Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

Кіберфізична система "Розумна парковка" з розпізнаванням моделі та номера автомобіля
Назва теми

КвРКІ 200101.20.01.01 ПЗ
Шифр

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *123 «Комп'ютерна інженерія»*
Шифр, назва

Освітня програма *«Комп'ютерна інженерія та програмування»*
Назва

Виконав: студент IV курсу, група КІ2-20-1 _____ В. Р. Авсієвич
Підпис Ініціали, прізвище

Керівник _____ О. О. Павлова
Підпис, дата Ініціали, прізвище

Нормоконтролер _____ І.О. Засорнова
Підпис, дата Ініціали, прізвище

До захисту допускаю:
Зав. кафедри комп'ютерної інженерії та інформаційних систем _____ Т.О. Говорущенко
Підпис Ініціали, прізвище

«5» червня 2024 р.

Хмельницький 2024

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

Освітній рівень БАКАЛАВР

Галузь знань 12 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Спеціальність 123 КОМП'ЮТЕРНА ІНЖЕНЕРІЯ

Освітня програма «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ ТА ПРОГРАМУВАННЯ»

ЗАТВЕРДЖУЮ

Зав. кафедри Т.О. Говорущенко

"10" 01 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

Авсієвичу Володимиру Руслановичу
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Кіберфізична система "Розумна парковка" з розпізнаванням моделі та номера автомобіля
Керівник проекту (роботи) Павлова О.О., д. ф., доцент каф. комп. інж та інф систем.
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 15.02.2024 р. № 8

2. Строк подання студентом проекту (роботи) на кафедру 01.06.2024 р.

3. Вихідні дані до проекту (роботи) Завдання на кваліфікаційну роботу

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Теоретичні основи досліджуваної проблеми
Обґрунтування вибору компонентів та середовища реалізації
Реалізація підсистеми розпізнавання номерних знаків та моделей автомобілів у кіберфізичній системі «розумна парковка»

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)
Структурна схема та алгоритм роботи підсистеми розпізнавання номерних знаків та моделей автомобілів у кіберфізичній системі «Розумна парковка»
Схема роботи компонентів кіберфізичної системи «Розумна парковка» з розпізнаванням моделі та номера автомобіля та Алгоритм отримання фото з камери через RTSP
Архітектура підсистеми розпізнавання номерних знаків та моделей автомобілів у кіберфізичній системі «Розумна парковка»

6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийня
Нормоконтроль	Засорнова І. О., доцент кафедри КІС		
Антиплагіат	Нічепорук А.О., к.т.н., доцент кафедри КІС		

7. Дата видачі завдання « 10 » 01 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Приміт
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	10.01.2024	виконан
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	01.02.2024	виконан
3	Робота над розділом 1 – теоретичні основи досліджуваної проблеми	01.03.2024	виконан
4	Робота над розділом 2 – обґрунтування вибору компонентів та середовища реалізації	01.04.2024	викона
5	Робота над розділом 3 – реалізація підсистеми розпізнавання номерних знаків та моделей автомобілів у кіберфізичній системі «Розумна парковка»	29.04.2024	викона
6	Оформлення пояснювальної записки згідно вимог	25.05.2024	викона
7	Попередній захист ВКР	30.05.2024	викона
8	Захист ВКР на засіданні ЕК	Червень 2024 року	

Студент _____ В. Р. Авсієвич
Підпис Ініціали, прізвище

Керівник роботи _____ О. О. Павлова
Підпис Ініціали, прізвище

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1		КвРКІ 200101.20.01.01 ПЗ	Пояснювальна записка	65		
			Графічні матеріали			
2		КвРКІ 200101.20.01.01 Е8	Структурна схема та алгоритм роботи підсистеми розпізнавання номерних знаків та моделей автомобілів у кіберфізичній системі «Розумна парковка»	1		
3		КвРКІ 200101.20.01.01 Е8	Схема роботи компонентів кіберфізичної системи «Розумна парковка» з розпізнаванням моделі та номера автомобіля та Алгоритм отримання фото з камери через RTSP	1		
4		КвРКІ 200101.20.01.01 Е8	Архітектура підсистеми розпізнавання номерних знаків та моделей автомобілів у кіберфізичній системі «Розумна парковка»	1		

Зм	Арк	№ докум	Підпис	Дата	КвРКІ 200101.20.01.01 ВП		
Розробив		Авсієвич		3.06	Відомість проекту	Літера	Аркуш	Аркушів
Перевір.		Павлова		3.06		У	1	1
Н. контр.		Засорнова		5.06		ХНУ, КІ2-20-1		
Затв.		Говорущенко		5.06				

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Кіберфізична система "Розумна парковка" з розпізнаванням моделі та номера автомобіля».

Автор роботи: Авсієвич Володимир Русланович.

Керівник роботи: Павлова Ольга Олександрівна.

Пояснювальна записка: 65 с., 22 рис., 2 табл., 4 дод., 53 джерела.

Графічна частина: 3 креслення.

РОЗУМНА ПАРКОВКА, КІБЕРФІЗИЧНА СИСТЕМА, СЕРВЕР, КАМЕРА СПОСТЕРЕЖЕННЯ, АРХІТЕКТУРА, ПРИКЛАДНИЙ ПРОГРАМНИЙ ІНТЕРФЕЙС.

Метою кваліфікаційної роботи є підвищення ефективності автоматизованого розпізнавання моделі та номерного знака автомобіля для кіберфізичної системи «Розумна парковка».

Об'єктом дослідження є процес розпізнавання номерних знаків та моделей автомобілів у кіберфізичній системі «Розумна парковка».

Предметом дослідження є підсистема розпізнавання моделі та номерного знаку автомобіля у кіберфізичній системі «Розумна парковка».

Для досягнення поставленої мети використовуються такі методи дослідження, як методи синтезу, аналізу та моделювання процесів, принципи системного аналізу, теоретико-множинні підходи.

Практичне значення має спроєктована та реалізована підсистема розпізнавання номера та марки автомобіля у кіберфізичній системі "Розумна парковка".

Підпис студента

04.06.2024

Дата

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

АРНЗ – автоматичне розпізнавання номерних знаків

КЗ – комп’ютерний зір

МЗ – машинний зір

МН – машинне навчання

ШІ – штучний інтелект

ПЗ – програмне забезпечення

APNR – automatic plate number recognition

ВСТУП

На сьогоднішній день, розумні речі оточують нас навколо: розумний телевізор, розумний холодильник, розумний дім. «Розум» цих речей полягає у їх можливості автоматично виконувати якісь дії без втручання людини. Ці речі оточують нас всюди і з кожним днем галузь Інтернету речей розвивається все більше і більше.

Одним із прикладів розумної системи є парковка. Ці системи є популярними у світі, але досі в більшості місць присутні парковки старого зразку, де людина в ручному режимі перевіряє хто в'їжджає на парковку та активує шлагбаум. Проте, навіть використання розумних парковок не є тривіальним. Різні рішення, їх ціна, особливості можуть заплутати клієнта, який хоче автоматизувати свою парковку.

В цій роботі було взято за мету створити рішення, яке буде мати збалансовану вартість, просте користування та встановлення. Це рішення буде здатне отримувати номерних знак транспортного засобу та його модель для аналізу та перевірки чи може ця машина проїхати на нашу парковку.

Актуальність роботи полягає у автоматизації розпізнавання номерного знаку та моделі автомобіля за допомогою штучної нейронної мережі. Метою кваліфікаційної роботи є підвищення ефективності автоматизованого розпізнавання моделі та номерного знака автомобіля для кіберфізичної системи "Розумна парковка".

Завдання роботи: розробити підсистему розпізнавання моделі та номерного знаку автомобіля кіберфізичної системі "Розумна парковка".

Об'єктом дослідження є процес розпізнавання номерних знаків та моделей автомобілів у кіберфізичній системі "Розумна парковка"

Предметом дослідження є підсистема розпізнавання моделі та номерного знаку автомобіля у кіберфізичній системі "Розумна парковка".

Практична цінність отриманих результатів полягає у розробці підсистеми розпізнавання моделі та номерного знаку автомобіля у кіберфізичній системі

“Розумна парковка” для підвищення ефективності автоматизованого розпізнавання моделі та номерного знака автомобіля для кіберфізичної системи “Розумна парковка”.

За темою дипломної роботи було взято участь у II Всеукраїнській конференції “Інноваційні технології як основа професійного становлення особистості”, Хмельницький інститут МАУП, м. Хмельницький, та опубліковано тези у збірнику конференцій.

1 ТЕОРЕТИЧНІ ОСНОВИ ДОСЛІДЖУВАНОЇ ПРОБЛЕМИ

1.1 Аналіз існуючих рішень для автоматизації розпізнавання автомобілів

Автоматичне розпізнавання номерних знаків (АРНЗ) – це технологія, яка використовує комп'ютерний зір для ідентифікації номерних знаків транспортних засобів на зображеннях та відео [1]. Її можна застосовувати з існуючими системами відеоспостереження, камерами фіксації порушень правил дорожнього руху або спеціально розробленими камерами. АРНЗ використовується правоохоронними органами в усьому світі для різних цілей, таких як перевірка реєстрації та наявності дійсної ліцензії у транспортних засобів.

Системи АРНЗ можуть зберігати зображення, зняті камерами, а також розпізнаний текст з номерного знаку. Деякі з них також зберігають фотографії водіїв. Зазвичай такі системи використовують інфрачервоне освітлення, щоб забезпечити чітке зображення в будь-який час доби. Технологія АРНЗ має враховувати різноманітність номерних знаків у різних регіонах.

Використання АРНЗ викликає певні занепокоєння щодо конфіденційності, такі як:

- відстеження урядом пересування громадян;
- неправильна ідентифікація;
- високий рівень помилок;
- збільшення державних витрат.

Незважаючи на ці проблеми, АРНЗ набула широкого поширення в багатьох країнах світу. Її популярність зумовлена автоматизацією рутинних завдань, які раніше потребували постійного втручання людини. Завдяки АРНЗ людині зазвичай лише потрібно підтвердити результат, що значно економить час.

Галузь розпізнавання номерних знаків постійно розвивається. Розробляються нові алгоритми, які є більш точними та стійкими до шуму та

спотворень. Знижується вартість обладнання та програмного забезпечення, що робить АРНЗ доступнішою.

Наразі штучний інтелект (ШІ) стрімко розвивається. Багато компаній пропонують відкриті рішення та API для роботи з ШІ. Використання вже навчених нейронних мереж робить такі рішення доступними за ціною та їх можна успішно застосовувати в різних проектах.

Одним із цікавих прикладів інтеграції ШІ є система розпізнавання номерних знаків та моделей транспортних засобів. Такі рішення можуть бути дешевше, ніж існуючі на ринку, та їх можна використовувати з звичайними камерами спостереження [2], без необхідності в спеціалізованому обладнанні.

В рамках нашого проєкту пропонується дослідити можливість використання системи АРНЗ з інтеграцією ШІ для контролю в'їзду на парковку [3]. Перед початком реалізації необхідно проаналізувати існуючі рішення, оцінити їхні переваги та недоліки, а також обрати оптимальний варіант для нашого проекту.

Існує два основних типи систем АРНЗ [4]:

– фіксовані системи. Ці системи встановлені в певному місці, наприклад, на пунктах оплати проїзду, прикордонних переходах або інших стратегічних локаціях. Їх використовують для постійного моніторингу та розпізнавання номерних знаків транспортних засобів, що проїжджають повз них;

– мобільні системи. Ці системи встановлюються на транспортному засобі, такому як поліцейський автомобіль, і можуть використовуватися для сканування номерних знаків інших транспортних засобів під час руху. Їх використовують для мобільного моніторингу та відстеження транспортних засобів, наприклад, розшукуваних.

Оскільки в нашому проєкті ми розглядаємо використання АРНЗ для розпізнавання номеру при в'їзді на парковку [5], зосередимося на фіксованих системах.

Розглянемо приклади популярних фіксованих систем, які вже використовуються у світі [6-9]:

1. SmartParking [10].

Це закрите рішення, орієнтоване на клієнтів у Великобританії, США, Австралії та Новій Зеландії. Воно пропонує повністю автоматизовану систему з використанням SmartCloud, цілодобовий моніторинг парковки, найсучасніше обладнання та безкоштовне встановлення. SmartParking надає можливість налаштувати низку параметрів, таких як дозволений час перебування на парковці, доступ лише для транспортних засобів з білого списку та обмежений час безкоштовного паркування. Система обмінюється даними бездротовим способом з інформаційною панеллю Smart Parking через SmartCloud API. За допомогою панелі керування SmartCloud можна отримувати зведені звіти про активність та події на парковці, де налаштована система, а також керувати білим списком транспортних засобів.

Переваги:

– екосистема продуктів;
– 30 років досвіду на ринку;
– висока ефективність.

Недоліки:

– закритий код;
– використання можливе тільки у 4 країнах світу.

Розглянемо рішення, яке можна застосувати на парковці в Україні.

2. ZKTeco [11].

Це рішення використовує LPR-камеру та UHF-зчитувачі (рисунок 1.1). Для доступу на парковку необхідний номерний знак транспортного засобу в білому списку або UHF-мітка, яка буває двох видів: перша фіксується на бампері або під номерним знаком, а друга - на лобовому склі зсередини. UHF-зчитувач - це RFID-пристрій для зчитування таких міток, який може одночасно зчитувати кілька пасивних UHF-тегів на відстані до 12 метрів. Цей зчитувач може

використовуватися не лише на парковках, а й в інших галузях, де потрібна логістика.

Рішення ZKTeco пропонує багатоступеневу аутентифікацію для контролю доступу на парковку:

– UHF-мітка: коли транспортний засіб під'їжджає до шлагбаума, UHF-зчитувач зчитує UHF-мітку, прикріплену до нього;

– номерний знак: LPR-камера одночасно зчитує номерний знак транспортного засобу.

Якщо номерний знак знаходиться в білому списку та UHF-мітка успішно верифікована, шлагбаум відкривається.

Рисунок 1.1 – Парковка з забезпеченням ZKTeco

Переваги:

– незалежність від географічного розташування;

– багатоступенева аутентифікація: поєднання UHF-міток та розпізнавання номерних знаків забезпечує додатковий рівень безпеки, роблячи неможливим доступ без дійсної мітки та відповідного номерного знаку;

– білий та чорний списки: система дозволяє створювати білий список дозволених номерних знаків, а також чорний список заборонених, забезпечуючи чіткий контроль доступу.

Недоліки:

– використання UHF-міток: клієнтам необхідно придбати та встановити UHF-мітки на свої транспортні засоби, що може бути додатковим витратою;

– закритий код: як і багато інших рішень АРНЗ, ZKTeco має закритий код, що обмежує можливості модифікації та інтеграції з іншими системами.

Рішення ZKTeco пропонує ефективний та гнучкий спосіб контролю доступу на парковку за допомогою багатоступеневої аутентифікації. Оскільки закритий код систем АРНЗ може обмежувати можливості, далі будемо розглядати рішення, яке пропонує свою бібліотеку розпізнавання номерних знаків з відкритим кодом.

3. OpenALPR [12].

Rekor.AI пропонує бібліотеку розпізнавання номерних знаків (АРНЗ) з відкритим кодом, написану на C++ [13]. Її можна використовувати в різних мовах програмування, таких як C#, Java, Go та Python. Бібліотека аналізує фото- та відеопотоки для ідентифікації номерних знаків.

Функціональні можливості:

– утиліта командного рядка для аналізу номерних знаків з ймовірностями результатів;

– підтримка Windows та Linux;

– використання OpenCV та Tesseract OCR.

Переваги:

– безкоштовне рішення: Rekor.AI пропонує безкоштовний план, що робить його доступним для некомерційних та бюджетних проектів;

– відкритий код: розробники можуть вільно модифікувати та інтегрувати бібліотеку у власні системи;

– кросплатформність: бібліотека сумісна з різними мовами програмування та операційними системами.

Недоліки:

– рішення не з коробки: Rekor.AI не пропонує готового продукту, тому користувачам необхідно самостійно інтегрувати бібліотеку у свої програми;

– відсутність підтримки розробників: оскільки бібліотека більше не підтримується з 2016 року, розробники можуть відчувати труднощі з отриманням допомоги з боку Rekor.AI;

– необхідність знання програмування: для використання бібліотеки Rekor.AI потрібні знання програмування та розуміння специфіки роботи з кодом.

На жаль, не існує ідеального рішення АРНЗ, яке б одночасно було безкоштовним, з відкритим кодом та готовим до використання "з коробки". Після аналізу доступних рішень АРНЗ, наступним кроком буде дослідження апаратних та програмних засобів, необхідних для створення власного продукту розумної парковки.

1.2 Аналіз апаратної частини реалізації

Розумна парковка, як і більшість кіберфізичних систем, складається з двох взаємопов'язаних частин: апаратної та програмної.

Апаратна частина відповідає за збір даних з навколишнього середовища під час в'їзду на парковку. До її складу можуть входити [14]:

1. Система відеоспостереження: камери, які використовуються в системах АРНЗ (автоматичного розпізнавання номерних знаків), схожі на звичайні IP-камери [15], що застосовуються для спостереження за дорожнім рухом (рисунок 1.2).

Рисунок 1.2 – АРНЗ-камера Hikvision iDS-2CD7A26G0/P-IZHS

Камери АРНЗ мають і звичайні характеристики [16], такі як:

– роздільна здатність: від 2,4 до 9 МП і більше;
– дальність підсвічування: від 20 до 200 метрів;
– тип підсвічування: ІЧ-підсвічування, видиме світло;
– тип встановлення: внутрішня, вулична;
– тип підключення: PoE, дротове;
– форм-фактор: купольна, циліндрична;
– запис: у хмару, на жорсткий диск або SD-карту;
– додаткові функції: мікрофон, віддалений доступ, тривога і т.д.;
– відеоаналітика: теплова карта, детектор облич, вторгнення і т.д.;
– кут огляду: до 50, 51-80, 81-99, 100 градусів і більше;
– фокусна відстань: 4 мм, 2,8-12 мм, 8-32 мм;
– тип об'єктива: фіксований, варіофокальний;
– матеріал корпусу: метал, пластик;
– формат запису: H.265, H.264.

Потрібні характеристики залежать від мети використання камери. Вбудована система АРНЗ сильно збільшує ціну камери, проте це дозволяє не витрачати гроші на розробку програмного забезпечення.

Не можна й нехтувати роздільною здатністю та кутом огляду камери, адже ці параметри є критичними для якості роботи систему. Тип підсвічування є необов’язковим, якщо система не має використовуватись в умовах відсутності

або поганого освітлення. Інші особливості камери в цілому необхідні тільки для зручності користування нею. Також треба врахувати формат запису, потрібне відеокодування може бути відсутнє.

2. Індукційна петля.

Ці датчики, що монтуються в дорожнє полотно, генерують сигнал при проїзді транспортного засобу над ними. Вони забезпечують точне спрацьовування, але потребують дорожніх робіт для монтажу. Часто працюють в поєднані з автоматичними шлагбаумами.

3. Датчики зважування в русі.

Також вбудовані в дорожнє полотно, вимірюють вагу транспортного засобу. Їх можна використовувати для обчислення плати за паркування або інших цілей.

4. Автоматичні шлагбауми та світлофори.

Шлагбауми використовуються для контролю доступу на парковку, а світлофори регулюють рух транспорту. Їх застосування доцільне на пунктах в'їзду/виїзду, де потрібен суворий контроль.

5. Сервер.

Сервер може бути розміщений локально або в хмарі. Локальний сервер забезпечує повний контроль над даними, але хмарне рішення може бути більш масштабованим та економічно вигідним.

6. Мережеве обладнання.

Мережеве обладнання є важливою частиною системи, адже воно забезпечує зв'язок між апаратними компонентами та передачу даних до програмного забезпечення. Можна використати Ethernet або Wi-Fi технологію. Ethernet кращий для великих відстаней або для стабільного з'єднання, яке критично важливе.

Wi-Fi підходить для менших відстаней або там, де прокладання кабелів неможливе або непрактичне. Мережа повинна мати достатню пропускну

здатність для обробки трафіку даних від усіх компонентів системи без затримок або перевантажень.

7. Освітлення.

Якісне освітлення може значно покращити роботу системи, особливо в умовах низької освітленості, якщо камера не має ІЧ-підсвітлення.

Після аналізу та вибору апаратних компонентів, наступним кроком буде вибір програмного забезпечення для розумної парковки.

1.3 Програмна частина реалізації

Апаратна частина системи розумної парковки, не може функціонувати самостійно. Для обробки даних, отриманих з датчиків та камер, а також для керування системою, необхідне програмне забезпечення [17].

Програмне забезпечення для такої кіберфізичної системи зазвичай складається з декількох основних компонентів:

1. Модуль зйомки.

Цей модуль відповідає за отримання зображення з камери. Зазвичай використовується вбудований модуль камери, до якого можна отримати доступ за допомогою функцій SDK камери.

Іншим варіантом є доступ до RTSP-потоку, який транслює камера в режимі реального часу. Доступ до нього можна отримати різними способами, наприклад, за допомогою бібліотеки OpenCV [18] (написана на C++). Також можна використовувати утиліту ffmpeg, яка є кросплатформною.

2. Модуль обробки зображення.

Цей компонент програмного забезпечення відповідає за первинну обробку зображення, отриманого з камери. До типових операцій обробки належать: шумозаглушення, налаштування контрастності, масштабування. Мета такої обробки - покращити результати розпізнавання номерних знаків. Рекомендується відокремити цей модуль, щоб наступні етапи аналізу зображення не повторювали цю первинну обробку.

Для обробки зображень зазвичай використовується бібліотека OpenCV [19]. Деякі альтернативи - Google Cloud Vision [20] та Microsoft Computer Vision API [21]. Їх варто розглянути, хоча вони й можуть мати певні обмеження.

3. Модуль розпізнавання номерних знаків.

Модуль автоматичного розпізнавання номерних знаків є ключовим компонентом нашої системи. Він відповідає за автоматичне виявлення номерного знаку, його аналіз за допомогою алгоритмів оптичного розпізнавання символів (OCR) та повернення результату. На точність результати впливають різні чинники, такі як освітлення, розташування та кут нахилу номерного знаку, його стан і чистота. Для реалізації цього модуля можна використати кілька підходів:

а) використання готових SDK або API, спеціально розроблених для цієї задачі. Наприклад, OpenALPR надає різні тарифні плани з обмеженою кількістю сканувань фотографій та додатковими функціями, такими як розпізнавання моделі транспортного засобу. Однак, це може бути дорогим рішенням для малого бізнесу, який хоче впровадити розумну парковку [22];

б) створення власного модуля розпізнавання номерних знаків з використанням безкоштовних інструментів, таких як бібліотека OpenCV. Проте сама по собі ця бібліотека не може виконувати цю задачу. Вона надає засоби для створення моделей, але ці моделі базуються на вагових параметрах, отриманих під час тренування штучної нейронної мережі на певному наборі даних.

Головне завдання в цьому підході — отримати потрібну модель, яка може бути доступна безкоштовно в Інтернеті, але не завжди підходить для конкретних умов, наприклад, якщо вона тренувалася на номерних знаках іншої країни. У такому випадку необхідно тренувати власну модель, що вимагає значних часових та обчислювальних ресурсів.

Важливо також глибоко розуміти сам процес навчання нейронної мережі для ефективного використання ресурсів. Також можна звернути увагу на засоби для знаходження об'єктів на фото. Популярність набирають моделі з

сімейства YOLO від Ultralytics, особливо YOLOv8, яка демонструє високу точність та швидкість роботи;

в) використання безкоштовних API комп'ютерного зору. Зазвичай такі API, які надають загальні можливості комп'ютерного зору, пропонують безкоштовні плани з обмеженою кількістю використань за певний період, при цьому вони є достатньо потужними.

Прикладом такого API є Google Cloud Vision API, який пропонує 1000 безкоштовних розпізнавань у різних категоріях, таких як розпізнавання міток, тексту, обличчя, логотипів та локалізації об'єктів. Після 1000 розпізнавань стягується плата: $1.50 за кожну тисячу розпізнавань до 5 000 000 за місяць і $1.00 після 5 мільйонів. Це рішення може бути економічно вигідним для невеликої розумної парковки.

4. Модуль зберігання даних.

Цей модуль відповідає за зберігання інформації про розпізнані номерні знаки. Вибір типу бази даних залежить від структури даних та потреб проекту. Є 2 основних типи баз: реляційні і нереляційні.

До реляційних належать такі бази даних як: Oracle, Microsoft SQL Server, MySQL. Їх перевагами є чітка структура, відповідність ACID, надійність. Проте їх буває складно налаштувати та вони погано масштабуються.

Прикладами нереляційних баз даних є Amazon DynamoDB, Berkeley DB, MemcacheDB, Redis, Riak. Вони швидкі, масштабуються, гнучкі. Проте не відповідають ACID та мають меншу надійність. Для малих проектів часто застосовують MySQL та PostgreSQL через простоту використання та швидкодію.

5. Модуль візуалізації даних.

Цей модуль надає користувачам доступ до інформації про розпізнані номерні знаки. Він може бути представлений у вигляді:

а) веб-інтерфейсу.

Є найпростішим та найдешевшим варіантом. Цей інтерфейс буде доступний з будь-якого пристрою, який має доступ до Інтернету, не потребує

установки, легко оновлюється, масштабований для роботи з великою кількістю користувачів. Може бути написаний на різних мовах програмування, наприклад, PHP, Node.JS, C#.

Переваги: доступність, простота використання, масштабованість. Недоліки: мала гнучкість, може мати багато вразливостей [23];

б) застосунку на ПК.

Це рішення встановлюється на ПК, наприклад у контрольно-пропускному пункті на парковці. Працює швидше ніж сайт, має звичний для системи вигляд. Для розробки можна використати різні фреймворки, наприклад кросплатформний Qt. Використовуються такі мови, як: Java, C#, C++.

Переваги: гнучкість, безпека, висока швидкодія. Недоліки: треба встановлювати, потрібне централізоване оновлення;

в) застосунку на Android/iOS.

Таке рішення є мобільним, також можна використовувати апаратні засоби пристрою [24] при потребі. Менш безпечний ніж веб-інтерфейс або застосунок на ПК. Є можливість кросплатформної розробки, використовуючи фреймворки React Native, Flutter, Xamarin. Складніше в розробці, ніж застосунок на ПК. Використовуються мови Java, Kotlin для Android, Swift для iOS.

Переваги: мобільність, використання апаратної частини пристрою. Недоліки: складність написання, менша гнучкість і безпека, складність підтримки.

Вибір варіанту цілком залежить від потреб. Якщо ж треба стаціонарність, то варто встановити програму на ПК або використати сайт. В інакшому випадку доцільно використати мобільний застосунок або сайт.

1.4 Вибір компонентів та середовища для реалізації

Ми провели аналіз основних складових систем розумної парковки, що дозволяють проводити автоматичне розпізнавання номерних знаків

транспортних засобів. Було визначено, що основними апаратними компонентами такої парковки можуть бути:

– камера(може бути з можливістю розпізнавання номерних знаків, від цього змінюється програмна реалізація);

– індукційна петля;

– датчики зважування в русі;

– автоматичні шлагбауми та світлофори;

– сервер;

– мережеве обладнання;

– освітлення.

Усі ці компоненти є частиною кіберфізичної системи обмінюються даними завдяки мережі. Камери без вбудованої технології розпізнавання номерних знаків значно дешевше, проте витрачається час на розробку чи використання готового програмного рішення в самій системі, яке виконує цю задачу.

Також було розглянуто частини програмних засобів системи розумної парки, яка розділені на модулі, що виконують різні задачі. Такими модулями є:

– модуль зйомки. Модуль відповідає за одержання фото з камери. Можливе отримання кадрів з відеопотоку із подальшою їх обробкою. Може використовуватись OpenCV або ffMPEG;

– модуль обробки зображення. Тут відбувається первинна обробка зображення, підготовка його для передачі далі. Використовуються будь-які засоби для обробки зображень, наприклад, OpenCV [25];

– модуль розпізнавання номерних знаків. Ключовий модуль. В ньому відбувається аналіз зображення, розпізнавання номерного знаку символів в ньому. Можливі 3 основних формати рішення:

1) використання спеціалізованих наборів, бібліотек або API для розпізнавання номерних знаків. Однією із таких бібліотек може бути OpenALPR. Такі рішення є платними;

2) написання своєї логіки пошуку та розпізнавання номерного знаку. Безкоштовне рішення при використанні загальнодоступних ресурсів, проте потребує часу для написання коду. Може використовуватись бібліотека OpenCV у парі з натренованими моделями, отриманими в результаті навчання нейронної мережі, наприклад, YOLOv8;

3) використання загальних API комп'ютерного зору. Умовно безкоштовне рішення, адже у таких API є ліміт безкоштовного використання перед оплатою. Такими API є Google Cloud Vision API, Microsoft Computer Vision API. Квота розрахована зазвичай на місяць.

Вибір необхідного рішення залежить від потреб по бюджету;

– модуль зберігання даних. Відповідає за зберігання даних на сервері та доступ до них. Може бути встановлена реляційна або нереляційна база даних. Вибір залежить від масштабу систему та даних;

– модуль візуалізації. Відповідає за перегляд даних з системи кінцевим користувачем. Може бути представлений у трьох різних варіантах:

1) веб-інтерфейс. Доступний усюди, де є доступ у внутрішню мережу або Інтернет. Простий у написанні, проте може бути вразливий [26-27]. Можливе використання фреймворків, наприклад: Laravel, Django, Node.JS;

2) застосунок на ПК. Має бути встановлений на ПК. Працює швидше ніж веб-інтерфейс, проте складніше у написанні, безпечніше. Може бути кросплатформним;

3) мобільний застосунок на iOS/Android. Складніше у написанні, ніж застосунок на ПК. Є можливість використовувати камеру та різні датчики пристрою. Може бути написаний на кросплатформних фреймворках React Native, Xamarin або Flutter.

Врахувавши особливості усіх компонентів, їх переваги та недоліки, ми пропонуємо концепцію рішення в якості альтернативи вже відомим:

– модульна система, що складається з апаратної та програмної частини;

– апаратна частина має звичайну камеру, що може отримувати зображення з великим кутом огляду, достатньою роздільною здатністю, опціонально ІЧ-підсвічення. Сервер розташований в хмарі, наприклад, використовуючи послуга хостингу для доступу 24/7, це дешевше, ніж розміщення та налаштування свого серверу;

– програмна частина написана з використанням бібліотеки OpenCV для доступу до відеопотоку з камери, первинної обробки зображення, клієнтська бібліотека Google Cloud Vision API для розпізнавання номерного знаку на фото, мова Python та фреймворк Flask для використання бібліотеки OpenCV, PHP для створення REST API на основі фреймворку Laravel, PHP та Laravel Blade для створення веб-інтерфейсу на основі Laravel, React Native для мобільного застосунку. Для зберігання даних використовується реляційна база даних [28] MySQL.

Перевагами такої системи є:

– масштабованість. Можливості системи можна розширити, додавши функціонал у REST API;

– умовно безкоштовне використання при контролі ліміту безкоштовних сканувань Google Cloud Vision;

– гнучкість. Можна реалізувати додатково до веб-інтерфейсу та мобільного застосунку програму на ПК, адже усі основні дії виконуються на сервері.

Недоліками такої системи є:

– складність розробки та впровадження – розробка такої системи потребує навичок написання свого REST API, роботи з стороннім API, знання різних фреймворків;

– неідеальна точність. Google Cloud Vision не розроблялось для використання саме для розпізнавання номерних знаків, точність може бути не 100%. Проте за результатами досліджень, в більшості випадків використання, Google Cloud Vision API дає цілком чудові результати;

– менша гнучкість в контексті розпізнавання саме номерних знаків. Система зконфігурована на розпізнавання саме українських номерних знаків сучасного формату.

1.5 Висновки

У першому розділі даної роботи було виконано аналіз наявних систем на ринку для автоматизованого розпізнавання номерних знаків і проведено огляд різноманітних рішень, спрямованих на виконання цієї задачі. Було досліджено як програмне, так і апаратне забезпечення, що застосовується у подібних системах, та проведено порівняльний аналіз різних користувацьких додатків, призначених для автоматизації управління парковкою.

В результаті аналізу компонентів системи розумної парковки було запропоновано концепцію власного рішення, яке може стати ефективною альтернативою наявним готовим рішенням, що пропонуються на ринку. Використаний у дипломній роботі підхід спрямований на вдосконалення процесу управління системою автоматизованого розпізнавання номерних знаків, забезпечуючи оптимальне використання ресурсів та більш високий рівень автоматизації для кінцевих користувачів.

2 ОБҐРУНТУВАННЯ ВИБОРУ КОМПОНЕНТІВ ТА СЕРЕДОВИЩА РЕАЛІЗАЦІЇ

2.1 Апаратне середовище реалізації

Система «Розумна парковка» має клієнт-серверну архітектуру з програмної сторони розробки. З апаратної частини це також включає клієнта, сервер, але крім цього – апаратні засоби обробки зовнішньої інформації, такі як камери. Також присутнє мережеве обладнання для підключення.

Сервер повинен мати достатню потужність для обробки простих запитів, мати аптайм, що досягає 100%, мати невисоку ціну встановлення та підтримки. При таких умовах є 3 варіанти встановлення сервера:

– використання власного сервера, що включає його збірку, налаштування та забезпечення постійної роботи;

– використання віртуального shared хостингу;

– використання VPS/виділеного серверу.

Складемо таблицю порівняння усіх 3 варіантів (таблиця 2.1).

Таблиця 2.1 – Порівняння варіантів встановлення серверу

Критерій	Власний сервер	Віртуальний shared хостинг	VPS/Виділений сервер
1	2	3	4
Вартість	Висока(початкові інвестиції в обладнання)	Низька	Середня
Складність	Висока(знання та досвід налаштування)	Низька(такий хостинг потребує мінімального налаштування від користувача)	Середня-висока

Кінець таблиці 2.1

1	2	3	4
Гнучкість	Висока(повний контроль)	Низька	Висока
Масштабованість	Висока(можна легко додати більше ресурсів)	Низька	Висока
Безпека	Низька-висока(все залежить від адміністратора серверу)	Середня	Середня-висока
Підтримка	Низька(самостійне вирішення питань)	Висока	Середня

В перспективі власний сервер або VPS/Виділений сервер буде краще, проте на початку або з невеликим бюджетом кращим варіантом буде віртуальний shared хостинг. Крім цього, якщо віртуальний хостинг має необхідні функції та непогану потужність, це також буде перевагою його використання. У нашому випадку, для роботи з відео, камерами, обробкою HTTP запитів необхідно мати стандартний набір Apache/NGINX з підтримкою Python. Звичайний спільний хостинг під управлінням cPanel та доступом до терміналу має ці функції.

Також ще до розробки системи був оплачений тариф такого хостингу на сайті S-HOST [29] на рік, що також додало ще одну перевагу використання саме цього варіанту (рисунок 2.1). Тариф Small має усі необхідні для нас особливості, безкоштовну реєстрацію домену, SSL сертифікацію для захисту, 5 субдоменів, PHP найновіших версій, SSH доступ. Присутня досвідчена технічна підтримка у робочий час. У частих питаннях зазначено, що аптайм серверів – мінімально 99,95%. Технічні характеристики серверу, що надано не вказано, проте ми можемо дізнатись їх, використовуючи термінал.

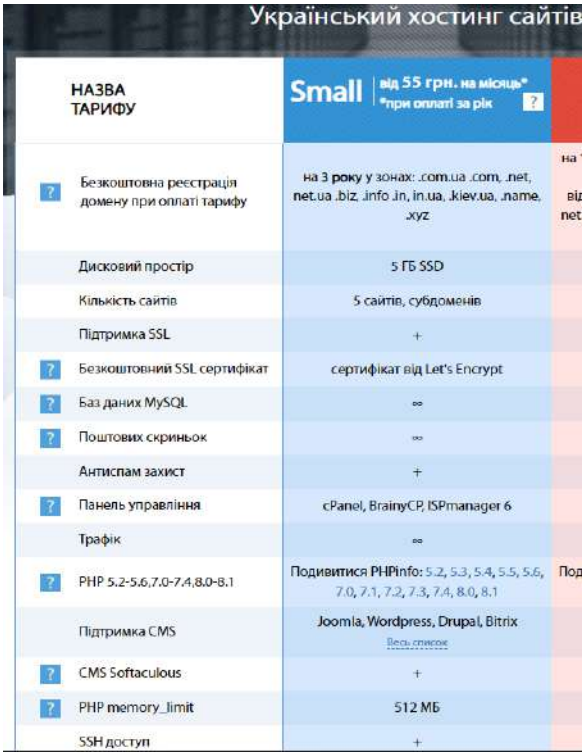


Рисунок 2.1 - Тариф віртуального хостингу

Командами cat /proc/cpuinfo та cat /proc/meminfo дізнаємось характеристики центрального процесору та оперативної пам'яті. Вивід першої команди (рисунок 2.2) свідчить про те, що на борту встановлено 48(нумерація з нуля) процесорів Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz з 12 ядрами, що складає доволі потужний сервер, навіть враховуючи, що він є спільним для багатьох користувачів.

```
processor       : 47
vendor_id       : GenuineIntel
cpu family      : 6
model           : 63
model name      : Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz
stepping        : 2
microcode       : 0x49
cpu MHz         : 1265.087
cache size      : 30720 KB
physical id     : 1
siblings        : 24
core id         : 13
cpu cores       : 12
apicid          : 59
initial apicid  : 59
fpu             : yes
fpu_exception   : yes
cpuid level     : 15
wp              : yes
```

Рисунок 2.2 - Вивід команди cat /proc/cpuinfo

В свою чергу, команда cat /proc/meminfo (рисунок 2.3) виведе нам характеристики оперативної пам'яті серверу. Число MemTotal покаже нам число кілобайт пам'яті, встановленої на комп'ютер, що становить 197790880 кБ = 188,62 гБ, що є дуже непоганим показником.

```
MemTotal:       197790880 kB
MemFree:         7938896 kB
MemAvailable:   80325396 kB
Buffers:         4636092 kB
Cached:         76805852 kB
SwapCached:        55484 kB
Active:         122366984 kB
Inactive:       52599252 kB
Active(anon):   92826260 kB
Inactive(anon):  5466208 kB
```

Рисунок 2.3 - Вивід команди cat /proc/meminfo

Сервер є доволі потужним, має усі необхідні характеристики, панель адміністрування cPanel, створення баз даних MySQL, PostgreSQL, можливість створення до 5 субдоменів, Python WSGI застосунки для обробки засобів Python фреймворками, тому зупинимось на цьому варіанті.

Наступною задачею буде вибір камери. Як вже згадувалось раніше, в вимогах може не стояти розпізнавання номерів, якщо це забезпечується програмною частиною. Так як в роботі розпізнавання номеру реалізується програмно, тоді можемо вибрати камеру без цієї можливості. Звернемось до сайту постачальнику IP камер Hikvision [30], який є дуже популярним і часто використовується для парковок (рисунок 2.4). Оберемо у фільтрі тип встановлення – вулична та тип підключення – PoE, встановимо сортування по зростанню ціни.

Виберемо камеру Hikvision DS-2CD1021-I (4 мм). Вона має такі характеристики:

– форм фактор – циліндрична;
– особливості – ІЧ-підсвічування;

– роздільна здатність – 2 МП;
– живлення – DC 12В;
– клас захисту – IP67;
– матриця – 1/2.8 Progressive Scan CMOS;
– встановлення – вулична, внутрішня.

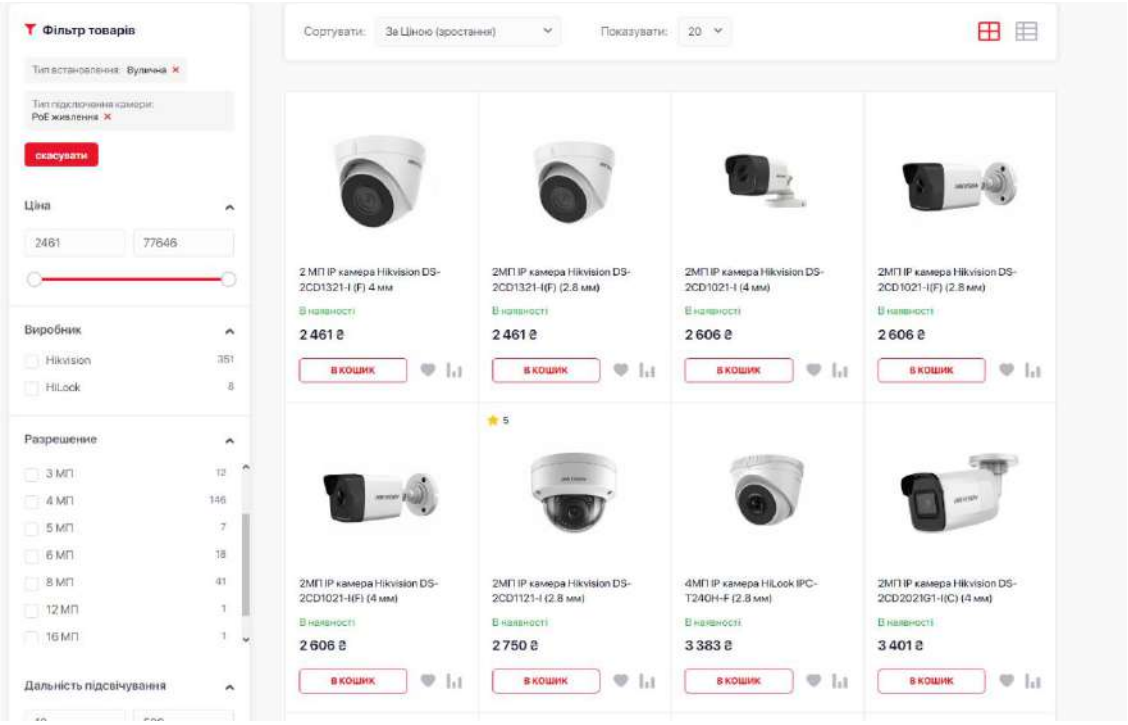


Рисунок 2.4 - Камери Hikvision

Ця камера є доволі бюджетною, має достатні характеристики для чіткого фото, що нам і потрібно. Також необхідно взяти PoE інжектор для більш гнучкого підключення камери. Для під'єднання усіх елементів парковки у мережу необхідно ще купити PoE комутатор TP-Link TL-SG1005LP.

Останнім апаратнім засобом є клієнт. Застосунок буде розроблений на усі платформи, тому необхідно описати вимоги для них всіх.

Клієнт, що працює за ПК має мати можливість виходити у мережу Інтернет через веб-браузер. Сформуємо вимоги такого комп'ютера на 2024 рік:

– процесор: AMD Ryzen 3 3200 G 3.6 ГГц / Intel Core i3-1005G1 3.1 ГГц;
– материнська плата ASRock B450M Pro4 / ASRock B450M Pro4;

– оперативна пам’ять: 8 ГБ DDR4;
– вбудована графіка: AMD Radeon Vega 8 / Intel UHD Graphics 630;
– SSD 256 GB;
– Блок живлення Corsair CV450 80+ Bronze Certified Semi-modular ATX.

Також можна вибрати ноутбук Lenovo IdeaPad 3 15ALC6 (82KU0243RA) Arctic Grey / AMD Ryzen 5 5500U / RAM 8 ГБ / SSD 256 ГБ.

Мобільні клієнти поділяються на два типи: користувачі з ОС iOS та з ОС Android.

Для мобільних пристроїв не так сильно окреслюються апаратні вимоги для запуску застосунків. В нашому випадку, планується використання React Native для написання коду, який буде успішно виконуватись як на Android, так і на iOS, тому мова йде більше про програмні вимоги до ОС(Android 4.1(API 16) [31] iOS 13.4 відповідно [32]).

Наш застосунок не вимагає графічних обчислень, а лише передачу HTTP запиту з даними, які можуть картинкою або посиланням. Тому, мінімальні апаратні вимоги для пристроїв можуть бути визначені так:

– процесор з частотою 1.2ГГц і більше;
– ОЗП об’ємом 1 ГБ, рекомендовано 2 ГБ;
– внутрішня пам’ять: 50 МБ для встановлення застосунку;
– розширення екрану 480*800 пікселів, розмір 4.7 дюйми мінімум;
– підключення до безпровідного інтернету або стільникові дані.

2.2 Функційні вимоги

Функційні вимоги – вимоги, які описують роботу системи, те як вона маніпулює даними, оброблює їх, фактично визначають, що система повинна роботи. Ці вимоги визначаються на першій стадії розробки програмного засобу.

Розглянемо функціонал, який має забезпечити кіберфізична система «Розумна парковка» з розпізнаванням номеру і знаку.

На рисунку 2.5 (додаток В) показано шлях запиту клієнта до кіберфізичної системи. Система обробляє HTTP запити, що поступають від клієнтської частини, якою може бути веб-сайт або мобільний додаток на iOS/Android.

Серверна частина складається з 2 умовних частин – PHP ANPR REST API, яка реалізовано за допомогою фреймворку Laravel, та Python OpenCV Image Getting API, яка реалізована за допомогою фреймворку Flask та бібліотеки OpenCV. Сервер, в свою чергу відправляє запит до сервісу Google Cloud Vision, використовуючи API від Google. Отримані дані формуються у відповідь на запит та віддаються на клієнтську частину, яка отримує цей результат і користувач отримує дані про номерний знак і модель авто, зображеного на фото.

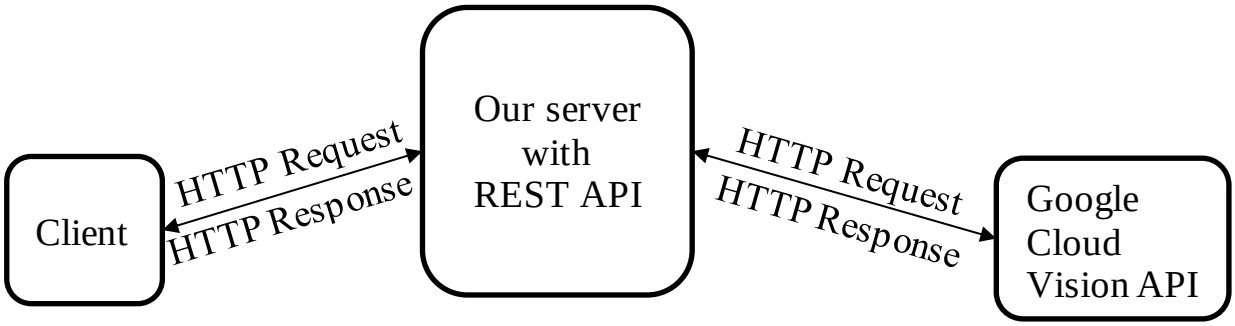


Рисунок 2.5 – Схема шляху запиту від клієнта до кіберфізичної системи «Розумна парковка» з розпізнаванням номерного знаку і моделі

Визначивши архітектуру програмної частини, можна сформулювати вимоги щодо функціональності:

– отримання фото шляхом HTTP запиту або з камери по RTSP протоколу;

– аналіз фото та визначення номерного знаку та моделі авто в Україні;

– інтерфейс клієнтської частини, що забезпечує роботу з функціями, які забезпечує сервер.

2.3 Нефункційні вимоги

Нефункційні вимоги – це ті вимоги, які описують якість роботи системи при забезпеченні нею потрібних функцій, на відміну від функційних вимог. Вони визначаються на тому ж етапі проєктування, що і функційні вимоги.

Серед цих вимог можна виділити наступні:

1. Швидкодія. Система не має бути надто повільною. Прийнятним часовим проміжком на аналіз фото буде 2 секунди для розпізнавання номерного знаку та 5 секунд на розпізнавання моделі.

2. Доступність. Система повинна бути доступна 24/7. Допустимий простій – 1% на рік. Хостинг забезпечує 99,95% доступності, тому можна очікувати 99% доступності системи, що працює на серверах цього хостингу.

3. Надійність і безпека. Система повинна бути стійкою до збоїв, відповідати вимогам кібербезпеки.

4. Зручність використання. Інтерфейси мобільного застосунку і веб-застосунку мають бути простими і інтуїтивними, не навантаженими зайвими елементами.

5. Масштабованість. Система повинна мати можливість розширення для збільшення потужності або кількості функцій, які вона виконує.

6. Сумісність. Система повинна бути сумісною з різними видами обладнання, в нашому випадку, камерами.

7. Естетичність. Інтерфейс повинен мати прийнятний вигляд з точки зору дизайну.

2.4 Вибір методів та середовища для реалізації програмного забезпечення

Для роботи з зображеннями з камери зазвичай використовується бібліотека OpenCV (рисунок 2.6) [33], а саме робота з цією бібліотекою, використовуючи мову програмування Python. Для роботи з Python хорошим вибором є середовище PyCharm від JetBrains. Його звичайна версія є безкоштовною та

достатньо потужною. Вона забезпечує автоматичне доповнення коду, підсвічування синтаксису, рефакторинг коду, вбудований дебагінг програми, інструменти тестування, підтримка систем контролю версій, робота з базами даних та приємний інтерфейс.

Рисунок 2.6 – Логотип OpenCV

OpenCV (Open Source Computer Vision Library) - це бібліотека програмного забезпечення для комп'ютерного зору та машинного навчання з відкритим вихідним кодом. OpenCV [34] була створена, щоб забезпечити загальну інфраструктуру для додатків комп'ютерного зору і прискорити використання машинного сприйняття в комерційних продуктах. Будучи продуктом з ліцензією Apache 2, OpenCV дозволяє компаніям легко використовувати та модифікувати код.

Бібліотека налічує понад 2500 оптимізованих алгоритмів, що включає в себе повний набір як класичних, так і сучасних алгоритмів комп'ютерного зору та машинного навчання. Ці алгоритми можна використовувати для виявлення та розпізнавання облич, ідентифікації об'єктів, класифікації дій людини на відео, відстеження руху камери, відстеження рухомих об'єктів, вилучення 3D-моделей об'єктів, створення 3D-хмар точок зі стереокамер, зшивання зображень для отримання зображення всієї сцени з високою роздільною здатністю, пошуку схожих зображень у базі даних зображень, видалення червоних очей зі знімків, зроблених зі спалахом, відстеження рухів очей, розпізнавання пейзажу та встановлення маркерів для накладання на нього доповненої реальності, тощо.

Спільнота користувачів OpenCV налічує понад 47 тисяч осіб, а кількість завантажень перевищує 18 мільйонів. Бібліотека широко використовується в компаніях, дослідницьких групах та урядових установах.

Бібліотека має інтерфейси C++, Python, Java та MATLAB і підтримує Windows, Linux, Android та Mac OS. OpenCV здебільшого орієнтована на додатки технічного зору в реальному часі і використовує переваги інструкцій MMX та SSE, коли вони доступні. Зараз активно розробляються повнофункціональні інтерфейси CUDA та OpenCL. Існує понад 500 алгоритмів і приблизно в 10 разів більше функцій, які складають або підтримують ці алгоритми. OpenCV написана на мові C++ і має шаблонний інтерфейс, який без проблем працює з контейнерами STL.

OpenCV-Python – це бібліотека для Python, яка виконує функцію прив'язки Python до C++. Вона розроблена для спрощення роботи з алгоритмами комп'ютерного зору. На Python код писати простіше, ніж на C++. Проте, Python виконується повільніше, ніж C++, адже це інтерпретована мова. Python розширюється кодом C++ для збільшення його можливостей. Ця бібліотека використовує Numpy, бібліотеку, що оптимізована для числових операцій над великими масивами даних. Більшість операцій OpenCV, які повертають масиви даних, повертають ці дані у вигляді масиву Numpy.

Для роботи з HTTP запитами на Python, використаємо фреймворк Flask. Flask - це веб-фреймворк, модуль Python, який дозволяє легко розробляти веб-додатки. Він має невелике ядро, яке легко розширюється: це мікрофреймворк, який не містить ORM (Object Relational Manager) або подібних функцій.

Він має багато потужних функцій, таких як маршрутизація URL-адрес, шаблонний движок. Це фреймворк для веб-додатків WSGI.

Фреймворк веб-додатків або просто веб-фреймворк - це набір бібліотек і модулів, які дозволяють розробникам веб-додатків писати програми, не турбуючись про низькорівневі деталі, такі як протокол, управління потоками і так далі.

Flask [35] - це фреймворк для веб-додатків, написаний на Python. Його розробив Армін Ронахер, який очолював команду міжнародних ентузіастів Python під назвою Pooccо. Flask базується на інструментарії Werkzeug WSGI та шаблонізаторі Jinja2, які є проектами Poccо.

Інтерфейс шлюзу веб-сервера (Web Server Gateway Interface, WSGI) використовується як стандарт для розробки веб-додатків на Python. WSGI - це специфікація загального інтерфейсу між веб-серверами та веб-додатками.

Werkzeug - це інструментарій WSGI, який реалізує запити, об'єкти-відповіді та утиліти. Це дозволяє будувати на ньому веб-фреймворк. Фреймворк Flask використовує Werkzeug як одну зі своїх основ.

jinja2 - популярний движок шаблонів для Python. Система веб-шаблонів поєднує шаблон з певним джерелом даних для відображення динамічної веб-сторінки.

Flask часто називають мікрофреймворком. Він розроблений для того, щоб ядро програми було простим і масштабованим.

Замість шару абстракції для підтримки баз даних, Flask підтримує розширення для додавання таких можливостей до програми.

На відміну від фреймворку Django, Flask дуже схожий на Pythonic. Почати роботу з Flask легко, тому що він не має величезної кривої навчання.

Крім того, він дуже зрозумілий, що підвищує читабельність. Щоб створити додаток "Hello World", вам знадобиться лише кілька рядків коду.

Шаблонний код для цього виглядає так:

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello World!'

if __name__ == '__main__':
    app.run()
```

Він визначає кінцеву точку (шлях “/”), тобто корінь. Цей код повертає стрічку “Hello world”, яка повертається як результат HTTP запиту. Код запускається тільки якщо виконується самостійно, а не імпортується як модуль. За замовчуванням працює на порту 5000. При використанні cPanel для керування хостингом, можна налаштувати Python застосунок і при цьому хостинг автоматично налаштовує Apache/Nginx для перенаправлення 5000 порту на 80/443 (порти HTTP/HTTPS.)

Для написання власного REST API використаємо мову PHP та фреймворк Laravel (рисунок 2.7). Laravel - це безкоштовний PHP-фреймворк з відкритим вихідним кодом. Він надає веб-розробникам інструменти та ресурси для створення сучасних веб-додатків на PHP.

Laravel має корисні вбудовані функції, такі як інтерфейс командного рядка (CLI) Artisan [36], вбудована автентифікація та архітектура модель-вид-контролер (MVC). Ці функції роблять фреймворк простим у використанні і є основною причиною його популярності.

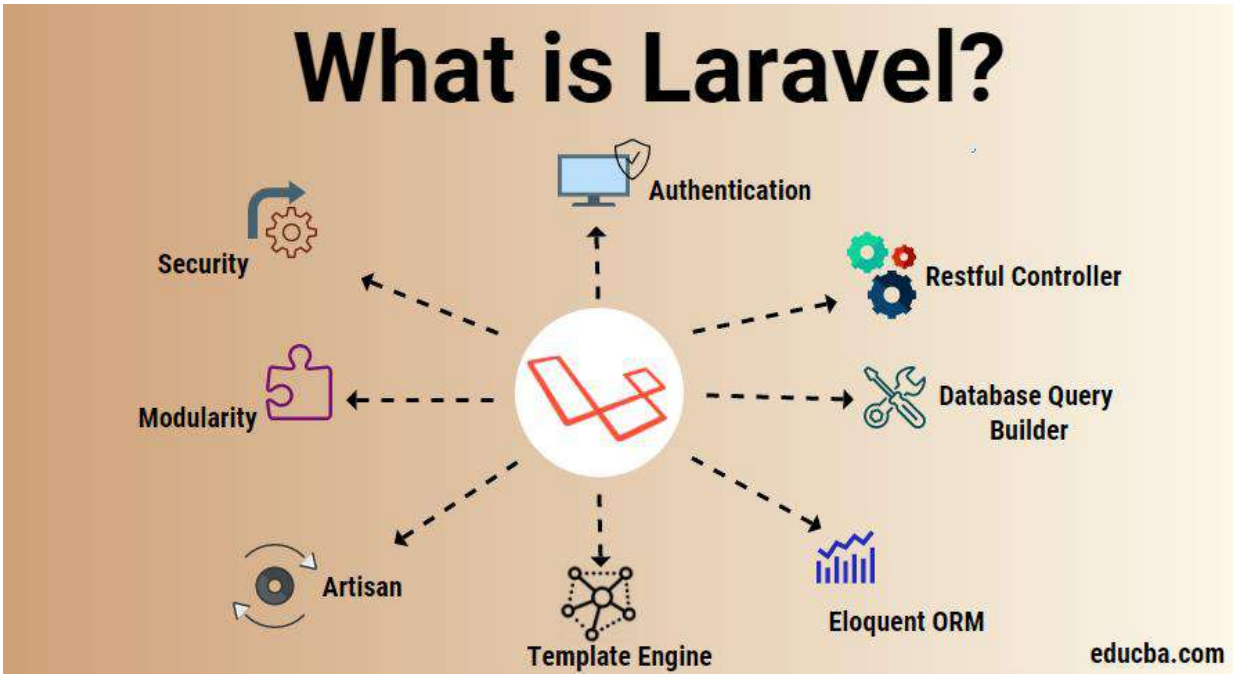


Рисунок 2.7 – Складові Laravel

Laravel – це трендовий веб-фреймворк. Він має понад 70 000 зірок на GitHub [37] і став затребуваною навичкою на ринку праці.

Laravel – це простий у використанні веб-фреймворк, який допоможе створювати розширювані веб-сайти та веб-додатки на основі PHP у великих масштабах [38].

Перш ніж створювати веб-додаток або веб-сайт, потрібно прийняти фундаментальне рішення про те, яка технологія буде використовуватись. Це одна з найскладніших частин процесу веб-розробки.

Щоб створити щось просте, наприклад, інтернет-магазин або портфоліо, можна покластися на дизайнерів веб-сайтів без коду. Якщо треба створити щось більш просунуте, рішення без коду може бути недостатньо. Натомість варто обрати фреймворк і почати писати код на ньому. Laravel – хороший вибір як простий у використанні фреймворк з відкритим вихідним кодом для створення сучасних веб-додатків у великих масштабах.

Переваги Laravel:

– архітектура MVC.

Фреймворк Laravel підтримує архітектурний патерн MVC для створення веб-додатків. Цей патерн визначає набір правил, які вказують, як створювати масштабовані та підтримувані веб-додатки.

Шаблон Laravel MVC допомагає розробникам впорядковувати та узгоджувати неструктурований код. Підхід MVC також полегшує розробку як малих, так і великих веб-додатків;

– інтерфейс командної стрічки Artisan.

Laravel використовує Artisan як CLI, що допомагає веб-розробникам мігрувати дані, керувати базами даних, створювати шаблонний код, контролери, моделі та багато іншого.

Artisan CLI полегшує веб-розробку завдяки функціям генерації коду та управління базами даних, які доступні лише кількома командами. Замість того,

щоб писати шаблонний код або налаштовувати базу даних, розробник може зосередитися на побудові логіки програми;

– вбудована автентифікація.

Laravel постачається з готовими рішеннями для автентифікації та авторизації. За допомогою декількох команд Artisan можна налаштувати надійну та надійну автентифікацію та авторизацію для веб-додатку;

– движок шаблонів Blade.

Blade Template Engine - це потужний інструмент, який постачається з Laravel. Blade використовується для:

1) підключення моделей даних;

2) обробки код програми всередині тегів шаблону;

3) перенаправлення виводу у текстовий файл або інші потоки.

Завдяки цим функціям Blade робить розробку швидшою та зручнішою;

– Eloquent ORM.

Об'єктно-реляційний маппер (ORM) Eloquent - це інструмент управління базами даних, який входить до складу фреймворку Laravel.

Eloquent дозволяє легко взаємодіяти з базою даних. З Eloquent кожна таблиця бази даних має відповідну модель, яку ви можете використовувати для взаємодії з таблицею. Окрім отримання записів з таблиці, модель Eloquent дозволяє вставляти, оновлювати та видаляти записи;

– Redis.

Redis - це сховище ключ-значення, яке також називають сервером структур даних, оскільки воно підтримує такі типи даних, як рядки, хеші, списки, множини та відсортовані множини.

Laravel підтримує використання Redis як кешу для тимчасового зберігання даних. Кешування даних в Redis прискорює запити до бази даних і зворотній зв'язок, що робить пошук даних швидшим. Це, в свою чергу, прискорює роботу ваших веб-додатків.

Як вже згадано вище, цей фреймворк використовується для написання нашого REST API. API [39] - це механізми, які дають змогу двом програмним компонентам взаємодіяти один з одним, використовуючи набір визначень і протоколів. Наприклад, система ПЗ метеослужби містить щоденні дані про погоду. Додаток погоди на телефоні "спілкується" з цією системою через API і показує щоденні оновлення погоди на телефоні.

API - Application Programming Interface, що означає програмний інтерфейс застосунку. У контексті API слово "застосунок" відноситься до будь-якого ПЗ з певною функцією. Інтерфейс можна розглядати як сервісний контракт між двома додатками. Цей контракт визначає, як вони взаємодіють один з одним, використовуючи запити та відповіді. Документація API містить інформацію про те, як розробники повинні структурувати ці запити та відповіді.

Архітектуру API [40] зазвичай пояснюють з погляду клієнта і сервера. Застосунок, що надсилає запит, називається клієнтом, а застосунок, що надсилає відповідь, називається сервером. Отже, у прикладі з погодою база даних служби – це сервер, а мобільний додаток – це клієнт.

Існує чотири різні способи роботи API залежно від того, коли і чому їх було створено:

– SOAP API.

SOAP - Simple Object Access Protocol, тобто простий протокол доступу до об'єктів. Клієнт і сервер обмінюються повідомленнями за допомогою XML. Це менш гнучкий API, який був більш популярний у минулому;

– RPC API.

Такі API називаються системою віддаленого виклику процедур. Клієнт виконує функцію (або процедуру) на сервері, і сервер відправляє результат назад клієнту;

– Websocket API.

Websocket API - це ще одна сучасна розробка web API, яка використовує об'єкти JSON для передачі даних. WebSocket API підтримує двосторонній зв'язок

між клієнтськими додатками та сервером. Сервер може надсилати повідомлення зворотного виклику підключеним клієнтам, що робить його ефективнішим, ніж REST API;

– REST API.

На сьогоднішній день це найпопулярніші та найгнучкіші API-інтерфейси в Інтернеті. Клієнт надсилає запити на сервер у вигляді даних. Сервер використовує це клієнтське введення для запуску внутрішніх функцій і повертає вихідні дані назад клієнту. Розглянемо API REST детальніше нижче.

REST - це Representational State Transfer, тобто передача репрезентативного стану. REST визначає набір функцій, таких як GET, PUT, DELETE тощо, які клієнти можуть використовувати для доступу до даних сервера. Клієнти та сервери обмінюються даними за протоколом HTTP.

Головною особливістю REST API є те, що така передача виконується без збереження стану. Без збереження стану означає, що сервери не зберігають клієнтські дані між запитами. Клієнтські запити до сервера аналогічні URL-адресам, які ви вводите в браузері для відвідування веб-сайту. Відповідь від сервера являє собою прості дані без типового графічного відображення веб-сторінки.

Web API або Web Service API - це інтерфейс обробки додатків між веб-сервером і веб-браузером. Усі веб-сервіси є API, але не всі API є веб-сервісами. REST API - це особливий тип Web API, у якому використовується стандартний архітектурний стиль, описаний вище.

Різні терміни, які відносяться до API, такі як Java API або сервісні API, існують тому, що історично API були створені до всесвітньої павутини. Сучасні web API - це REST API, і ці терміни можуть використовуватися взаємозамінно.

Власне, одним із прикладів API є Google Cloud Vision API (рисунок 2.8), який дозволяє нам використовувати можливості Google Cloud Vision [41] у своїй роботі. API реалізоване для багатьох мов, клієнтська частина представлена у

вигляді бібліотеки, в якої налаштовується параметр ключа, через який ми отримуємо доступ до API в рамках нашого проєкту.

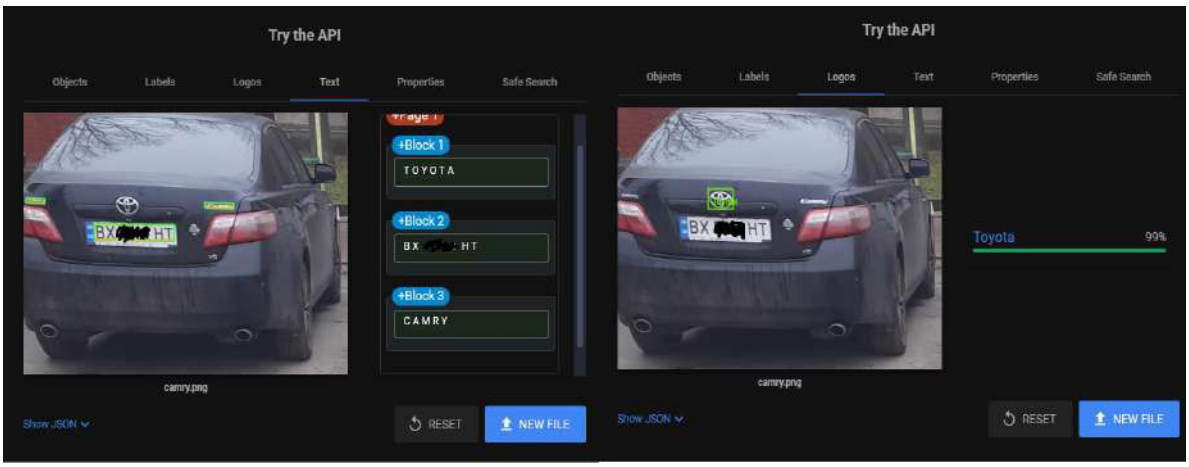


Рисунок 2.8 – Результати аналізу фото автомобіля

Останнім засобом програмування, який ми використаємо є фреймворк React Native [42].

React Native (також відомий як RN) - це популярний фреймворк мобільних застосунків на основі JavaScript, який дозволяє створювати мобільні додатки з нативною візуалізацією для iOS та Android. Фреймворк дозволяє створювати застосунки для різних платформ, використовуючи ту саму кодову базу.

React Native був вперше випущений Facebook [43] як проект з відкритим вихідним кодом у 2015 році. Всього за пару років він став одним з найкращих рішень для мобільної розробки.

Розробка на React Native лежить в основі деяких провідних світових мобільних застосунків, включаючи Instagram, Facebook та Skype.

Існує кілька причин глобального успіху React Native.

По-перше, використовуючи React Native, компанії можуть створити код лише один раз і використовувати його для своїх застосунки для iOS та Android. Це означає величезну економію часу та ресурсів.

По-друге, React Native створено на основі React - бібліотеки JavaScript, яка вже була надзвичайно популярною, коли був випущений мобільний фреймворк.

По-третє, фреймворк надав можливість фронтенд-розробникам, які раніше могли працювати лише з веб-технологіями, створювати надійні, готові до виробництва застосунки для мобільних платформ.

Можна сплутати React з React Native. Проте, це різні речі. React (також відомий як ReactJS) - це бібліотека JavaScript, яка використовується для створення інтерфейсу веб-сайту. Як і React Native, вона також була розроблена командою розробників Facebook.

Тим часом, React Native, який працює на основі React, дозволяє розробникам використовувати набір компонентів інтерфейсу користувача для швидкої компіляції та запуску додатків для iOS та Android.

І React, і React Native використовують суміш JavaScript та спеціальної мови розмітки JSX. Однак синтаксис, який використовується для рендерингу елементів у JSX-компонентах, відрізняється між React та React Native додатками. Крім того, React використовує деякі HTML та CSS, тоді як React Native дозволяє використовувати нативні елементи мобільного інтерфейсу користувача.

Хоча React Native і може використовуватись для роботи з веб-застосунком, проте було вирішено розділити програмний код для мобільного застосунку та для веб-застосунку, використавши різні програмні засоби.

Для роботи з PHP та JavaScript кодом для React Native використаємо середовище розробки JetBrains PhpStorm (рисунок 2.8).

Це середовище не є безкоштовним, проте надає ліцензію для студентів вищих навчальних закладів, що співпрацюють з компанією. Як і PyCharm, це IDE є дуже потужним, в даному випадку більше пристосоване для роботи з веб-стеком (HTML, CSS, JS, PHP). Серед особливостей можна виділити вбудовану підтримку різних фреймворків, тестування коду, асистент зі штучним інтелектом, встановлення різних плагінів.

Рисунок 2.8 – Логотип середовища розробки JetBrains PhpStorm

2.5 Висновки

Отже, було визначено апаратні та програмні засоби реалізації необхідної кіберфізичної системи, окреслено її функційні і нефункційні вимоги.

У ході дослідження ми визначили, що з апаратної частини для реалізації кіберфізичної системи «Розумна парковка» з розпізнаванням номерного знаку і моделі авто необхідно мати сервер, камеру, а також клієнта. Було визначено мінімальні вимоги до цих апаратних засобів або описано характеристики вже наявних.

Програмна частина була розділена на декілька складових, враховуючи запропоновану нами архітектуру для неї. Такими частинами стали:

1) клієнт, що представляє собою веб-застосунок або мобільний застосунок;

2) сервер, що розміщений на хостингу, який в свою чергу поділяється на дві підчастини:

а) Laravel REST API. Прикладний інтерфейс застосунку, що обробляє запити від клієнту та повертає на них відповідь. Запити містять у собі фото, яке необхідно проаналізувати або шлях до камери, звідки необхідно отримати фото;

б) Python OpenCV Image Getting API. Ця частина не обробляє запити клієнта напряму, а лише локальні запити від Laravel REST API. Вона отримує фото з камери, або обробляє вже наявні.

3) Google Cloud Vision API. Сторонній прикладний інтерфейс застосунку, який дозволяє використовувати можливості комп’ютерного зору, що надає компанія Google в рамках тарифу.

3 РЕАЛІЗАЦІЯ ПІДСИСТЕМИ РОЗПІЗНАВАННЯ НОМЕРНИХ ЗНАКІВ ТА МОДЕЛЕЙ АВТОМОБІЛІВ У КІБЕРФІЗИЧНІЙ СИСТЕМІ «РОЗУМНА ПАРКОВКА»

3.1 Принцип роботи підсистеми розпізнавання номерних знаків та моделей автомобілів у кіберфізичній системі “Розумна парковка”

На рисунку 3.1 зображено схему архітектури програмного забезпечення розробленої кіберфізичної системи (додаток В).

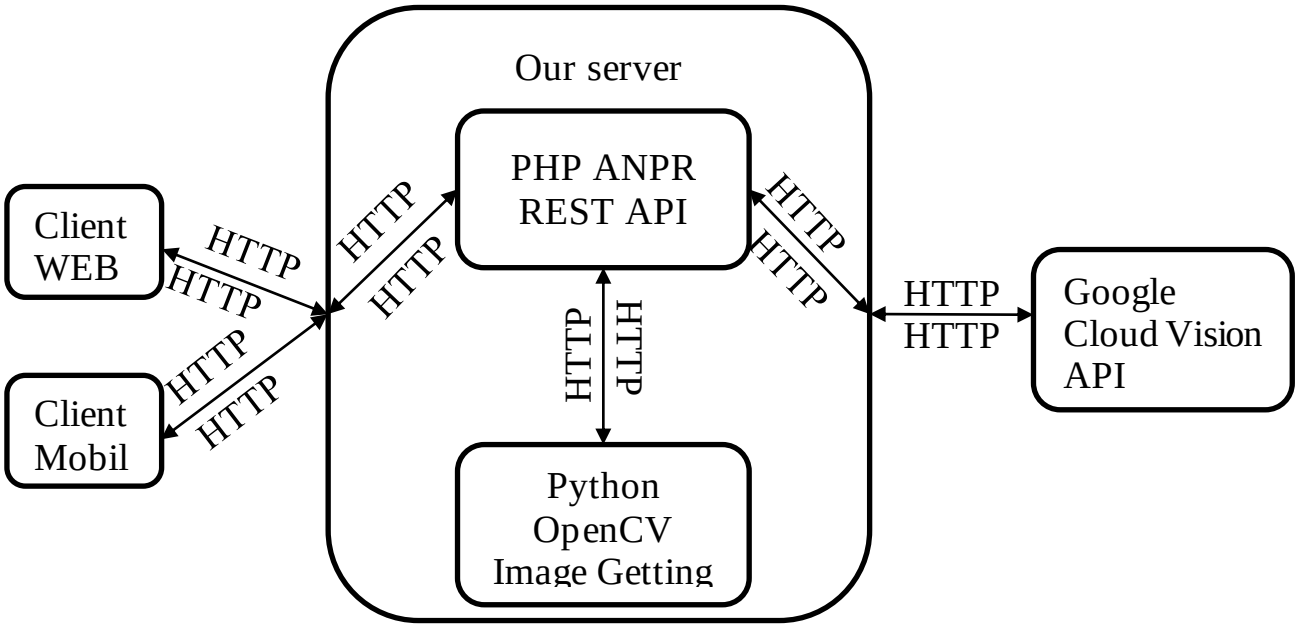


Рисунок 3.1 – Архітектура програмного забезпечення кіберфізичної системи «Розумна парковка» з розпізнаванням номерного знаку і моделі

На схемі видно, що система складається з декількох частин [44]:

– клієнтська частина, якою є веб-сторінка з можливістю завантаження файлу для розпізнавання номерних знаків або застосунок на Android/iOS з аналогічними можливостями. Клієнт отримує відповідь на HTTP POST запит у вигляді JSON об’єкту;

– PHP ANPR REST API – власний прикладний програмний інтерфейс, який розроблений з врахуванням існування відокремленої клієнтської частини, який обробляє GET/POST запити з JSON payload та повертає у відповідь JSON

об’єкт. Цей прикладний інтерфейс дозволяє отримувати дані про результати розпізнавання об’єктів на фото з Google Cloud Vision;

– Python OpenCV Image Getting API – власний прикладний інтерфейс, який дозволяє отримувати одне фото з камери або відео, закодоване у форматі Base64. Відповідь повертається у форматі JSON;

– Google Cloud Vision API – сторонній прикладний програмний інтерфейс, який дозволяє взаємодіяти з системою комп’ютерного зору Google Cloud Vision. Є багато варіантів взаємодії з цим інтерфейсом, але найпопулярнішими є два: використання REST API та використання клієнтської бібліотеки. При використанні REST API дані приймаються та передаються у форматі JSON. Клієнтські бібліотеки розроблені, враховуючи особливості мови та використовуючи можливості ООП. Хоча створення запитів простіше з використанням REST API, було вирішено використати клієнтську бібліотеку через її більшу ефективність та описану документацію.

До написання коду було проведено аналіз результатів, отриманих з Google Cloud Vision API [45], використовуючи демонстраційний варіант (рисунок 3.2). В ньому можна завантажити фото та отримати результат розпізнавання онлайн.

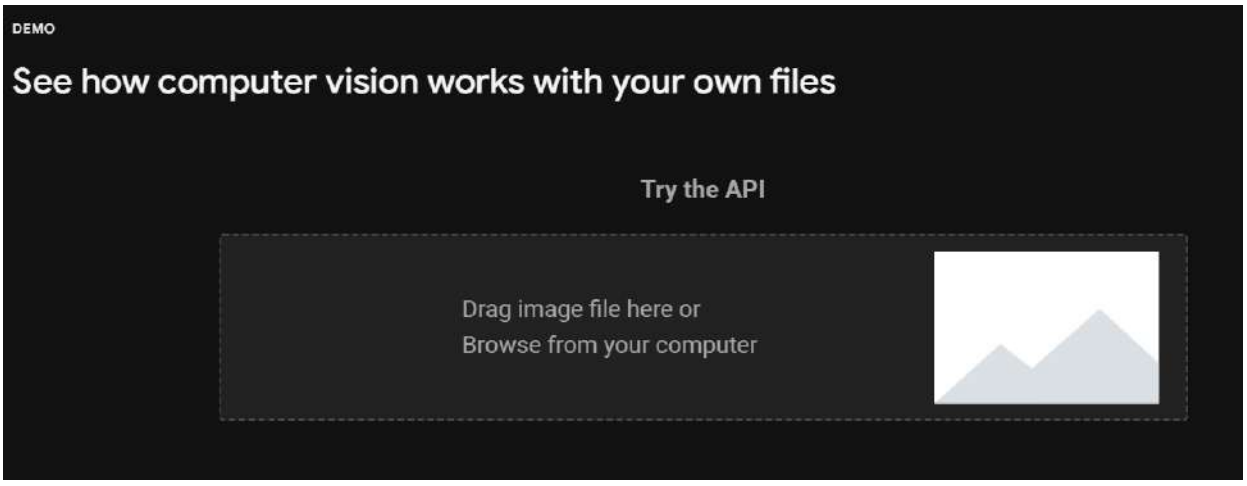


Рисунок 3.2 – Демонстрація роботи Google Cloud Vision API

Під час проведення тестувань розпізнавання [46] було визначено, що Google Cloud Vision має похибки при розпізнаванні номерних знаків. Їх можна описати наступним чином:

1) неправильне розпізнавання символів номерів та плутання їх з цифрами. Наприклад, ОО на початку стає 00(нуль-нуль);

2) відсутні об'єкти типу «машина», проте є «номерний знак»;

3) немає ні «номерних знаків», ні «машин», проте є текст, який може бути номерним знаком;

4) розпізнавання номерного знаку як англійські букви, а не українські;

5) розпізнаний текст у форматі «речень» поданий повністю без розуміння системою, в якій позиції він розпізнаний;

6) абсолютно невірне розпізнавання символу.

Дослідження проводились на датасеті фотографій [47-50] під різним кутом, освітленням, якістю, проте були помічені закономірності, завдяки яким були знайдені рішення проблем розпізнавання, що виникли:

1) таких пар символів є дві – 1 (один) – I, 0 (нуль) – O. Якщо послідовність символів може бути номером авто, тоді можна однозначно перетворити символи цифр на букви. Перевірка стрічки може відбуватись, наприклад, за допомогою регулярного виразу (2 букви/цифри, 4 цифри, 2 букви/цифри);

2) можна створити фіктивний об'єкт машини в системі, який буде містити цей номерний знак, як одну з властивостей. Це дозволить абстрагувати систему від різних граничних випадків;

3) ця проблема вирішується аналогічно другій створенням фіктивного об'єкту;

4) українські номери зазвичай представлені символами, які є в латиниці, і в кирилиці, тому можна за допомогою словника перетворити ідентичні англійські букви на українські. Якщо ж номер буде мати букви, які є лише в українській мові, то Google Cloud Vision розпізнає його українською;

5) таке подання тексту є зручним, бо воно об'єднує слова в залежності від їх розташування і практично завжди містить номерний знак, якщо він справді є. Знаючи слова та їх окреме положення на картинці, можна перебрати послідовність слів, утворивши речення та визначити положення цього речення;

6) така проблема виникає зазвичай при поганій якості фото, великому куту зйомки номера, або через його забруднення. Усі ці проблеми можна вирішити апаратним чином, тому немає сенсу загострювати на них увагу.

Варто зазначити, що більшість сутностей, розпізнаних комп'ютерних зором мають свій обмежуючий чотирикутник, який повертається у відповіді (рисунок 3.3).

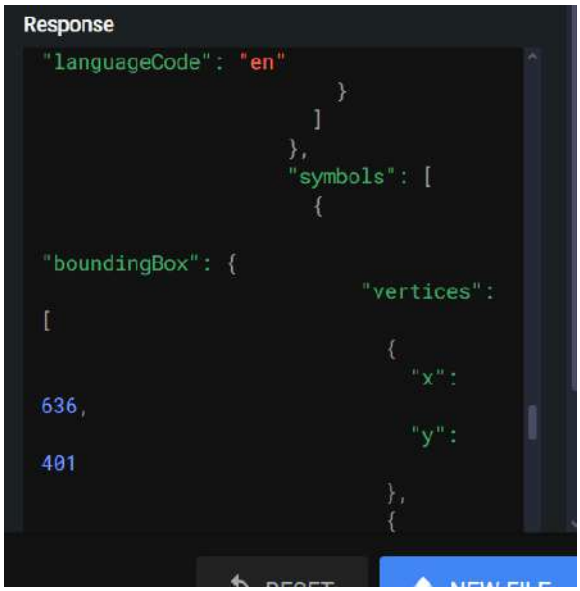


Рисунок 3.3 – Об'єкт тексту, представлений з вершинами

Ці вершини вказують 4 точки чотирикутника, які ми можемо використати для локації сутності на фото. Наприклад, ми можемо визначити чи розпізнаний номерний знак є частиною машини, чи є текст частиною номерного знаку. Для визначення чи лежить один чотирикутник всередині іншого необхідно використати загальне рішення, використовуючи аналітичну геометрію, а саме псевдоскалярний добуток векторів для перевірки перетину ребер чотирикутників. Завдяки цьому ми зможемо виділити об'єкт типу «машина» та

зібрати у ньому усі сутності, що були розпізнані (номерний знак та текст). Також ця задача дуже важлива для відокремлення тексту на самому номерному знаку, для пошуку його серед розпізнаних «речень», які не мають обмежуючого чотирикутника.

Після визначення машин та сутностей в них, ми можемо остаточно визначити номерний знак, що був розпізнаний.

3.2 Структурна схема та алгоритм роботи підсистеми розпізнавання номерних знаків та моделей автомобілів у кіберфізичній системі «Розумна парковка»

Наша кіберфізична система «Розумна парковка» складається з 3 частин: камера для розпізнавання номерного знаку при в'їзді машини, серверна частина (ПЗ) та клієнтської частини (телефони та веб-браузери). Структурна схема системи представлена на рисунку 3.4 (додаток А).

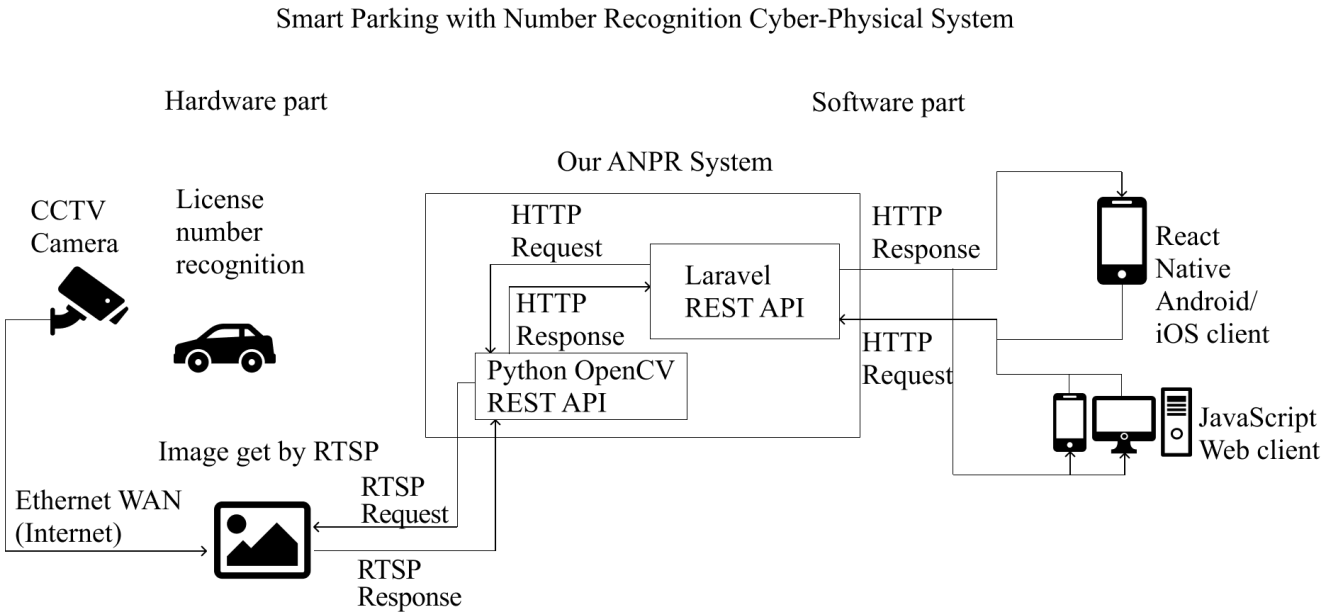


Рисунок 3.4 – Структурна схема кіберфізичної системи «Розумна паркова з розпізнаванням номерного знаку»

Схема показує взаємодію між компонентами системи:

1) отримання зображення у реальному часі з камери відеоспостереження, використовуючи Real-Time Streaming Protocol;

2) запит цього зображення бібліотекою OpenCV у Python REST API;

3) отримання зображення з камери через HTTP запит до Python REST API в Laravel REST API;

4) отримання розпізнаних знаків у клієнтській частині через HTTP запит до Laravel REST API.

Алгоритм роботи ПЗ(клієнтська і серверна частина разом) представлений на рисунку 3.5 (додаток А).

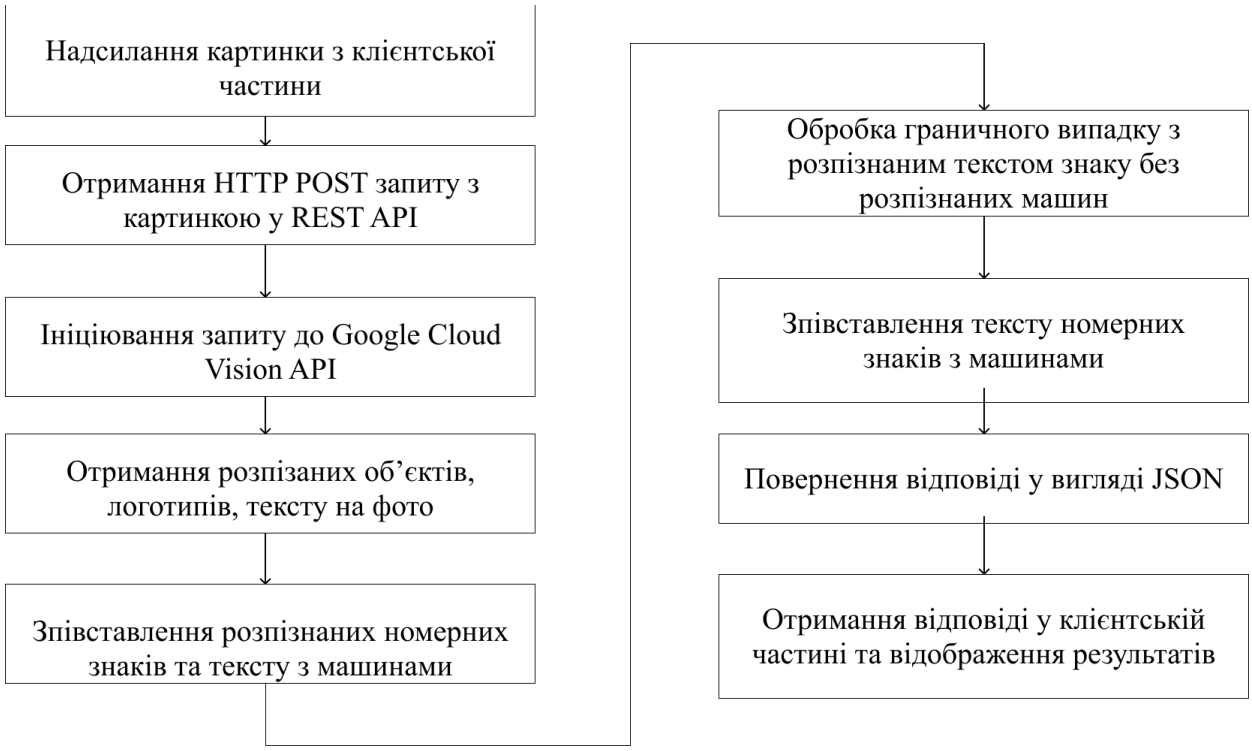


Рисунок 3.5 – Алгоритм роботи програмного забезпечення кіберфізичної системи «Розумна парковка з розпізнаванням номеру»

Після вибору фото з галереї або камери мобільного пристрою, застосунок надсилає HTTP POST запит до сервера. Цей запит потрапляє до Laravel REST API, з якого отримуються параметри, в яких знаходиться фото, закодоване в форматі Base64 для передачі по мережі.

Після отримання фото в текстовому вигляді ініціюється запит до Google Cloud Vision API, обираються необхідні можливості розпізнавання та передається контент фото до запиту. Коли повертається відповідь, ми отримуємо об’єкт відповіді з безліччю параметрів, таких як: розпізнаний текст, об’єкти, логотипи. Відбувається аналіз отриманої інформації, а саме:

– фільтрація об’єктів машин та їх обмежуючих чотирикутників серед усіх об’єктів;

– фільтрація номерних знаків та зіставлення їх з машинами;

– зіставлення окремо розпізнаних «слів» з машинами і номерними знаками на них;

– створення фіктивних об’єктів машин за необхідності (граничний випадок);

– перевірка розпізнаних номерів машин за допомогою регулярного виразу;

– форматування відповіді у форматі JSON та відправка.

Далі відповідь отримується у застосунку клієнта та виводиться у зрозумілому вигляді на екрані, наприклад у вигляді карток.

На рисунку 3.6 зображено схему взаємодії компонентів кіберфізичної системи (додаток Б).

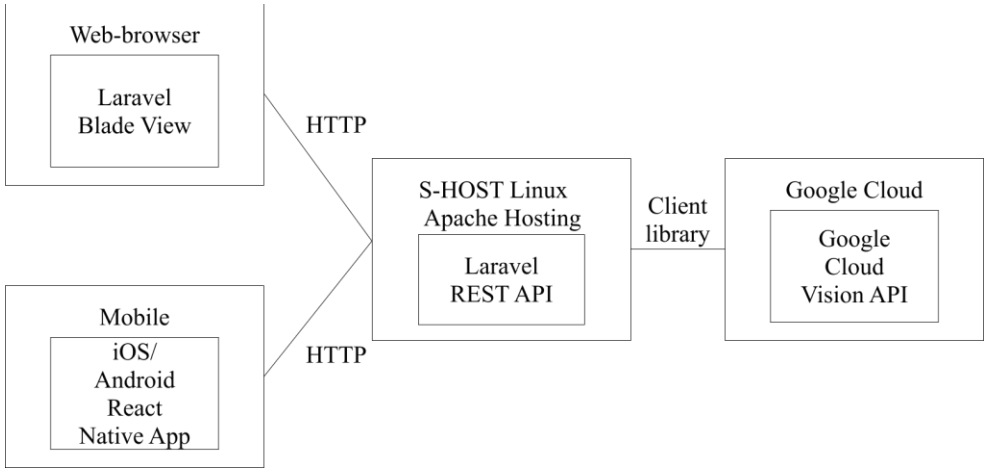


Рисунок 3.6 – Схема взаємодії компонентів кіберфізичної системи «Розумна парковка з розпізнаванням номерного знаку»

В лівій стороні розташовані 2 типи клієнтів: користувачі сайту та мобільні користувачі. Клієнти надсилають запити до Laravel REST API, яке розміщено на хостингу S-HOST з використанням веб-серверу Apache. REST API надсилає запити за допомогою бібліотеки до одного з сервісів Google Cloud, а саме Google Cloud Vision.

На момент написання роботи, не видавалось можливим використати встановлені камери на корпусах, адже вони націлені на зйомку цілої парковки, а не тільки в'їзду. Через це тестування проводилось на фотографіях, які були приблизними варіантами тих, які були б при встановленій камері на в'їзді. Система показала чудовий результат, тому можливе її використання з реальною парковкою на території університету. Представлена реалізація алгоритму на Python для отримання фото з камери, використовуючи RTSP. Блок-схема алгоритму наведена на рисунку 3.7 (додаток Б).

Перевірка кадру на коректність необхідна через те, що на деяких камерах перші кадри, отримані при відкритті потоку є некоректними, майже повністю сірими. Якщо середнє значення кольорів усіх пікселів з деякою похибкою дорівнює 126, тоді кадр некоректний. Також можна визначити коректний кадр за допомогою порівняння його з попереднім. Якщо середнє значення дуже сильно змінилось, це свідчить про коректність останнього кадру.

Рисунок 3.7 – Алгоритм отримання фото з камери через RTSP

3.3 Реалізація підсистеми розпізнавання номерних знаків та моделей автомобілів у кіберфізичній системі “Розумна парковка”

Після визначення принципу роботи та структури системи, було вирішено почати процес реалізації. Для цього були використані такі технології:

– PHP та фреймворк Laravel, Blade та бібліотеку jQuery для реалізації PHP ANPR REST API та веб-сторінки клієнтської частини;

– Python та фреймворк Flask, бібліотеку OpenCV для реалізації Python OpenCV Image Getting API;

– google/cloud-vision пакет в менеджері Composer для доступу до Google Cloud Vision API;

– React Native для реалізації кросплатформного мобільного застосунку.

Для використання Laravel необхідно встановити менеджер пакетів Composer. Він є кросплатформним та використовується для мови програмування PHP. Створення нового Laravel-проєкту використовується команда composer create-project laravel/laravel:^10.0 example-app, в якій вказується дія, бажана версія Laravel та назва. Також можливий варіант встановлення використовуючи веб-застосунок Softaculous (рисунок 3.8), який зазвичай є встановленним на хостингу.

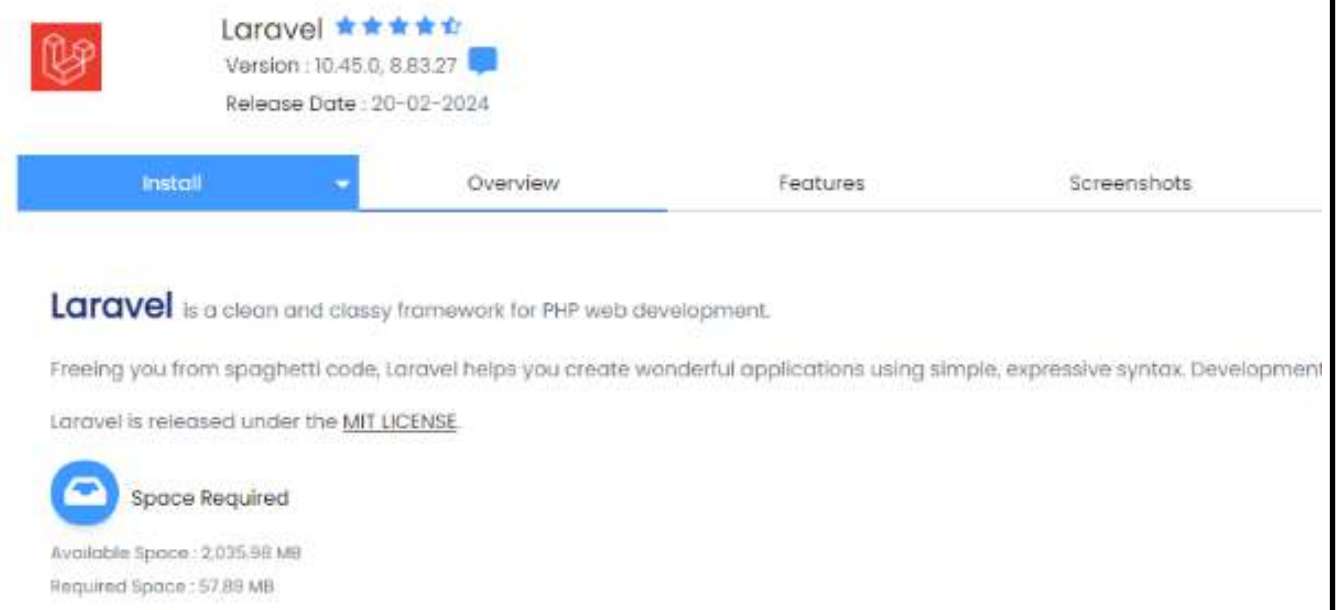


Рисунок 3.8 – Встановлення Laravel

Також встановимо бібліотеку для роботи з Google Cloud Vision API, використовуючи команду composer require google/cloud-vision.

Laravel пропонує певну структуру директорій (рисунок 3.9), яка підходить як для малих, так і для великих проєктів.

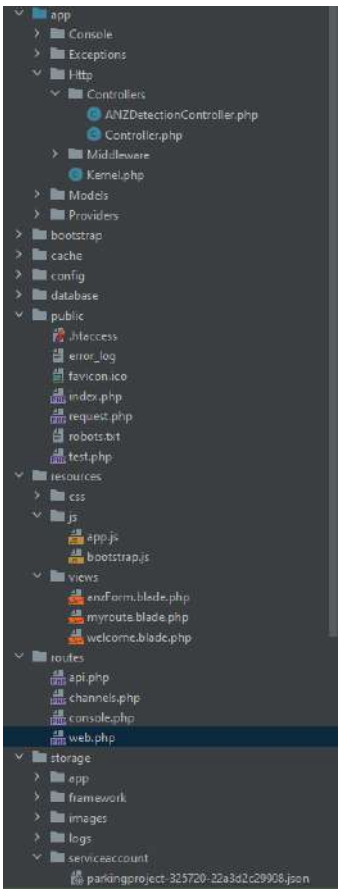


Рисунок 3.9 – Структура директорій PHP ANPR REST API

Обробка запитів відбувається у файлі public/index.php, на який перенаправляються усі запити, що надходять від користувачів. Це правило налаштовується автоматично у файлі public/.htaccess, проте також необхідно створити перенаправлення усіх запитів у директорію public в файлі .htaccess вищого рівня. У файлі index.php створюється екземпляр застосунку, який створює клас ядра для обробки HTTP запиту. Це ядро отримує запит та повертає відповідь користувачу. URL зіставляється з існуючими шляхами, якщо ж такого шляху не існує, повертається помилка 404. Інакше, викликається функція замикання, що пов’язана з шляхом та оброблює запит. Зазвичай, оброблення передається відповідному методу контролера або одразу повертається представлення, якщо цей запит є простим. Контролери існують для організації бізнес-логіки застосунку та оброблення запитів однієї групи. Також запити

можуть проходити через додаткові елементи проміжного програмного забезпечення (Middleware), які виконують функції авторизації, захисту, т.д.

При розробці PHP ANPR REST API було додано декілька сутностей:

– контролер ANZDetectionController (розпізнавання номерних знаків);

– представлення anzForm.blade.php (клієнтська форма з вибором файлу для аналізу);

– шляхи у api.php та web.php (шляхи у api.php мають префікс API, а у web.php подаються у такому ж вигляді);

– Serviceaccount/parkingproject….json (дані акаунту, отримані з Google Cloud Console).

Почнемо з контролеру ANZDetectionController. Опис його функцій подано у таблиці 3.1.

Таблиця 3.1 – Опис функцій контролеру ANZDetectionController.

№ п/п	Назва функції	Опис функції
1	2	3
1	normalizedVerticesToArray	Перетворення нормалізованих координат на звичайні та створення array з RepeatedField
2	verticesToArray	Перетворенння RepeatedField в array та відображення координати у по вертикальній вісі
3	isInside	Перевірка на належність одного чотирикутника іншому
4	isInsidePolygon	Перевірка чи точка належить чотирикутнику
5	rayIntersectsSegment	Перевірка орієнтації точки відносно ребра

Продовження таблиці 3.1

1	2	3
6	extendBounding	Розширення чотирикутника на певну величину
7	checkANZ	Перевірка чи стрічка містить номерний знак, використовуючи регулярний вираз
8	prepareANZ	Відокремлення номерного знаку, перетворення схожих на цифри букв, англійські букви в українські
9	getImageDimensions	Отримання розмірів зображення з файлу або base64 стрічки
10	getCars	Отримання масиву машин з розпізнаних об’єктів
11	getLicensePlates	Визначення номерних знаків серед розпізнаних об’єктів, співставлення їх з розпізнаними машинами
12	setTextToCars	Отримання розпізнаного тексту та співставлення його з машинами та номерними знаками(якщо є)
13	getLogosOnCar	Отримання логотипів та співставлення з розпізнаними машинами

Кінець таблиці 3.1

1	2	3
14	getANZFromImage	Основна функція, що викликає функції описані вище та повертає масив машин з розпізнаними елементами
15	getANZ	Обробка вхідного запиту та виклик функції getANZFromImage

Розглянемо деякі з функцій детальніше. При потраплянні запиту до контролеру викликається функція getANZ, яка приймає екземпляр запиту Request $request та отримує з нього поля content і isFile. Дані передаються в вигляді JSON payload POST запитом. Далі викликається функція getANZFromImage з цими параметрами.

Найпершою дією є створення клієнту ImageAnnotatorClient для можливості ініціювання запитів розпізнавання. Створюємо запит AnnotateImageRequest, отримуємо список через getFeatures, встановлюємо підказку мови розпізнавання на українську. Створюємо зображення Image, отримуємо його розміри через getImageDimensions та встановлюємо зміст з getImageContent(). Ініціюємо запит, використовуючи функцію batchAnnotateImages у ImageAnnotatorClient(). Отримавши результат, розпочинається його аналіз. Фрагмент коду наведено нижче:

```
$annotateImageRequest = new AnnotateImageRequest();
$image = new Image();
list($width, $height) = $this->getImageDimensions($content, $isFile);
$image->setContent($this->getImageContent($content, $isFile));
$annotateImageRequest->setImage($image);
$annotateImageRequest->setFeatures($features);
$annotateImageRequest->setImageContext($imageContext);
$requests[] = $annotateImageRequest;
$batchAnnotateImageResponse = $client->batchAnnotateImages($requests);
$annotateImageResponses = $batchAnnotateImageResponse->getResponses();
```

Викликаючи функцію getCars та передавши у неї список анотацій об’єктів, отримуємо список машин, які комп’ютерний зір розпізнав на фото. Отримуємо

номерні знаки з функції getLicensePlates та зіставляємо їх з машинами. Зіставлення відбувається при умові належності чотирикутника розпізнаного номерного знаку чотирикутнику розпізнаної машини, що перевіряється функціями isInside, isInsidePolygon, rayIntersectsSegment. Також розширяємо межі чотирикутника номерного знаку функцією extendBounding для пом'якшення похибок при роботі з дробовими числами. Аналогічним чином визначаємо текст та логотипи на машинах функціями setTextToCars, getLogosOnCar. Це відбувається за допомогою такого програмного коду:

```
$cars = $this->getCars($localizedObjectAnnotations, $width, $height,
$thresholdX, $thresholdY);
$this->getLicensePlates($localizedObjectAnnotations, $width, $height,
$thresholdX, $thresholdY, $cars);
$textAnnotations = $annotateImageResponses[0]->getTextAnnotations();
$detectedText = $textAnnotations[0]->getDescription();
$sentences = explode("\n", $detectedText);
foreach ($sentences as &$sentence) {
    $sentence = ["text" => $sentence, "used" => false];
}
unset($sentence);
$this->setTextToCars($textAnnotations, $height, $cars);
$logos = $annotateImageResponses[0]->getLogoAnnotations();
$this->getLogosOnCar($logos, $height, $cars);
```

Останнім етапом є формування відповіді. Провівши багато тестувань на різних прикладах фото машин, було визначено, що трапляються випадки, коли є розпізнаний текст, в якому містяться номерні знаки, проте немає розпізнаних машин. Такий випадок перевіряється і для кожного розпізнаного номеру створюється машина, обмежуючий чотирикутник якої є усе фото, це дозволяє програмі коректно видати результат.

В результаті отримуємо список машин з сутностями на них, такими як: номерний знак, текст, текст на номерному знаку. Перевіряємо текст на номерному знаку функцією prepareANZ або текст в цілому на те, чи він містить коректний номер машини і в разі успіху записуємо його у поля машини. Якщо номер розпізнано, то машина є одним з елементів масиву відповіді.

Фрагмент реалізації наведено нижче:

```
foreach ($cars as &$car) {
    if (isset($car["entities"])) {
        if (!(isset($car["anz"]) && $car["anz"] !== "Unrecognized")) {
            $anz = "";
            if (isset($car["entities"]["licensePlate"])) {
                if (isset($car["entities"]["textInLicensePlate"]))
                    $anz = $this->getSentenceBySequenceOfWords($sentences,
$this->getWordsFromCar($car["entities"]["textInLicensePlate"]));
            } else {
                if (isset($car["entities"]["text"]))
                    $anz = $this->getSentenceBySequenceOfWords($sentences,
$this->getWordsFromCar($car["entities"]["text"]));
            }
            $preparedANZ = $this->prepareANZ($anz);
            $preparedANZ = empty($preparedANZ) ? "Unrecognized" : $preparedANZ;
            $car["anz"] = $preparedANZ;
        }
        if (isset($car["anz"]) && $car["anz"] !== "Unrecognized") {
           $carsResponse[] = $car;
        }}}
unset($car);
return $carsResponse;
```

Функції getSentenseBySequenceOfWords та getWordsFromCar використовується для співставлення розпізанних «речень» комп'ютерним зором та окремих розпізаних «слів».

Нарешті, функція getANZ повертає результат у форматі JSON (рисунок 3.10) з кодом 200, що означає успіх.

Рисунок 3.10 – Приклад відповіді у JSON форматі

В складі проєкту Laravel також реалізовано клієнтську частину – веб сторінку, що дозволяє надіслати фото на обробку системою та отримати результат розпізнавання. Приклад взаємодії зображено на рисунку 3.11.

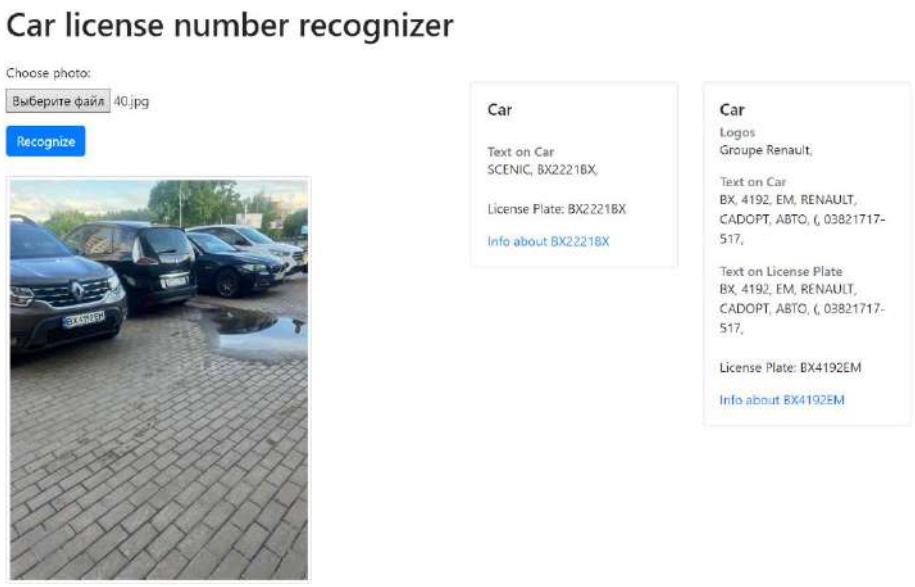


Рисунок 3.11 – Приклад використання веб-сторінки для розпізнавання

Для отримання результату з нашого PHP ANPR REST API використовується бібліотека jQuery, а саме метод $.ajax. Отримавши результат з серверу, відбувається його форматування у зрозумілому людині форматі. Фрагмент реалізації наведено нижче:

```
$.ajax({
    url: 'https://parkapi.turistmapa.com.ua/api/getANZ',
    type: 'POST',
    data: formData,
    processData: false,
    contentType: false,
    success: function (response) {
        $('#loadingSpinner').hide();
        console.log(response);
        displayCars(response);
    },
    error: function (xhr, status, error) {
        $('#loadingSpinner').hide();
        console.error(xhr.responseText);
    }
});
```

Для забезпечення мінімально допустимого зовнішнього вигляду [51] використовується CSS бібліотека Bootstrap.

Звернемо увагу на параметр url у формуванні запиту. Цим url є кінцева точка (endpoint) нашого API. Такі endpoint оголошуються у файлах routes/api.php та routes/web.php. Код для оголошення шляхів наведено нижче:

```
routes/api.php
    Route::post('/getANZ',[ANZDetectionController::class,'getANZ']);
routes/web.php
    Route::view('/anzForm','anzForm');
```

Запис Route::post означає, що методом цього запиту є POST, а Route::view – те, що ми одразу повертаємо представлення. Route::view дозволяє звертатись тільки методом GET.

Окремо також було розроблено мобільну частину проєкту у вигляді застосунку (рисунок 3.12) з використанням React Native.

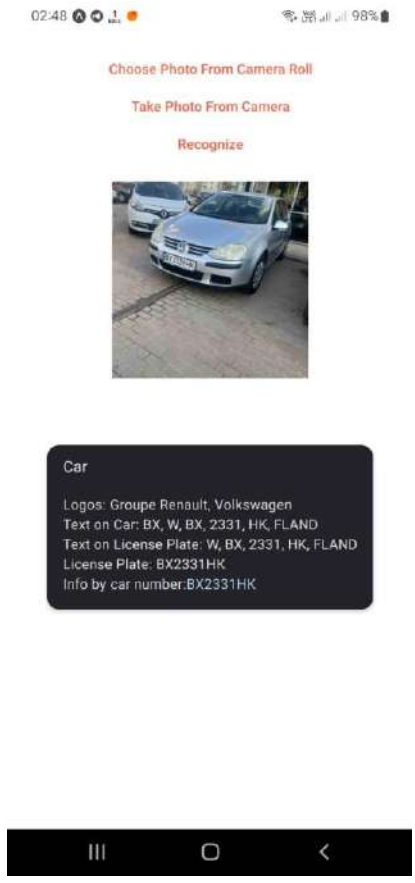


Рисунок 3.12 – Інтерфейс мобільного застосунку

Мобільний застосунок має можливість виконувати аналогічні дії з надсилання фото та отримання результату розпізнавання. Він має майже однаковий інтерфейс на Android та iOS, розбіжності є тільки у відображенні карточок з результатами. Нижче наведено фрагмент коду, що відповідає за відображення результату користувачу:

```
<View style={styles.column}>
    <ActivityIndicator animating={loading} size="large" color="#0000ff"/>
        {result && result.map((car, index) => {
            return (
            <Card key={index}>A
                <Card.Title title={car.name}></Card.Title>
                <Card.Content>
                    <Text variant={"bodyMedium"}>Logos:
{car.entities?.logo?.map((logo) => logo.description).join(', ')}</Text>
                    <Text variant={"bodyMedium"}>Text on Car:
{car.entities?.text?.map((text) => text.description).join(', ')}</Text>
                    <Text variant={"bodyMedium"}>Text on License Plate:
{car.entities?.textInLicensePlate?.map((text) => text.description).join(',
')}</Text>
                    <Text variant={"bodyMedium"}>License Plate: {car.anz}</Text>
                    <Text variant={"bodyMedium"}>Info by car number:
                        <Text style={{color:"lightblue"}}
variant={"bodyMedium"}
onPress={()=>Linking.openURL(`https://ua.carplates.app/en/number/${car.anz}`)}>{
car.anz}</Text>
     </Text>
                </Card.Content>
            </Card>);
         }
     )}
</View>
```

Змінна result є зміною стану, при зміні якої перемальовюється компонент та з’являються карточки з результатами. Для надсилання запиту на наш API використовується Fetch API з методом then у Promise для отримання результату та метод json для отримання об’єкту відповіді.

3.4 Висновки

Отже, було визначено принцип роботи та структурну схему підсистеми розпізнавання номерних знаків та моделей автомобілів кіберфізичної системи. Запропоновано архітектуру з використанням REST API серверної частини,

клієнтського застосунку, розробленого за допомогою React Native та веб-сторінки на Laravel Blade.

Розроблена система пропонує користувачу отримати розпізнані номерні знаки з машин на фото, дізнатись детальну інформацію про авто за його номером з ресурсу CarPlates. Тестування було проведено на власному датасеті [52] фотографій машин.

Було описано основні алгоритми роботи серверної частини програмного забезпечення, а саме алгоритм обміну даних між клієнтською частиною та серверною, аналіз розпізнаних об’єктів засобами Google Cloud Vision, алгоритм отримання фотографії з камери за допомогою RTSP [53].

Система реалізована вище переліченими засобами, а саме:

– PHP фреймворк Laravel;
– Python фреймворк Flask;
– клієнтська бібліотека Google Cloud Vision;
– JavaScript фреймворк React Native.

ВИСНОВКИ

У кваліфікаційній роботі було розроблено підсистему розпізнавання моделі та номерного знаку автомобіля кіберфізичної системи «Розумна парковка» з використанням комп'ютерного зору та REST API архітектури.

У першому розділі було досліджено існуючі рішення для автоматичного розпізнавання номерних знаків автомобілів на парковках. Проаналізовано та визначено переваги та недоліки кожної системи. Визначено основні складові кіберфізичної системи «Розумна парковка з розпізнаванням номерних знаків». Було проведено огляд апаратних засобів системи. Описано модульну структуру програмної частини. Запропоновано свою концепцію архітектури кіберфізичної системи.

У другому розділі обґрунтовано вибір компонентів та середовища реалізації кіберфізичної системи. Виконано опис використаного серверу для реалізації, пояснено використання віртуального хостингу, описано його характеристики. Сформовано вимоги до користувачів системи, а саме апаратні вимоги до персонального комп'ютера та мобільний пристроїв з операційною системою Android або iOS. Складено схему запиту у системі, розглянуто функційні та нефункційні вимоги до системи. Визначено архітектуру системи з використанням принципу REST API. Обрано та аргументовано вибір методів та середовища для реалізації програмного забезпечення проєкту, а саме:

- Python фреймворк Flask;
- Python реалізація бібліотеки OpenCV;
- PHP фреймворк Laravel;
- клієнтська бібліотека Google Cloud Vision;
- JavaScript фреймворк React Native;
- середовище розробки JetBrains PhpStorm.

У третьому розділі було описано програмну реалізацію підсистеми автоматичного розпізнавання номерних знаків. Побудовано схему архітектури

програмного забезпечення кіберфізичної системи, яка складається з 4 складових частин:

– клієнтська частина – веб-сторінка або мобільний застосунок;

– PHP ANPR REST API – прикладний програмний інтерфейс для розпізнавання номерних знаків;

– Python OpenCV Image Getting API – прикладний програмний інтерфейс для отримання фото з камери або відео;

– Google Cloud Vision – прикладний програмний інтерфейс для взаємодії з комп'ютерним зором Google Cloud Vision.

Проаналізовано результати, отриманні під час розпізнавання номерних знаків, використовуючи Google Cloud Vision. Визначено проблеми, що виникають при цьому, запропоновано способи їх рішення. Спроєктовано структурну схему кіберфізичної системи, наведено алгоритми роботи прикладного інтерфейсу розпізнавання номерних знаків та отримання фото з камери. Представлено актуальну схему компонентів системи. Описано процес реалізації програмної частини. Подано таблицю функцій, що використовуються при процесі розпізнавання номерних знаків. Наведено фрагменти програмного коду, що реалізовує алгоритм розпізнавання, а також фрагменти клієнтської програмної частини – React Native застосунок та Laravel Blade сторінку.

Розроблена кіберфізична система допоможе спростити роботу працівникам контрольно-пропускних пунктів на парковках, а саме дозволить швидко авторизувати транспорт.

Подальше дослідження можна присвятити покращенню системи, наприклад, створенню повноцінної системи автоматичної розумної парковки з білим списком транспортних засобів, авторизацією працівників. Також варто зазначити, що на даний момент система сприймає лише українські номери конкретного зразку, тому цей напрямок досліджень є дуже важливим.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Detectron2 – a platform for object detection, segmentation and other visual recognition tasks. URL: https://github.com/facebookresearch/detectron2 (дата звернення: 07.12.2023).

2. IP-відеокамера та комплектуючі. URL: https://rozetka.com.ua/hikvision_ds_2cd2121g0_is_c_2_8_mm/p322796167/characteristics/ (дата звернення: 07.12.2023).

3. Radiuk, P., Pavlova, O., El Bouhissi, H., Avsiyevych, V., Kovalenko, V. Convolutional Neural Network for Parking Slots Detection. *CEUR Workshop Proceedings*. 2022, Vol. 3156. P. 284–293.

4. Automatic Number Plate Recognition (ANPR) – 2024 Guide. URL: https://viso.ai/computer-vision/automatic-number-plate-recognition-anpr/ (дата звернення: 29.03.2024).

5. Avsiyevych V., Kovalenko V. Cyber-Physical System For Smart Parking Based On Computer Vision Technology. *Black Sea Science 2022: Proceedings of the International Competition of Student Scientific Works* / Odesa National University of Technology; B. Iegorov, M. Mardar (editors-in-chief) [et al.]. Odesa: ONUT, 2022. – P. 335-346.

6. Львівська компанія SoftServe почала тестування паркінг-системи на базі машинного навчання. URL: https://zaxid.net/lvivska_kompaniya_softserve_pochala_testuvannya_parking_sistemi_na_bazi_mashinnogo_navchannya_n1471000 (дата звернення: 30.11.2023).

7. Паркінг в Буковелі. Трансфер до Карпат. URL: https://transferdokarpat.com.ua/articles/bukovel-vartist-poslugi-parkingiv (дата звернення: 30.11.2023).

8. У Києві стартував пілот із впровадження «розумної» системи паркування. Офіційний портал Києва. URL: https://kyivcity.gov.ua/news/u_kiyevi_startuvav_pilot_iz_vprovadzhennya_rozumno_sistemi_parkuvannya/ (дата звернення: 30.11.2023).

9. Інтелектуальна система парковки Acer. URL: https://www.acer.com/ac/ru/RU/content/acerdesign-smart-parking (дата звернення: 30.11.2023).

10. Innovative Parking Transformation by Smart Parking Ltd. Smart Parking. URL: https://www.smartparking.com/nz (дата звернення: 29.03.2024).

11. Керування парковкою – Офіційний сайт ZKTeco. URL: https://zkteco.technology/solution/parking-management/ (дата звернення: 20.04.2024).

12. OpenALPR – Automatic License Plate Recognition. URL: https://www.openalpr.com/ (дата звернення: 20.04.2024).

13. OpenALPR Library URL: https://github.com/openalpr/openalpr (дата звернення: 20.04.2024).

14. Under the Magnifier. ANPR Systems' Components. Adaptive Recognition. URL: https://adaptiverecognition.com/blog/traffic-transportation/under-the-magnifier-components-of-anpr-systems/ (дата звернення: 20.04.2024).

15. Камери відеоспостереження з ANPR (визначення номерних знаків). URL: https://hikvision.co.ua/ua/kamery-videonablyudeniya/anpr/ (дата звернення: 20.04.2024).

16. IP-відеокамера Dahua DH IPC HDW2431. URL: https://rozetka.com.ua/dahua_dh_ipc_hdw2431tp_as_s2_3_6_mm/p175174490/characteristics/ (дата звернення: 07.12.2023).

17. Коваленко В. В. Кіберфізична система розумної парковки на основі технології комп'ютерного зору: кваліфікаційна робота магістра: 123 Комп'ютерна інженерія. ХНУ. Хмельницький, 2022. 113 с.

18. Learn OpenCVC++ in 4 hours. URL: https://www.youtube.com/watch?v=2FYm3GOonhk (дата звернення: 02.12.2023).

19. Introduction to OpenCV-Python Tutorials. URL: https://docs.opencv.org/3.4/d0/de3/tutorial_py_intro.html (дата звернення: 19.02.2024).

20. Vision AI. Google Cloud. URL: https://cloud.google.com/vision (дата звернення: 02.12.2023).

21. Top 10 OpenCV Alternatives & Competitors. URL: https://www.g2.com/products/opencv/competitors/alternatives (дата звернення: 20.04.2024).

22. Parking Startups Are Cashing In on America's Traffic Surge. URL: https://www.bloomberg.com/news/articles/2021-07-22/parking-startups-cash-in-on-america-s-post-pandemic-traffic-surge-with-apps (дата звернення: 07.12.2023).

23. Авсієвич В., Кузьмін А. Дослідження вразливостей системи розумної парковки та способи їх усунення. *Актуальні проблеми комп'ютерних наук (АПКН-2022).* Хмельницький, Україна, 18-19 листопада 2022. Хмельницький: ХНУ, 2022. С. 11-14.

24. Hovorushchenko, T., Pavlova, O., Avsiyevych, V. Method of Assessing the Impact of External Factors on Geopositioning System Operation Using Android GPS API. *International Scientific and Technical Conference on Computer Sciences and Information Technologies*, 2021. No 1. P. 295–298.

25. Gollapudi S. Learn computer vision using opencv: with deep learning CNNs and RNNs : eBook. Berkeley, CA : Apress, 2019. 171 p.

26. Avsiyevych V., Kawonga R. Security of smart parking cybersphysical system. *Information Technology & Engineering – 2023*. February 7-10, 2023, Mykolayiv, Ukraine. P. 59-61.

27. Павлова О.О., Авсієвич В.Р., Кузьмін А.А. Дослідження факторів впливу на безпеку мобільних застосунків на прикладі клієнтської частини кіберфізичної системи розумної парковки. *Стан, досягнення та перспективи інформаційних систем і технологій*: матеріали XXIII Всеукраїнської науково-технічної конференції молодих вчених, аспірантів та студентів. Одеса, 20-21 квітня 2023 р. Одеса, Видавництво ОНТУ, 2023. С. 98-99.

28. Parking Lot Database. URL: https://web.inf.ufpr.br/vri/databases/parking-lot-database/ (дата звернення: 07.12.2023).

29. СХОСТ Український хостинг сайтів. URL: https://s-host.com.ua/ (дата звернення: 20.04.2024).

30. Камери Hikvision з підтримкою PoE та вуличним встановленням. URL:https://hikvision.co.ua/ua/kamery-videonablyudeniya/ip-kamery/?ocf=F76S3V430F60S3V373 (дата звернення: 20.04.2024).

31. Google Play Market. URL: https://play.google.com/store (дата звернення: 30.11.2023).

32. Apple App Store. URL: https://www.apple.com/ua/app-store/ (дата звернення: 30.11.2023).

33. Balaji G.V., Bharath K., Nithin S., Pranesh D.M. Shilpa S.B. Object detection using OpenCV and deep learning. *International Journal for Research in Applied Science and Engineering Technology*. 2021. Vol. 9. No. 1. P. 3920-3923.

34. About OpenCV. URL: https://opencv.org/about/#:~:text=OpenCV%20(Open%20Source%20Computer%20Vision,perception%20in%20the%20commercial%20products (дата звернення: 20.04.2024).

35. What is Flask Python. URL: https://pythonbasics.org/what-is-flask-python/ (дата звернення: 20.04.2024).

36. The PHP Framework for Web Artisans. URL: https://laravel.com/ (дата звернення: 19.02.2024).

37. Laravel Framework GitHub. URL: https://github.com/laravel/framework (дата звернення: 19.02.2024).

38. Laravel – Overview. URL: https://www.tutorialspoint.com/laravel/laravel_overview.htm (дата звернення: 19.02.2024).

39. What is API. URL: https://aws.amazon.com/what-is/api/ (дата звернення: 20.04.2024).

40. How to Build an Effective API Security Strategy. URL: https://www.gartner.com/en/documents/3834704 (дата звернення: 19.02.2024).

41. Google Cloud Vision API. URL: https://cloud.google.com/vision?demo (дата звернення: 20.04.2024).

42. What is React Native. URL: https://www.netguru.com/glossary/react-native (дата звернення: 20.04.2024).

43. Facebook React Native Github. URL: https://github.com/facebook/react-native/blob/main/packages/react-native/scripts/cocoapods/helpers.rb (дата звернення: 20.04.2024).

44. Кузьмін А. А. Серверна підсистема кіберфізичної системи "Розумна парковка" : кваліфікаційна робота бакалавра : 123 Комп'ютерна інженерія А. А. Кузьмін ; ХНУ. Хмельницький, 2023. 78 с.

45. Pavlova O., Kovalenko V., Hovorushchenko T., Avsiyevych V. Neural network based image recognition method for smart parking. *Comput. Syst. Inf. Technol.* 2021. No 3. P. 49–55.

46. Авсієвич В., Коваленко В. Аналіз інформаційних технологій для розумної парковки на основі штучних нейронних мереж. *Актуальні проблеми комп'ютерних наук (АПКН-2021)*, Хмельницький, Україна, 15-16 жовтня 2021. Хмельницький: ХНУ, 2021. С. 12-14.

47. Almeida P., Oliveira L. S., Silva E., Britto A., Koerich A. PKLot – A robust dataset for parking lot classification. *Expert Systems with Applications*. 2015. Vol. 42(11). P. 4937-4949.

48. Melbourne University Dataset. URL: https://melbourne.figshare.com/articles/dataset/MATLABCodeCNNSVM_zip/12978932/1?file=24726374 (дата звернення: 07.12.2023).

49. Marek M. Official repository for the «Image-Based Parking Space Occupancy Classification: Dataset and Baseline» paper. URL: https://github.com/martin-marek/parking-space-occupancy (дата звернення: 07.12.2023).

50. Репозитарій із датасетом зображень, зібраних з камер зовнішнього спостереження ХНУ. URL: https://github.com/soolstafir/khnu-parking-lot (дата звернення: 02.12.2023).

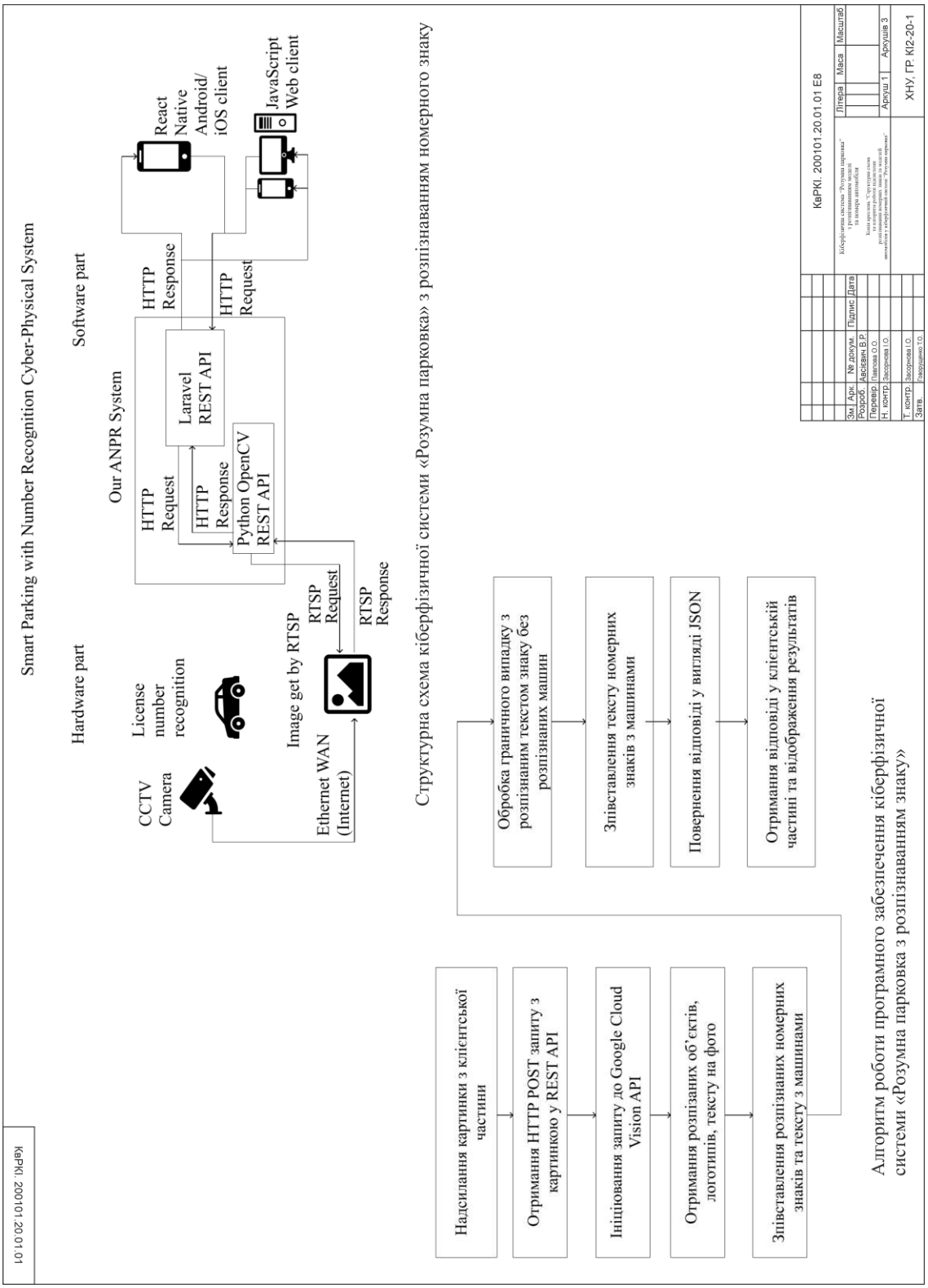
51. Швайко В. К., Авсієвич В. Р. Інформаційна система візуалізації пунктів переробки вторинної сировини для забезпечення концепції сталого розвитку. *Актуальні проблеми комп'ютерних наук (АПКН-2021)*, Хмельницький, Україна, 15-16 жовтня 2021. Хмельницький: ХНУ, 2021. С. 268-271.

52. Amato G., Carrara F., Falchi F., Gennaro C., Vairo C. CNRPark+EXT. A Dataset for Visual Occupancy Detection of Parking Lots. URL: http://cnrpark.it/ (дата звернення: 07.12.2023).

53. Kaehler A., Bradski G. Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library 1st Edition. O'Reilly Media. 2017. 1024 p.

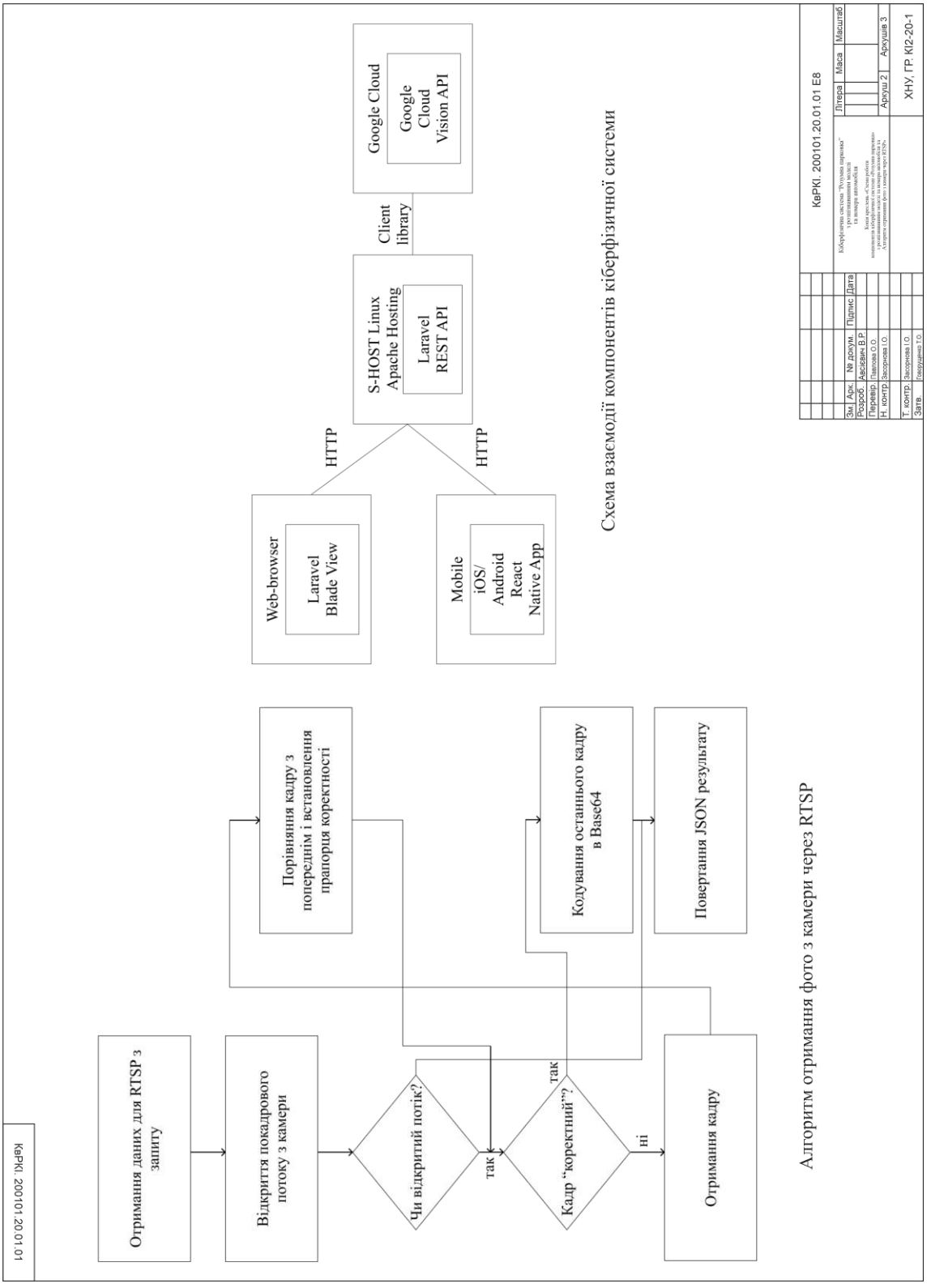
Додаток А

Копія креслень «Структурна схема та алгоритм роботи підсистеми розпізнавання номерних знаків та моделей автомобілів у кіберфізичній системі «Розумна парковка»



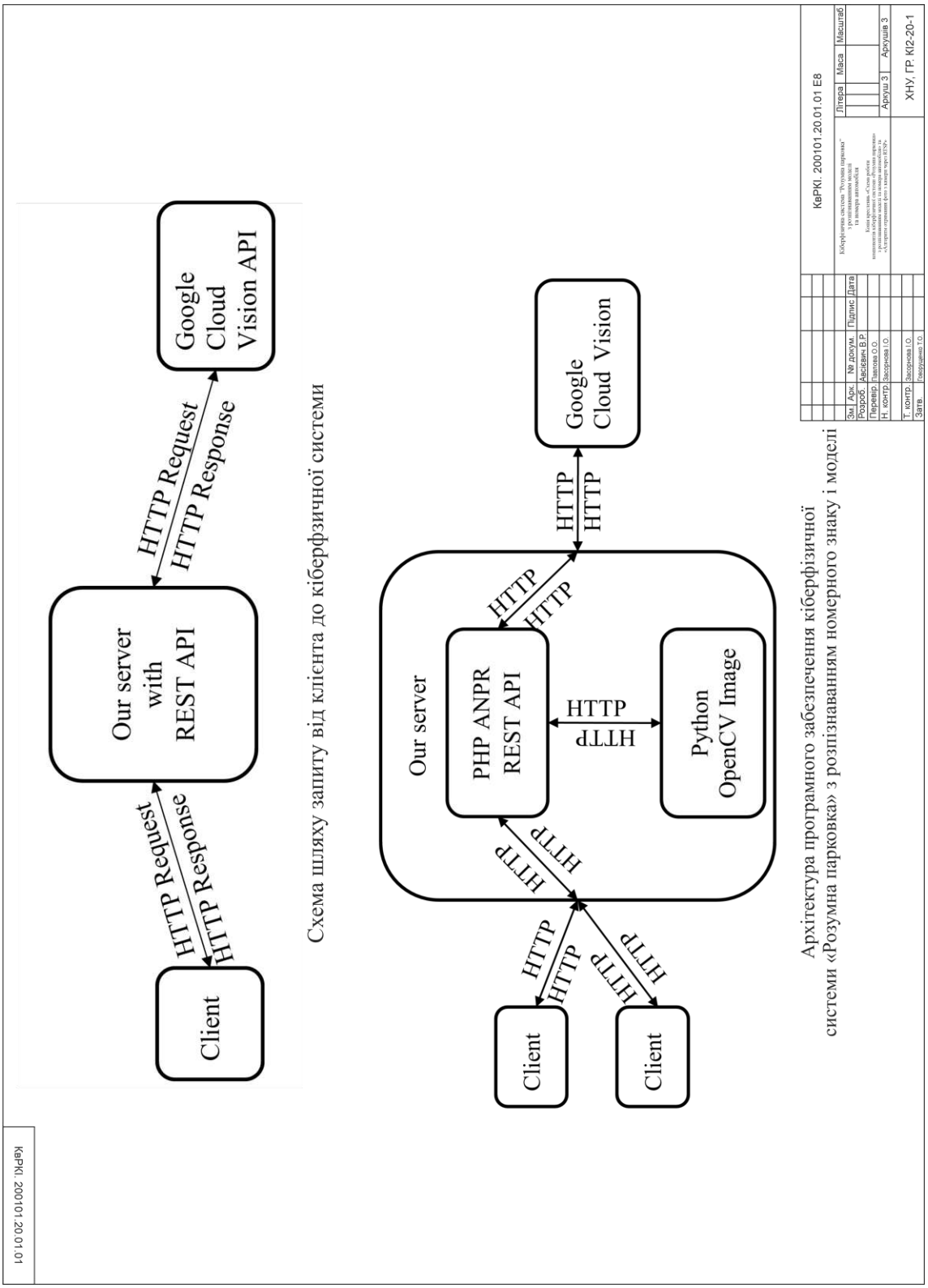
Додаток Б

Копія креслень «Схема роботи компонентів кіберфізичної системи «Розумна парковка» з розпізнаванням моделі та номера автомобіля та Алгоритм отримання фото з камери через RTSP»



Додаток В

Копія креслень «Архітектура підсистеми розпізнавання номерних знаків та моделей автомобілів у кіберфізичній системі «Розумна парковка»



Додаток Г

Лістинг коду програмного забезпечення підсистеми розпізнавання номерних знаків та моделей автомобілів у кіберфізичній системі «Розумна парковка»

Модуль отримання зображення з камери, використовуючи RTSP(Python Flask і бібліотека OpenCV)

```
from flask import Flask, abort, jsonify, request
import cv2
import time
import datetime
import base64
import numpy
import sys

app = Flask(__name__)

def get_rtsp_url(data):
    # with open(config_file, "r") as f:
    # data = f.read()
    login = data['login']
    password = data['password']
    address = data['address']
    port = data['port']
    channel = data['channel']
    #login, password, address, port, channel = data.split(":")
    url = "rtsp://{}:{}@{}:{}/Streaming/Channels/{}".format(login, password, address, port, channel)
    return url

@app.route("/get_cam_photo", methods=['POST'])
def imageroute():
    data = request.get_json()
    rtsp_url = get_rtsp_url(data['params'])
    #return jsonify({'data': rtsp_url}), 200
    cap = cv2.VideoCapture(rtsp_url)
    #return jsonify({'data': rtsp_url}), 200
    fileLog = open('mylog.txt', 'w')
    means = []
    alpha = 0.15
    lastiter = 0
    if cap.isOpened():
        ret = False
        frame = None
        imgenconde = None
        good = False
        badval = 126
        for iter in range(0, 10):
            lastiter = iter
            ret, frame = cap.read()
            means.append(frame.mean())
            threshold = -1
            diff = -1
            if iter > 0:
                diff = abs(means[iter] - means[iter - 1])
                threshold = alpha * means[iter - 1]
                if diff > threshold:
                    good = True
```

```
print(f"frame{iter}value={means[iter]}diff={diff}threshold={threshold}", file=fileLog)

        name = "frame{}.png".format(iter)
        cv2.imwrite(name, frame)
        if good:
           break

    if good or (not good and means[lastiter] < badval):
        encode_param = [int(cv2.IMWRITE_PNG_COMPRESSION), 0]
        result, imgencode = cv2.imencode('.png', frame, encode_param)
        # data = numpy.array(imgencode)
        # stringData = data.tobytes()
        cap.release()
        image_bytes = imgencode.tobytes()
        # Encode the image bytes using Base64
        base64_encoded = base64.b64encode(image_bytes).decode('utf-8')
        return jsonify({'image_data': base64_encoded})

    else:
        cap.release()
        return jsonify({'image_data': 'no data'})

    cap.release()
```

Модуль розпізнавання номерних знаків (PHP Laravel)

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Google\Cloud\Vision\V1\Feature\Type;
use Google\Cloud\Vision\V1\Feature as GFeature;
use Google\Cloud\Vision\V1\ImageAnnotatorClient;
use Google\Cloud\Vision\V1\ImageContext as ImageContext;
use Google\Cloud\Vision\V1\AnnotateImageRequest as AnnotateImageRequest;
use Google\Cloud\Vision\V1\Image as Image;

class ANZDetectionController extends Controller
{

…
private function getANZFromImage($content, $isFile)
  {
    if ($isFile) $content = __DIR__ . "/../../../storage/images/" . $content;
    else $content = base64_decode($content);
    //return ["dir"=>env("GOOGLE_APPLICATION_CREDENTIALS")];
    //return ["dir"=>__DIR__];
    putenv("GOOGLE_APPLICATION_CREDENTIALS=" . __DIR__ . "/../../../" . env("GOOGLE_APPLICATION_CREDENTIALS"));

    $client = new ImageAnnotatorClient();

    $requests = array();

    $features = $this->getFeatures();

    $imageContext = new ImageContext();
    $imageContext->setLanguageHints(["uk"]);

    $annotateImageRequest = new AnnotateImageRequest();

    $image = new Image();

    list($width, $height) = $this->getImageDimensions($content, $isFile);
```

```
        $image->setContent($this->getImageContent($content,
$isFile));

        $annotateImageRequest->setImage($image);
        $annotateImageRequest->setFeatures($features);
        $annotateImageRequest-
>setImageContext($imageContext);

        $requests[] = $annotateImageRequest;

        $batchAnnotateImageResponse = $client-
>batchAnnotateImages($requests);
        $annotateImageResponses =
$batchAnnotateImageResponse->getResponses();



        $localizedObjectAnnotations =
$annotateImageResponses[0]-
>getLocalizedObjectAnnotations();

        $thresholdX = $width * 0.05;
        $thresholdY = $height * 0.05;

        $cars = $this->getCars($localizedObjectAnnotations,
$width, $height, $thresholdX, $thresholdY);

        $this->getLicensePlates($localizedObjectAnnotations,
$width, $height, $thresholdX, $thresholdY, $cars);

        $textAnnotations = $annotateImageResponses[0]-
>getTextAnnotations();
        $detectedText = $textAnnotations[0]->getDescription();

        $sentences = explode("\n", $detectedText);
        foreach ($sentences as &$sentence) {
            $sentence = ["text" => $sentence, "used" => false];
        }
        unset($sentence);

        $this->setTextToCars($textAnnotations, $height, $cars);



        $logos = $annotateImageResponses[0]-
>getLogoAnnotations();

        $this->getLogosOnCar($logos, $height, $cars);

        $carsResponse = [];
        if (count($cars) == 0) {
            foreach ($sentences as &$sentence) {
                if ($sentence["used"]) continue;

                $preparedANZ = $this-
>prepareANZ($sentence["text"]);

                $preparedANZ = empty($preparedANZ) ?
"Unrecognized" : $preparedANZ;

                if ($preparedANZ !== "Unrecognized") {
                    $carContent = ["name" => "Unidentified",
                        "vertices" =>
                            [
                                ["x" => 0, "y" => $height],
                                ["x" => $width, "y" => $height],
                                ["x" => $width, "y" => 0],
                                ["x" => 0, "y" => 0]
                            ],
                        "entities" =>
                            [
                                "text" =>
                                    [
                                        [
                                            "description" => $preparedANZ,
                                            "vertices" =>
                                                [
                                                    ["x" => 0, "y" => $height],
                                                    ["x" => $width, "y" =>
$height],
                                                    ["x" => $width, "y" => 0],
                                                    ["x" => 0, "y" => 0]
                                                ]
                                        ]
                                    ],
                                "anz" => $preparedANZ
                            ];
                    $cars[] = $carContent;
                    $sentence["used"] = true;
                }

            }
            unset($sentence);
```

```
        }

        foreach ($cars as &$car) {
            if (isset($car["entities"])) {
                if (!(isset($car["anz"]) && $car["anz"] !==
"Unrecognized")) {
                    $anz = "";
                    if (isset($car["entities"]["licensePlate"])) {
                        if (isset($car["entities"]["textInLicensePlate"]))
                            $anz = $this-
>getSentenceBySequenceOfWords($sentences, $this-
>getWordsFromCar($car["entities"]["textInLicensePlate"]));
                    } else {
                        if (isset($car["entities"]["text"]))
                            $anz = $this-
>getSentenceBySequenceOfWords($sentences, $this-
>getWordsFromCar($car["entities"]["text"]));
                    }

                    $preparedANZ = $this->prepareANZ($anz);

                    $preparedANZ = empty($preparedANZ) ?
"Unrecognized" : $preparedANZ;

                    $car["anz"] = $preparedANZ;
                }

                if (isset($car["anz"]) && $car["anz"] !==
"Unrecognized") {
                    $carsResponse[] = $car;
                }
            }
        }
        unset($car);
        return $carsResponse;
    }

    public function getANZ(Request $request)
    {
        $content = $request->string("content");
        $isFile = $request->boolean("isFile");
        $data = $this->getANZFromImage($content, $isFile);
        return response()->json($data, 200);
    }
```

UNICHECK
by Turnitin

Ім'я користувача:
Кафедра КІ

ID перевірки:
1016312432

Дата перевірки:
02.06.2024 23:07:40 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
03.06.2024 06:43:13 EEST

ID користувача:
100005591

Назва документа: Авсієвич_Кіберфізична система "Розумна парковка" з розпізнаванням моделі та номера ав...

Кількість сторінок: 71 **Кількість слів:** 12064 **Кількість символів:** 94885 **Розмір файлу:** 2.08 MB **ID файлу:** 1016109342

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

16.8%
Схожість

Найбільша схожість: 6.78% з джерелом з Бібліотеки (ID файлу: 1014562924)

16.2% Джерела з Інтернету | 743 ..... Сторінка 73

8.91% Джерела з Бібліотеки | 152 ..... Сторінка 78

1.45% Цитат

Цитати | 9 ..... Сторінка 79

Не знайдено жодних посилань

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи | 1

Підозріле форматування | 15 сторінок

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 4.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 15%**

ID: 128181 Назва: БКР Кіберфізична система “Розумна парковка” з розпізнаванням моделі та номера автомобіля Додано в БД: 2024-06-03 Автора: В. Р. Авсієвич Керівники: О. О. Павлова Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	78583	754	4006 (5%)	46 (6%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Авсієвич Володимир Русланович

Тема: Кіберфізична система "Розумна парковка" з розпізнаванням моделі та номера автомобіля.

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг дипломної роботи:

Кількість сторінок записки 65 с.

1. Короткий зміст роботи та прийнятих рішень: розроблено систему розпізнавання номерного знаку та моделі автомобіля із застосуванням моделі нейронної мережі та машинного зору.

2. Висновок про відповідність роботи дипломному завданню: Робота повністю відповідає поставленому завданню.

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У першому розділі проаналізовано існуючі рішення у галузі розумних парковок із розпізнаванням моделей та номерних знаків автомобіля. У другому розділі розглянуто апаратну та програмну реалізацію пропонованої у роботі системи. У третьому розділі запропоновано архітектуру та структурну схему систем розпізнавання номерного знаку та моделі автомобіля, а також реалізацію клієнтської частини у вигляді вебсайту та мобільного застосунку.

4. Позитивні сторони роботи: висока практична цінність, наявність готових технічних рішень.

5. Негативні сторони роботи:

6.Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка та графічний матеріал оформлені коректно, згідно діючих стандартів оформлення документації.

7. Відгук про роботу в цілому: Робота виконана на високому технічному рівні.

8. Інші зауваження: _____

9. Оцінка дипломної роботи: відмінно.

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) _____

Мартинюк Валерій Володимирович, зав. каф. АІЕТтаР

"23" 05 2024 р. _____ (підпис)

Завідувачу кафедри КІІС
д-р.техн.наук, проф. Говорущенко Т. О.

Авсієвича Володимира Руслановича
ПІБ здобувача вищої освіти

ФІТ, 4 курсу, групи КІ2-20-1

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений(а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

22 квітня 2024 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Кіберфізична система "Розумна парковка" з розпізнаванням моделі та номера автомобіля

Автор: Авсієвич Володимир Русланович

Спеціальність: 123– Комп'ютерна інженерія

Освітня програма: освітньо-професійна

Науковий керівник: Павлова Ольга Олександрівна, д. ф., доц. каф. КІІС

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) запозичення розміщені в розділах аналізу існуючих аналогів та прототипів, які не описують безпосередньо авторське дослідження і не стосуються результатів роботи;

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості, складає 16.8% і адресується до 895 першоджерела, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ О. О. Павлова

Гарант ОП _____ С.М. Лисенко

Завідувач кафедри КІІС _____ Т. О. Говорущенко