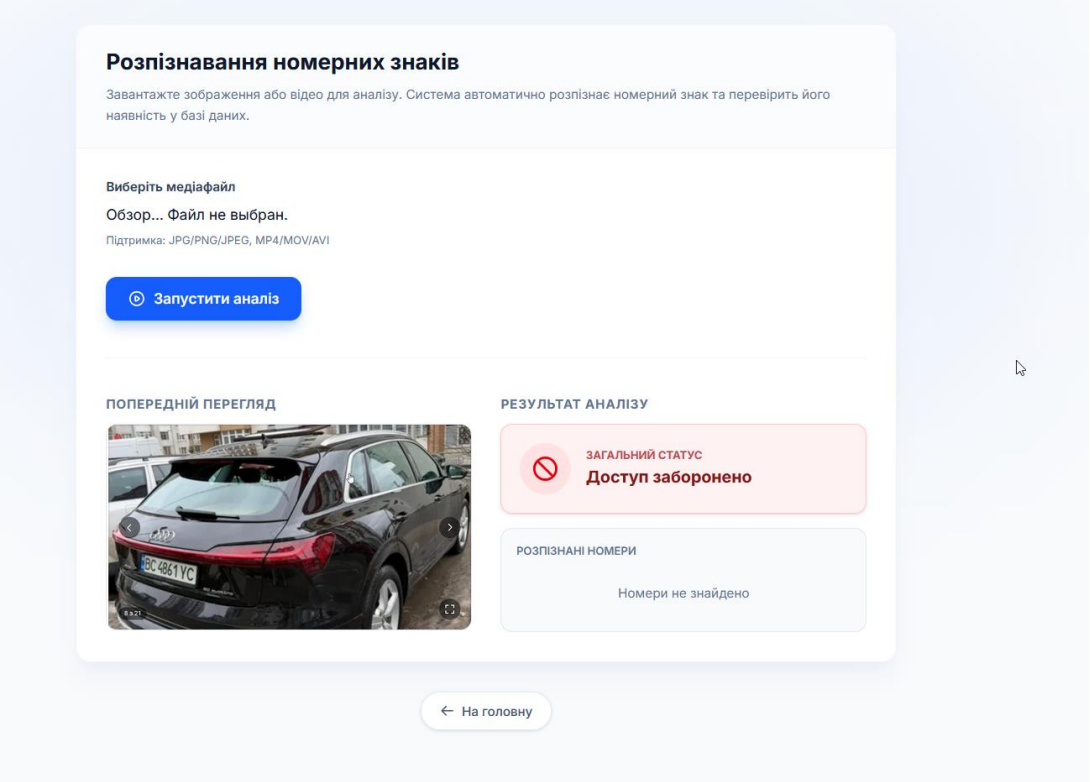

Рисунок 4.9 – Результат розпізнавання при заборонених номерах

Висновки до розділу 4

У четвертому розділі кваліфікаційної роботи було здійснено безпосередню програмну реалізацію системи розпізнавання автомобільних номерів «Свій/Чужий». Описано структуру Django-проєкту, що складається з конфігураційного пакету automobile_application, модуля розпізнавання anpr із попередньо навченими моделями YOLO та основного додатку main, який містить моделі бази даних, представлення, форми та шаблони вебінтерфейсу. Реалізовано ORM-моделі бази даних PostgreSQL із використанням спеціалізованих типів полів ArrayField та JSONField для зберігання результатів роботи нейромережевого модуля.

Модуль розпізнавання номерних знаків, реалізований у файлі anpr/detector.py, об'єднує попередню обробку зображень засобами OpenCV, детекцію області номерного знаку моделлю YOLO та оптичне розпізнавання символів бібліотекою EasyOCR із подальшою нормалізацією тексту. Особливу увагу приділено розробці алгоритмів фільтрації та бінаризації зображень, що

дозволило мінімізувати вплив сторонніх шумів, нерівномірного освітлення та погодних умов на фінальну точність розпізнавання літерно-цифрових комбінацій. Завдяки інтеграції механізмів Django ORM, кожен факт фіксації транспортного засобу автоматично заноситься до журналу логування подій із прив'язкою до унікального ідентифікатора камери, часового штампу та коефіцієнта впевненості моделі (confidence score).

Для забезпечення високої швидкодії системи обробку відеопотоку було винесено в окремі асинхронні потоки, що запобігає блокуванню основного потоку вебсервера під час тривалих обчислень нейромережі. Окрім цього, у межах проєктування логіки «Свій/Чужий» розроблено сервісний шар, який виконує миттєву валідацію розпізнаного номера за масками шаблонів українських державних реєстраційних знаків та здійснює пошук збігів у базі даних дозволених автомобілів.

У випадку відсутності номера у білому списку, система генерує тривожне сповіщення в інтерфейсі оператора та маркує подію як потенційну загрозу безпеці об'єкта. Реалізовано вебінтерфейс користувача з підтримкою світлої та темної теми оформлення, інформаційним блоком оперативних даних про втрати ворога через зовнішній API, сторінками сканування, управління білим списком номерних знаків, категоріями та профілем користувача. Безпеку доступу до адміністративної панелі та функцій керування списками реалізовано за допомогою вбудованої підсистеми автентифікації Django, яка розподіляє права між ролями адміністратора об'єкта та чергового охоронця. Клієнтську частину розроблено з використанням адаптивної верстки, що забезпечує коректне відображення та зручність моніторингу подій як на стаціонарних моніторах КПП, так і на мобільних пристроях персоналу охорони.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було досягнуто поставленої мети — розроблено вебзастосунок для автоматизації контролю доступу транспортних засобів на закриту територію на базі технологій комп'ютерного зору. Для цього було проаналізовано предметну область та існуючі аналоги, що обґрунтувало доцільність розробки власного рішення, адаптованого до українських номерних знаків. Спроєктовано архітектуру системи та структуру бази даних PostgreSQL. Реалізовано модуль розпізнавання номерних знаків на базі YOLO та EasyOCR із точністю розпізнавання 80–95% залежно від умов зйомки. Розроблено веб-інтерфейс на Django із логікою перевірки «Свій/Чужий», білим списком номерів, категоризацією за країнами реєстрації та підтримкою світлої і темної теми оформлення. Проведено комплексне тестування системи, яке підтвердило коректність логіки перевірки доступу на рівні 100% у всіх тестових сценаріях.

Було проведено аналіз предметної області контролю доступу транспортних засобів, досліджено існуючі методи ідентифікації автомобілів та виявлено ключові недоліки традиційних підходів. Огляд комерційних та відкритих аналогів дозволив обґрунтувати доцільність розробки власного рішення, адаптованого до українських номерних знаків.

Обрано та обґрунтовано стек технологій для реалізації системи: вебфреймворк Django, СКБД PostgreSQL, бібліотеку OpenCV для попередньої обробки зображень, модель YOLO для детекції номерних знаків та бібліотеку EasyOCR для оптичного розпізнавання символів. Розроблено специфікацію вимог до програмного забезпечення, структуру бази даних та UML-моделі системи.

Здійснено програмну реалізацію системи із модулем розпізнавання на базі попередньо навчених моделей YOLO та вебінтерфейсом із підтримкою світлої та темної теми оформлення, сторінками сканування, управління білим списком номерних знаків та категоризацією за країнами реєстрації.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Lubna, Mufti N., Shah S. A. A. Automatic Number Plate Recognition: A Detailed Survey of Relevant Algorithms. Sensors. 2021. Vol. 21, № 9. P. 3028. DOI: 10.3390/s21093028.

2. Shashirangana J., Padmasiri H., Meedeniya D., Perera C. Automated License Plate Recognition: A Survey on Methods and Techniques. IEEE Access. 2021. Vol. 9. P. 11203–11225. DOI: 10.1109/ACCESS.2020.3047929.

3. Redmon J., Farhadi A. YOLOv3: An Incremental Improvement. arXiv preprint. 2018. arXiv:1804.02767.

4. Baek Y., Lee B., Han D., Yun S., Lee H. Character Region Awareness for Text Detection. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2019. P. 9365–9374. DOI: 10.1109/CVPR.2019.00959.

5. Shi B., Bai X., Yao C. An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2017. Vol. 39, № 11. P. 2298–2304. DOI: 10.1109/TPAMI.2016.2646371.

6. Meesad P., Thumthong W. Advanced Deep Learning Techniques for Automated License Plate Recognition. Scientific Reports. 2025. Vol. 15. DOI: 10.1038/s41598-025-24967-9.

7. Bradski G., Kaehler A. Learning OpenCV: Computer Vision with the OpenCV Library. Sebastopol : O'Reilly Media, 2008. 555 p.

8. Holovaty A., Kaplan-Moss J. The Definitive Guide to Django: Web Development Done Right. 2nd ed. Berkeley : Apress, 2009. 536 p.

9. OpenCV: Open Source Computer Vision Library. Documentation. URL: https://docs.opencv.org/4.x/ (дата звернення: 25.04.2026).

10. JaidedAI. EasyOCR: Ready-to-use OCR with 80+ supported languages. URL: https://github.com/JaidedAI/EasyOCR (дата звернення: 25.04.2026).

11. Ultralytics. YOLOv8 Documentation. URL: https://docs.ultralytics.com/ (дата звернення: 25.04.2026).

12. Django documentation. Version 5.0. URL: https://docs.djangoproject.com/en/5.0/ (дата звернення: 25.04.2026).

13. PostgreSQL: The World's Most Advanced Open Source Relational Database. Documentation. URL: https://www.postgresql.org/docs/ (дата звернення: 25.04.2026).

14. Python documentation. Version 3.12. URL: https://docs.python.org/3/ (дата звернення: 25.04.2026).

15. Tailwind CSS. Documentation. URL: https://tailwindcss.com/docs (дата звернення: 25.04.2026).

16. Hikvision. ANPR - Automatic Number Plate Recognition. URL: https://www.hikvision.com/en/core-technologies/see-smarter-technology/anpr-automatic-number-plate-recognition/ (дата звернення: 25.04.2026).

17. Dahua Technology. ANPR: Automatic Number Plate Recognition. URL: https://www.dahuasecurity.com/ (дата звернення: 25.04.2026).

18. Plate Recognizer. Automatic License Plate Recognition API. URL: https://platerecognizer.com/ (дата звернення: 25.04.2026).

19. OpenALPR. Open Source Automatic License Plate Recognition. URL: https://github.com/openalpr/openalpr (дата звернення: 25.04.2026).

20. Bootstrap. The most popular HTML, CSS, and JS library in the world. URL: https://getbootstrap.com/docs/ (дата звернення: 25.04.2026).

21. Jocher G., Chaurasia A., Qiu J. Ultralytics YOLOv8. Version 8.0.0. 2023. URL: https://github.com/ultralytics/ultralytics (дата звернення: 25.04.2026).

22. Terven J., Córdova-Esparza D. M., Romero-González J. A. A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS. Machine Learning and Knowledge Extraction. 2023. Vol. 5, № 4. P. 1680–1716. DOI: 10.3390/make5041083.

23. Graves A., Fernández S., Gomez F., Schmidhuber J. Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks.

Proceedings of the 23rd International Conference on Machine Learning (ICML). 2006. P. 369–376. DOI: 10.1145/1143844.1143891.

24. Lin T.-Y., Dollár P., Girshick R., He K., Hariharan B., Belongie S. Feature Pyramid Networks for Object Detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017. P. 2117–2125. DOI: 10.1109/CVPR.2017.106.

25. Laroca R., Severo E., Zanlorensi L. A., Oliveira L. S., Gonçalves G. R., Schwartz W. R., Menotti D. A Robust Real-Time Automatic License Plate Recognition Based on the YOLO Detector. Proceedings of the International Joint Conference on Neural Networks (IJCNN). 2018. P. 1–10. DOI: 10.1109/IJCNN.2018.8489629.

26. He K., Zhang X., Ren S., Sun J. Deep Residual Learning for Image Recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016. P. 770–778. DOI: 10.1109/CVPR.2016.90.

27. Hochreiter S., Schmidhuber J. Long Short-Term Memory. Neural Computation. 1997. Vol. 9, № 8. P. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

28. Almeida E. R., Anttila A., Oliveira L. S., Menotti D. Real-Time License Plate Detection and Recognition Using Deep Neural Network on a Multi-Stage Framework. Sensors. 2023. Vol. 23, № 4. P. 2120. DOI: 10.3390/s23042120.

29. NumPy. The fundamental package for scientific computing with Python. URL: https://numpy.org/doc/ (дата звернення: 25.04.2026).

30. Pillow. Python Imaging Library (Fork). Documentation. URL: https://pillow.readthedocs.io/ (дата звернення: 25.04.2026).

31. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. 3rd ed. Boston : Addison-Wesley, 2003. 208 p.

32. python-decouple. Strict separation of settings from code. URL: https://github.com/HBNetwork/python-decouple (дата звернення: 25.04.2026).

ДОДАТОК А

Код інтеграції API на головну сторінку застосунку

```
def main_view(request):
    url = 'https://russianwarship.rip/api/v2/statistics/latest/'
    logo = Logo.objects.all()
    prof_pic = None

    if request.user.is_authenticated:
        prof_pic = UserProfile.objects.filter(user=request.user).first()


    try:
        response = requests.get(url, timeout=5)
        data_json = response.json()

        stats = data_json['data']['stats']

        units = stats['personnel_units']
        tanks = stats['tanks']
        afv = stats['armoured_fighting_vehicles']
        art = stats['artillery_systems']
        mlrs = stats['mlrs']
        aaws = stats['aa_warfare_systems']
        planes = stats['planes']
        heli = stats['helicopters']
        vft = stats['vehicles_fuel_tanks']
        wcut = stats['warships_cutters']
        cm = stats['cruise_missiles']
        uav = stats['uav_systems']
        sme = stats['special_military_equip']
        submarines = stats['submarines']
        atgm = stats['atgm_srbm_systems']

    except requests.exceptions.RequestException as e:
        return render('theme/base.html', {'error': f'Помилка запиту:{e}.'})
    except requests.exceptions.Timeout:
        return render('theme/base.html', {'error': 'Час очікування сплив.'})

    context = {
        'units':units,
        'tanks':tanks,
        'afv':afv,
        'art':art,
        'mlrs':mlrs,
        'aaws':aaws,
        'planes':planes,
        'heli':heli,
        'vft':vft,
        'wcut':wcut,
        'cm':cm,
        'uav':uav,
        'sme':sme,
        'subm':submarines,
        'atgm': atgm,
        'logo': logo,
        }
return render(request, 'base.html', context)
```

ДОДАТОК Б

Код завантаження відео для розпізнавання

```
@login_required
def anpr_upload(request):
    if request.method == “POST”:
        form = ANPRUploadForm(request.POST, request.FILES)
        if form.is_valid():
            f = form.cleaned_data[“file”]

            name = default_storage.save(f”anpr_uploads/{f.name}”, ContentFile(f.read()))
            path = default_storage.path(name)

            ctype, _ = mimetypes.guess_type(f.name)
            is_image = (ctype or ““).startswith(“image/”)
            is_video = (ctype or ““).startswith(“video/”)

            try:
                if is_image:
                    result = anpr_infer_image_path(path)
                elif is_video:
                    result = anpr_infer_video_path(path)
                else:
                    messages.error(request, “Невідомий тип файлу. Завантажте фото або відео.”)
                    return redirect(“anpr_upload”)
            except Exception as e:
                messages.error(request, f”Помилка під час обробки: {e}”)
                return redirect(“anpr_upload”)

            if “error” in result:
                messages.error(request, result[“error”])
                return redirect(“anpr_upload”)

            accepted = bool(result.get(“accepted”))
            plates = result.get(“plates”, [])

            normalized = []
            for p in plates:
                if isinstance(p, dict):
                    normalized.append({
                        “text”: p.get(“text”, ““),
                        “is_ua”: bool(p.get(“is_ua”)),
                        “score”: p.get(“score”),
                    })
                else:
                    normalized.append({“text”: str(p), “is_ua”: None, “score”: None})

            for p in normalized:
                if p[“text”]:
                    p[“in_db”] = AutoNumbers.objects.filter(numbers=p[“text”][:8]).exists()
                else:
                    p[“in_db”] = False

            accepted = any(p.get(“in_db”) for p in normalized)

            if request.user.is_authenticated:
                plate_texts = [p[“text”] for p in normalized if p[“text”]]
                raw_bboxes = [p.get(“bbox”, []) for p in result.get(“plates”, []) if
isinstance(p, dict)]

                scan = PlateScan(
```

```
                owner=request.user,
                name=f.name,
                plate_texts=plate_texts,
                is_ukrainian=accepted,
                raw_bboxes=raw_bboxes,
            )

            if is_image:
                scan.image.name = name
            elif is_video:
                scan.video.name = name

            scan.save()

        context = {
            “form”: ANPRUploadForm(),
            “uploaded_rel”: name,
            “is_image”: is_image,
            “is_video”: is_video,
            “accepted”: accepted,
            “plates”: normalized,
        }
        return render(request, “anpr_upload.html”, context)

else:
    form = ANPRUploadForm()

return render(request, “anpr_upload.html”, {“form”: form})
```