

Міністерство освіти і науки України
Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки

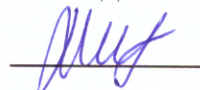
ПОЯСНЮВАЛЬНА ЗАПИСКА

до випускної кваліфікаційної роботи бакалавра
на тему: «Проект програмно-апаратного комплексу для
моніторингу паркувальних місць у місті з використан-
ням мережевих та IoT-технологій»

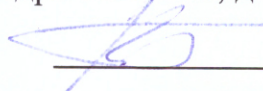
Виконав студент 4-го курсу групи КІ-21-1
спеціальності 123 «Комп'ютерна інженерія»

 Назар ЗІНЕВИЧ

Керівник старший викладач кафедри КІ та КБ

 Константин МАТВЕЄВ

Професор кафедри КІ та КБ, д.т.н., с.н.с.

 Сергій КОВБАСЮК

Житомир – 2025

Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки
Спеціальність 123 «Комп'ютерна інженерія»
Освітня програма «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач
кафедри комп'ютерної
інженерії та кібербезпеки
 Андрій Єфіменко
1 березня 2025 р.

ЗАВДАННЯ

на випускню кваліфікаційну роботу бакалавра

Студент: Зіневич Назар Ярославович

Тема роботи: «Проект програмно-апаратного комплексу для моніторингу паркувальних місць у місті з використанням мережевих та IoT-технологій»

затверджена Наказом університету від 5 березня 2025 р. № 97

Термін здачі студентом закінченої роботи 9 червня 2025 р.

Вихідні дані роботи (зазначається предмет і об'єкт дослідження):

Об'єктом дослідження є процес моніторингу та управління зайнятістю паркувальних місць. Предметом дослідження виступають методи та програмні засоби для розробки комплексу автоматичного визначення стану паркомісць на основі візуального аналізу та взаємодії через Telegram-бот.

Обладнання та програмне забезпечення:

1. Персональний комп'ютер (ОС Windows або аналогічна)
2. Python 3.x (з PyCharm IDE)
3. TensorFlow (Keras API), OpenCV, SQLite

Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Бродський Ю.Б.		

Студент Назар ЗІНЕВИЧ

Керівник Константин МАТВЄЄВ

Календарний план

№ з/п	Назви етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Аналіз та уточнення завдання	17.03.2024	Виконано
2	Аналіз літератури та існуючих рішень	24.03.2024	Виконано
3	Формулювання вимог та вибір технологій	02.04.2024	Виконано
4	Розробка архітектури системи	04.04.2024	Виконано
5	Проектування модуля розпізнавання (CNN)	06.04.2024	Виконано
6	Проектування Telegram-бота та візуалізації	21.04.2024	Виконано
7	Підготовка середовища та даних (PKLot)	04.05.2024	Виконано
8	Розробка та навчання моделі CNN	13.05.2024	Виконано
9	Розробка Telegram-бота та інтеграція БД	18.05.2024	Виконано
10	Тестування MVP та аналіз результатів	26.05.2024	Виконано
11	Оформлення пояснювальної записки	29.05.2024	Виконано
12	Оформлення презентації	09.06.2024	Виконано
13	Захист роботи	18.06.2024	Виконано

Студент Назар ЗІНЕВИЧ

Керівник Константин МАТВЄЄВ

АНОТАЦІЯ

Кваліфікаційна робота бакалавра містить відомості про проект, розроблений у межах даної роботи. Обсяг пояснювальної записки становить 73 сторінки, включає 39 рисунки та 1 таблиця. Список використаних джерел налічує 20 найменувань і розміщений на двох сторінках.

Метою роботи є розробка та дослідження прототипу програмно-апаратного комплексу для автоматизованого моніторингу паркомісць на основі аналізу візуальної інформації (CNN) та взаємодії з користувачами через Telegram-бот.

В результаті роботи розроблено та досліджено прототип комплексу, що включає: модуль розпізнавання паркомісць на базі CNN, функціональний Telegram-бот, модуль генерації карти парковки та інтеграцію даних через SQLite.

Отримані результати демонструють ефективність CNN у класифікації паркомісць, забезпечують зручний доступ користувачів до інформації та підтверджують коректну взаємодію модулів MVP через комплексне тестування.

Результати роботи сприятимуть оптимізації використання паркувального простору та є основою для подальшого розвитку систем управління парковками, включаючи інтеграцію з реальними камерами, розширення функціоналу та розгортання в хмарній архітектурі.

Ключові слова: АВТОМАТИЗОВАНИЙ МОНІТОРИНГ, ЗГОРТКОВА НЕЙРОННА МЕРЕЖА, КОМП'ЮТЕРНИЙ ЗІР, ПАРКУВАЛЬНІ МІСЦЯ, TELEGRAM-БОТ.

					ККІК<.КР.Б – 123 – 4 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Зіневич Н.Я.		16.08.25	Проект програмно-апаратного комплексу для моніторингу паркувальних місць у місті з використанням мережевих та IoT-технологій	Літ.	Арк.	Аркуші
Керівник		Матвеев К.І.		16.08.25			4	71
Реценз.		Ковбасюк С.В.		16.08.25		ФІКТ, Гр. КІ-21-1		
Н. контр.		Бродський Ю.Б.		16.08.25				
Затверд.		Єфіменко А.А.		16.08.25				

ANNOTATION

The bachelor's qualification thesis contains information about the project developed within this work. The explanatory note volume is 73 pages, including 39 figures and 1 table. The list of references includes 20 titles and is arranged on two pages.

The aim of the work is the development and research of a prototype hardware-software complex for automated monitoring of parking spaces based on visual information analysis (CNN) and user interaction via a Telegram bot.

As a result of the work, a prototype of the complex was developed and researched, which includes: a parking space recognition module based on CNN, a functional Telegram bot, a parking map generation module, and data integration via SQLite.

The obtained results demonstrate the effectiveness of CNN in classifying parking spaces, provide users with convenient access to information, and confirm the correct interaction of MVP modules through comprehensive testing.

The results of the work will contribute to the optimization of parking space usage and serve as a basis for the further development of parking management systems, including integration with real cameras, functionality expansion, and deployment in a cloud architecture.

Keywords: AUTOMATED MONITORING, COMPUTER VISION, CONVOLUTIONAL NEURAL NETWORK, PARKING SPACES, TELEGRAM BOT.

		<i>Зіневич Н.Я.</i>			<i>ККІКБ.КР.Б – 123 – 4 – ПЗ</i>	<i>Арк.</i>
		<i>Матвеев К.І.</i>				
<i>Змн.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		<i>5</i>

ЗМІСТ

ВСТУП.....	9
РОЗДІЛ 1 АНАЛІЗ ПРОБЛЕМИ МОНІТОРИНГУ ПАРКУВАЛЬНИХ МІСЦЬ ТА ОБҐРУНТУВАННЯ ТЕХНОЛОГІЧНИХ РІШЕНЬ	11
1.1 Огляд існуючих підходів та технологій для моніторингу паркувальних місць (датчики, системи відеоспостереження, IoT-технології)	11
1.2 Аналіз застосування нейронних мереж для розпізнавання зайнятості паркомісць	16
1.3 Формулювання функціональних та нефункціональних вимог до програмно-апаратного комплексу	21
1.4 Обґрунтування вибору інструментальних засобів та технологій для реалізації проєкту	23
РОЗДІЛ 2 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ КОМПОНЕНТІВ ПРОГРАМНО-АПАРАТНОГО КОМПЛЕКСУ	27
2.1 Розробка загальної архітектури програмно-апаратного комплексу	27
2.2 Проєктування модуля розпізнавання зайнятості паркомісць	31
2.3 Проєктування модуля взаємодії з користувачем через Telegram-бот.....	34
2.4 Проєктування системи візуалізації та інтеграції даних про статуси паркомісць	36
РОЗДІЛ 3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОТОТИПУ СИСТЕМИ МОНІТОРИНГУ.....	41
3.1 Реалізація модуля розпізнавання зайнятості паркомісць та його функціональне тестування	41
3.2 Реалізація модуля Telegram-бота та його функціональне тестування	49
3.3 Реалізація статичної системи візуалізації паркувальних місць та її інтеграція	57
3.4 Комплексне тестування взаємодії модулів MVP, опис користувацького інтерфейсу та ключові аспекти для демонстрації працездатності.....	60
ВИСНОВКИ	68
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ	69
ДОДАТКИ	71

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API (Application Programming Interface) – Інтерфейс прикладного програмування

CNN (Convolutional Neural Network) – Згорткова нейронна мережа

COCO (Common Objects in Context) – Загальні об'єкти в контексті (назва датасету)

FN (False Negatives) – Хибно негативні спрацювання

FP (False Positives) – Хибно позитивні спрацювання

FPN (Feature Pyramid Network) – Пірамідальна мережа ознак

FPS (Frames Per Second) – Кадри за секунду

HTTP (HyperText Transfer Protocol) – Протокол передачі гіпертексту

IoT (Internet of Things) – Інтернет речей

IoU (Intersection over Union) – Перетин над об'єднанням (метрика)

JSON (JavaScript Object Notation) – Об'єктна нотація JavaScript (формат даних)

LoRaWAN (Long Range Wide Area Network) – Глобальна мережа великого радіусу дії

Mask R-CNN (Mask Region-based Convolutional Neural Network) - Згорткова нейронна мережа на основі регіонів з маскуванням

mAP (mean Average Precision) – Середня точність (метрика)

MQTT (Message Queuing Telemetry Transport) – Протокол обміну повідомленнями для телеметрії

MVP (Minimum Viable Product) – Мінімально життєздатний продукт

NB-IoT (Narrowband Internet of Things) – Вузькосмуговий Інтернет речей

OpenCV (Open Source Computer Vision Library) – Відкрита бібліотека комп'ютерного зору

PIL (Python Imaging Library) – Бібліотека обробки зображень Python (Pillow – її активний форк)

PNG (Portable Network Graphics) – Портативна мережева графіка (формат файлів зображень)

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

R-CNN (Region-based Convolutional Neural Network) - Згорткова нейронна мережа на основі регіонів

ReLU (Rectified Linear Unit) – Випрямлений лінійний блок (функція активації)

ROI (Region of Interest) – Область інтересу

SQLite – Полегшена система управління базами даних на основі SQL

SSD (Single Shot MultiBox Detector) – Одноетапний детектор з кількома рамками

TN (True Negatives) – Істинно негативні спрацювання

TP (True Positives) – Істинно позитивні спрацювання

VPS (Virtual Private Server) – Віртуальний приватний сервер

YOLO (You Only Look Once) – Ти дивишся лише раз (архітектура нейронної мережі)

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Проблема ефективного управління паркувальними місцями в умовах зростаючої автомобілізації міст є надзвичайно актуальною. Труднощі з пошуком вільних місць призводять до збільшення трафіку, марної витрати часу та пального, а також негативно впливають на екологію. Розробка інтелектуальних систем моніторингу, здатних автоматизувати контроль зайнятості та надавати інформацію в реальному часі, є важливим кроком до оптимізації міської інфраструктури. Практичне значення даної роботи полягає у створенні програмно-апаратного комплексу, що вирішує ці завдання, пропонуючи користувачам зручний доступ до інформації про парковку.

Метою кваліфікаційної роботи є розробка та дослідження прототипу програмно-апаратного комплексу для автоматизованого моніторингу зайнятості паркувальних місць на основі аналізу візуальної інформації з використанням згорткової нейронної мережі та забезпечення взаємодії з користувачами через Telegram-бот.

Для досягнення мети поставлено наступні завдання: проаналізувати існуючі технології моніторингу; обґрунтувати вибір архітектури нейронної мережі; сформулювати вимоги до системи; розробити загальну архітектуру комплексу, включаючи концепцію масштабованого розгортання; спроектувати та реалізувати модулі розпізнавання, взаємодії з користувачем (Telegram-бот), візуалізації та інтеграції даних; провести комплексне тестування розробленого прототипу.

Об'єктом дослідження є процес моніторингу та управління зайнятістю паркувальних місць. Предметом дослідження є методи та програмні засоби для розробки комплексу автоматичного визначення стану паркомісць на основі візуального аналізу та взаємодії через Telegram-бот.

У роботі застосовано методи аналізу науково-технічної літератури, системного аналізу, комп'ютерного зору та машинного навчання (зокрема, CNN), об'єктно-орієнтованого проєктування та програмування, а також тестування програмного забезпечення. Інформаційною базою слугували наукові публікації, технічна документація та відкритий датасет PKLot.

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвєєв К.І.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

Практичне значення одержаних результатів полягає у створенні прототипу системи, що автоматизує моніторинг парковок та надає користувачам зручний доступ до інформації через Telegram, включаючи концептуальне бронювання та оплату. Це може сприяти оптимізації використання паркувального простору. Запропоновані рішення можуть слугувати основою для подальшого розвитку систем управління парковками, враховуючи аспекти безпеки даних.

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1 АНАЛІЗ ПРОБЛЕМИ МОНІТОРИНГУ ПАРКУВАЛЬНИХ МІСЦЬ ТА ОБҐРУНТУВАННЯ ТЕХНОЛОГІЧНИХ РІШЕНЬ

1.1 Огляд існуючих підходів та технологій для моніторингу паркувальних місць (датчики, системи відеоспостереження, IoT-технології)

У сучасних містах ефективно управління паркувальними місцями є важливою складовою інфраструктури. Розглянемо основні підходи та технології, що застосовуються для моніторингу зайнятості паркомісць

Системи моніторингу паркувальних місць поділяються на кілька основних категорій:

- Сенсорні системи – базуються на використанні різних типів датчиків для визначення наявності автомобіля на конкретному місці;
- Відеоаналітичні системи – застосовують камери та алгоритми комп'ютерного зору для аналізу зайнятості паркомісць;
- IoT-рішення – інтегрують сенсорні або відеоаналітичні підходи з хмарними платформами, мобільними додатками та протоколами передачі даних для централізованого збору, обробки й візуалізації інформації;
- Комбіновані системи – поєднують декілька підходів для підвищення точності та надійності.

Сенсорні технології для детекції автомобілів:

1. Ультразвукові датчики:

Принцип роботи полягає у вимірюванні часу проходження ультразвукової хвилі до об'єкта (автомобіля) та назад. Переваги – невисока вартість, простота монтажу, достатня точність у закритих приміщеннях. Недоліки – чутливість до погодних умов (дощ, сніг), можливість хибних спрацювань через сміття чи інші перешкоди. Застосовуються переважно на критих паркінгах або у місцях із контрольованим середовищем [1].

2. Інфрачервоні (ІЧ) датчики

Використовують ІЧ-промені для фіксації наявності об'єкта. Переваги – низьке енергоспоживання, простота інтеграції. Недоліки – зниження точності при

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		11

потраплянні прямих сонячних променів, пилу чи снігу. Застосовуються для контролю окремих місць або в'їздів [2].

3. Магнітометричні/індуктивні датчики

Визначають зміну магнітного поля при появі металевого об'єкта (автомобіля). Переваги – висока точність, стійкість до погодних умов, можливість автономної роботи. Недоліки – складність монтажу (зазвичай вбудовуються у дорожнє покриття), потреба у калібруванні. Приклад – сенсори Urbiotica U-Spot 3.0, які використовують LoRaWAN/NB-IoT для передачі даних і забезпечують довгий термін роботи завдяки енергоефективності [3].

4. Радарні/мікрохвильові датчики

Застосовують мікрохвильове випромінювання для виявлення руху та присутності транспортних засобів. Переваги – нечутливість до погодних умов, можливість детекції на великій відстані, ігнорування пішоходів (наприклад, ORTEX Viik Gate). Недоліки – вища вартість, потреба в налаштуванні для уникнення хибних спрацювань [4].

Таблиця 1.1

Порівняльна таблиця основних типів датчиків

Тип датчика	Переваги	Недоліки	Приклади застосування
Ультразвуковий	Дешевизна, простота монтажу	Чутливість до погодних умов	Криті паркінги
Інфрачервоний	Низьке енергоспоживання	Зниження точності на сонці, пилу	В'їзди, окремі місця
Магнітометричний	Висока точність, автономність	Складність монтажу, потреба калібрування	Відкриті паркінги, IoT-системи
Радарний/мікрохвильовий	Стійкість до погодних умов, ігнорування пішоходів	Вартість, налаштування	В'їзди, логістичні центри

Системи моніторингу на основі аналізу відеозображень, принципом роботи є , що відеоаналітичні системи використовують стаціонарні камери, які охоплюють

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		12

кілька паркомісць. Зображення або відеопотік обробляються алгоритмами комп’ютерного зору, зокрема глибокого навчання (наприклад, CNN, YOLO, Mask R-CNN), для визначення зайнятості місць, а також, за необхідності, розпізнавання номерних знаків та типу транспортного засобу.

Перевагою є можливість покриття великої площі однією камерою. Отримання додаткової інформації (розпізнавання номерних знаків, класифікація авто).Гнучкість у масштабуванні та інтеграції з іншими системами (наприклад, відеонагляд, безпека).

Недоліками є вартість встановлення та обслуговування камер і серверів для обробки відео. Вимоги до якісного освітлення та погодних умов (дощ, сніг, туман можуть знижувати точність). Обчислювальна складність алгоритмів, потреба у потужному обладнанні. Проблеми з приватністю та доступом до міських камер, особливо в Україні, де доступ до відеопотоків часто обмежений.

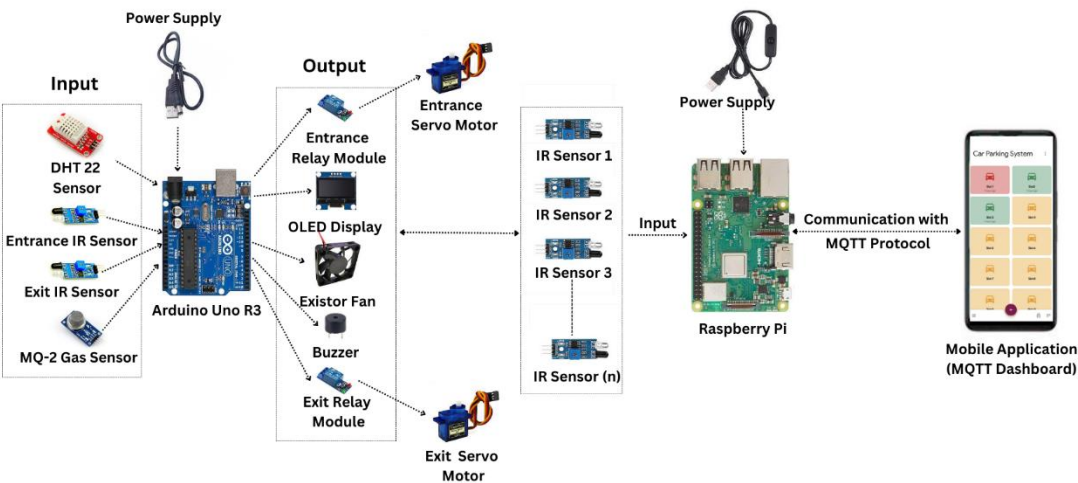


Рис. 1.1. Архітектурна схема IoT-рішення для моніторингу паркувальних місць

Застосування IoT-технологій у системах моніторингу парковок:

Типова архітектура IoT-рішення

- Сенсорний рівень: датчики (ультразвукові, магнітометричні, ІЧ, радарні), які збирають дані про зайнятість місця;
- Мережевий рівень: шлюзи (гейтвеї), що збирають дані з датчиків і передають їх далі;

- Рівень обробки: сервери, хмарні платформи, бази даних, де відбувається аналіз та зберігання інформації;

- Рівень додатків: мобільні чи веб-застосунки для користувачів і адміністраторів

Протоколи передачі даних:

- LoRaWAN: велика дальність, низьке енергоспоживання, підходить для міських умов, інтегрується з різними мережами;

- NB-IoT: працює у ліцензованих діапазонах, забезпечує високу якість сервісу, довгий термін роботи пристроїв, використовується у великих містах (наприклад, China Mobile Smart Parking) [5];

- Sigfox: низька пропускна здатність, але висока енергоефективність;

- Wi-Fi, Bluetooth Low Energy: для локальних мереж, обмежена дальність, вища енергозатратність.

Перевагою IoT-підходу є масштабованість і гнучкість, віддалений моніторинг і управління, збір великих обсягів даних для аналітики та оптимізації, енергоефективність пристроїв, можливість автономної роботи до 10 років.

Приклади застосування

- Urbiotica U-Spot 3.0 (LoRaWAN, NB-IoT, Bluetooth) – автономний сенсор із хмарною калібровкою та інтеграцією у будь-яку мережу [3];

- China Mobile Smart Parking – NB-IoT-система з детекцією зайнятості, розпізнаванням номерних знаків, мобільними платежами та навігацією для водіїв [5];

- Інтегровані рішення з мобільними додатками, які показують у реальному часі наявність вільних місць, дозволяють бронювати паркомісця та здійснювати оплату.

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				14
Змн.	Арк.	№ докум.	Підпис	Дата		



Рис. 1.2. Фото реального IoT-сенсора на паркінгу

Огляд існуючих комерційних та наукових проєктів

1. Urbiotica U-Spot 3.0

Використовує магнітометричний сенсор, підтримує LoRaWAN/NB-IoT, має автономність до 10 років, інтегрується у будь-яку мережу, забезпечує точне визначення зайнятості місця навіть у складних умовах (наприклад, підземні паркінги) [3].

2. China Mobile Smart Parking

Масштабована система на основі NB-IoT, включає детекцію зайнятості, розпізнавання номерних знаків, мобільні платежі, навігацію для водіїв і централізоване управління для міста. Забезпечує високу якість сервісу завдяки використанню ліцензованих частот і довготривалій автономній роботі сенсорів [5].

3. OPTEX Viik Gate

Використовує поєднання мікрохвильових і ультразвукових технологій, не потребує монтажу в дорожнє покриття, забезпечує стійкість до погодних умов і точність детекції, ігнорує пішоходів, що знижує кількість хибних спрацювань [4].

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвєєв К.І.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

4. Smart Parking Systems (TIMES24 Malaysia)

Інтегрує відеоаналітику (розпізнавання номерних знаків), IoT-сенсори та мобільні додатки для моніторингу зайнятості, навігації, безконтактної оплати й аналітики для операторів паркінгу [6].

Загальна тенденція розвитку полягає у переході до комбінованих систем, які інтегрують сенсорні та відеоаналітичні підходи з IoT-платформами, використанні штучного інтелекту для підвищення точності розпізнавання та автоматизації процесів. Це дозволяє створювати гнучкі, масштабовані та ефективні системи моніторингу паркувальних місць, що відповідають вимогам сучасних міст.

1.2 Аналіз застосування нейронних мереж для розпізнавання зайнятості паркомісць

Нейронні мережі, зокрема глибокі згорткові нейронні мережі (CNN), стали ключовим інструментом для автоматичного аналізу візуальної інформації з паркувальних майданчиків. Їхня здатність навчатися на великих обсягах даних, розпізнавати складні патерни та адаптуватися до різних умов дозволяє ефективно вирішувати задачу визначення зайнятості паркомісць на основі зображень або відео.

CNN складаються зі згорткових шарів, шарів підвибірки (пулінгу) та повнозв'язних шарів. Згорткові шари автоматично виділяють ознаки (features) з вхідних зображень, що робить CNN особливо ефективними для задач комп'ютерного зору, включаючи класифікацію, детекцію та сегментацію об'єктів [7].

Популярні архітектури для детекції та сегментації:

1. YOLO (You Only Look Once, v3–v11): Одноетапна архітектура для швидкої детекції об'єктів, що дозволяє визначати зайнятість паркомісць у режимі реального часу. Переваги — висока швидкість і достатня точність, недоліки — іноді менша точність при складних перекриттях чи малих об'єктах [8].

2. SSD (Single Shot MultiBox Detector): Також одноетапний детектор, добре підходить для мобільних пристроїв, забезпечує баланс між швидкістю та точністю.

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				16
Змн.	Арк.	№ докум.	Підпис	Дата		

3. Faster R-CNN, R-CNN, Faster R-CNN FPN: Двоетапні архітектури з високою точністю, особливо ефективні для складних сцен із перекриттями та різними ракурсами, але потребують більше обчислювальних ресурсів [9].

4. Mask R-CNN: Розширення Faster R-CNN для сегментації, дозволяє визначати точні межі зайнятих паркомісць, а не лише їхню наявність. Це особливо корисно для паркінгів зі складною розміткою або нестандартними формами місць.

5. Кастомні CNN: Для задачі класифікації окремих слотів (вільний/зайнятий) використовують простіші CNN, які працюють із вирізаними ділянками зображення кожного паркомісця

Таблиця 1.2

Порівняння архітектур

Архітектура	Сильні сторони	Слабкі сторони	Релевантність для парковок
YOLO (v3–v11)	Висока швидкість, реальний час	Може втрачати точність на малих об'єктах	Оптимально для онлайн-моніторингу
SSD	Баланс швидкість/точність, мобільність	Менш точний для складних сцен	Мобільні та edge-рішення
Faster R-CNN	Висока точність, робота з перекриттями	Високі вимоги до ресурсів	Складні сцени, офлайн-аналітика
Mask R-CNN	Сегментація, точні межі	Вимогливість до ресурсів	Аналіз складних паркувальних зон
Кастомні CNN	Простота, швидкість	Локальна робота, потрібна сегментація	Окремі слоти, прості задач

Трансферне навчання дозволяє використовувати вже навчені на великих датасетах моделі (наприклад, ImageNet, COCO) як базу, яку донавчають на менших спеціалізованих наборах даних для парковок. Це значно зменшує потребу у великих датасетах і прискорює навчання [10].

Перевагою є зменшення потреби у власних даних, прискорення навчання, підвищення точності, особливо у випадках з обмеженою кількістю розмічених зображень.

Прикладом використання є YOLOv5, ResNet50, MobileNet, VGG для детекції автомобілів і класифікації стану місця.

Вимогою до датасетів є велика кількість зображень різних паркінгів, ракурсів, погодних і освітлювальних умов та різноманітність типів автомобілів, щільності паркування, перекриттів.

Публічні датасети:

1. PKLot: Понад 695 тис. зображень слотів із різних паркінгів, різні погодні умови, багато ракурсів [11].

2. CNRPark-EXT: 150 тис. зображень із 9 камер, охоплює різні кути, освітлення, часткові перекриття, тіні [11].

3. Cityscapes: Містить сегментовані зображення міських сцен, може використовуватись для допоміжних задач.

Збір і розмітка власних даних: витяг з відео або фото, ручна розмітка слотів (bounding boxes, маски) та аугментація: повороти, масштабування, зміна яскравості/контрасту, дзеркальне відображення для підвищення стійкості моделі до змін умов

Критерії та метрики оцінки якості роботи нейронних мереж:

- Точність (Precision): частка правильно визначених зайнятих/вільних місць серед усіх визначених як зайняті/вільні.
- Повнота (Recall): частка правильно знайдених зайнятих/вільних місць серед усіх реальних зайнятих/вільних.
- F1-score: гармонічне середнє між точністю і повнотою.
- Intersection over Union (IoU): ступінь перекриття передбаченої та реальної області (важливо для сегментації та детекції).
- Mean Average Precision (mAP): середня точність по всіх класах і порогах IoU — ключова метрика для задач детекції.
- FPS (Frames per Second): швидкість обробки кадрів, важлива для реального часу.

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				18
Змн.	Арк.	№ докум.	Підпис	Дата		

Для систем моніторингу паркувальних місць критичними є висока точність (щоб уникнути хибних спрацювань), достатня повнота (щоб не пропускати зайняті місця) та швидкодія для оперативного оновлення інформації.

Проблеми та виклики при використанні нейронних мереж для моніторингу паркомісць є зміни освітлення (день/ніч, сонце/тіні): можуть суттєво впливати на якість розпізнавання. Погодні умови (дощ, сніг, туман, бруд): знижують якість зображень, ускладнюють детекцію, часткове перекриття автомобілів (occlusion): ускладнює визначення статусу місця, особливо при щільному паркуванні. Варіативність розмірів і типів авто: потребує універсальних моделей або адаптації під конкретний паркінг. Різні конфігурації паркінгів і ракурси камер: вимагають гнучких моделей, часто — специфічного донавчання та мінімізація впливу факторів: специфічна аугментація, використання мультисенсорних даних (наприклад, поєднання з IoT-датчиками), доменна адаптація.



Рис. 1.3. Паркінг (Сонячна погода)

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		19



Рис. 1.4. Паркінг (Хмарна погода)

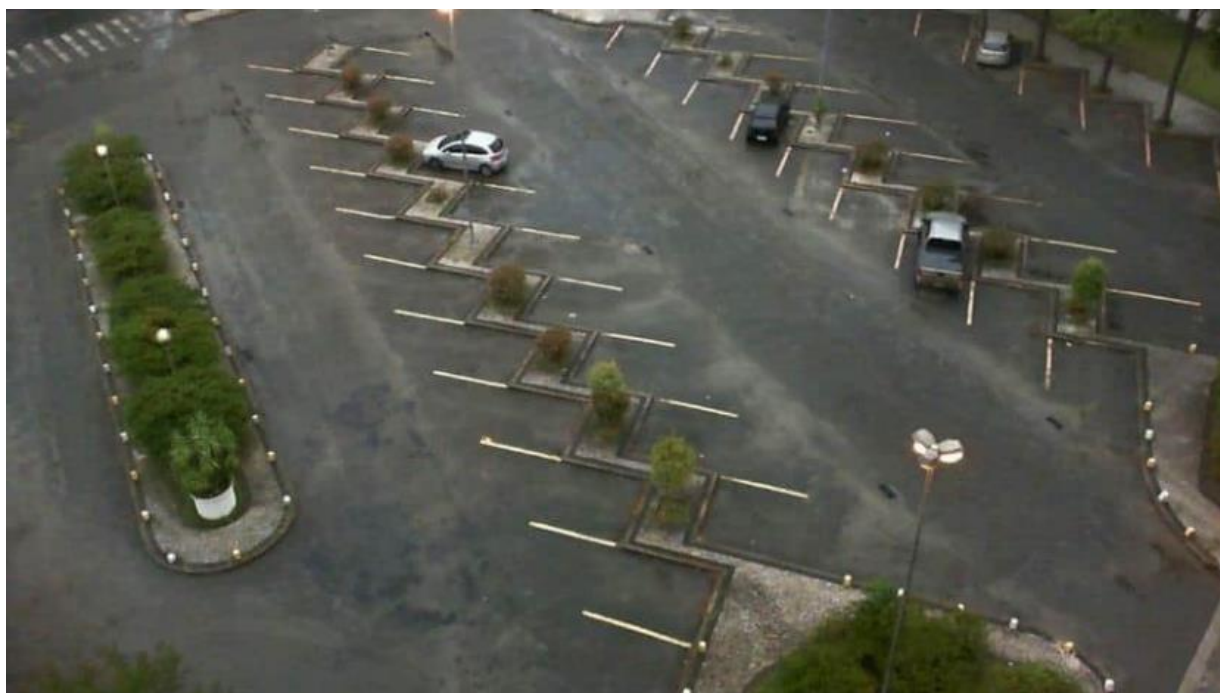


Рис. 1.5. Паркінг (Дощова погода)

Застосування нейронних мереж, особливо сучасних архітектур CNN, дозволяє автоматизувати моніторинг паркувальних місць із високою точністю, гнучкістю та масштабованістю. Основні переваги — здатність працювати в різних умовах, адаптація до нових локацій через трансферне навчання, можливість роботи в реальному часі. Водночас, для досягнення стабільної якості необхідна якісна розмітка даних, врахування специфіки місця та регулярне донавчання моделей. Для MVP-проєкту доцільно використати готові архітектури (YOLO, Mask R-CNN) із

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвєєв К.І.				20
Змн.	Арк.	№ докум.	Підпис	Дата		

трансферним навчанням на відкритих датасетах (PKLot, CNRPark-EXT), що дозволить швидко продемонструвати працездатність концепції навіть на обмежених даних.

1.3 Формулювання функціональних та нефункціональних вимог до програмно-апаратного комплексу

Цей підрозділ присвячений формулюванню функціональних та нефункціональних вимог до розроблюваного програмно-апаратного комплексу. Ці вимоги визначатимуть його основні можливості, характеристики та обмеження в рамках створення мінімально життєздатного продукту (MVP).

Функціональні вимоги визначають, які завдання та функції має виконувати система.

Центральним елементом комплексу є модуль розпізнавання зайнятості паркомісць. Його основне завдання – автоматично визначати статус кожного паркувального місця на основі аналізу візуальних даних, таких як статичні фотографії або окремі кадри з відео, з використанням алгоритмів нейронних мереж. Система повинна розрізняти щонайменше два основні статуси: "вільно" та "зайнято". Вхідними даними для цього модуля будуть зображення окремих паркомісць або загальні кадри паркувальної зони, з яких система виділятиме окремі слоти. Для MVP планується використовувати набір тестових зображень, що імітують різні умови освітлення та погоди, подібно до даних, представлених у публічних датасетах, як-от PKLot або CNRPark-EXT. Наприклад, ці зображення можуть демонструвати паркувальні майданчики за сонячної, хмарної або дощової погоди (рис. 1.3-5). Вихідними даними модуля буде структурована інформація, що містить ідентифікатор кожного паркомісця та його поточний визначений статус.

Для взаємодії з кінцевим користувачем передбачено **модуль Telegram-бота**. Бот повинен надавати користувачам актуальну інформацію про наявність вільних паркувальних місць. Серед основних команд, які підтримуватиме бот, – запит на перегляд кількості та розташування вільних місць. Інформація може подаватися у вигляді текстового повідомлення або шляхом відображення номерів вільних місць на статичній схемі парковки. Концептуально, бот може підтримувати функцію

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		21

бронювання місця на певний час та скасування бронювання. Функція "оплати" в рамках MVP буде імітована – наприклад, бот може видавати повідомлення про успішну концептуальну оплату або перенаправлення на платіжний сервіс без реального проведення транзакції.

Інформація про стан паркувальних місць буде візуалізуватися за допомогою статичної карти (схеми) парковки. На цій схемі кожне паркомісце матиме унікальний ідентифікатор, а його статус ("вільно", "зайнято", концептуально "заброньовано") буде позначатися відповідним кольоровим кодуванням для інтуїтивного сприйняття [12]. Передбачається, що дана візуалізація буде оновлюватися на основі даних, що надходять від модуля розпізнавання, хоча для MVP частота оновлення може бути обмеженою.

Концептуально, система повинна підтримувати функцію бронювання паркомісць. Це означає можливість зміни статусу обраного місця на "заброньовано" на визначений період часу, а також зберігання цієї інформації. Хоча повна реалізація бронювання виходить за межі MVP, основні механізми зміни статусу та обліку заброньованих місць будуть закладені. Також, у концепції передбачено ведення базової статистики використання парковки, наприклад, збір даних про завантаженість у різні години дня або дні тижня, що в майбутньому може бути використано для аналітики та оптимізації роботи паркувального майданчика [1].

Нефункціональні вимоги описують атрибути якості та обмеження системи, тобто те, як система повинна працювати.

Вимоги до продуктивності для MVP включають орієнтовний час розпізнавання статусу одного паркомісця на тестових статичних зображеннях (наприклад, до декількох секунд на зображення, залежно від складності моделі та апаратного забезпечення). Час відповіді Telegram-бота на стандартні запити користувача має бути мінімальним, в межах кількох секунд. Система в рамках MVP повинна бути здатна обробляти інформацію для невеликої кількості паркомісць, наприклад, 10-20 місць, що відповідатиме тестовій схемі парковки [13].

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		22

Хоча MVP не передбачає повномасштабного розгортання, масштабованість закладається концептуально. Це означає, що архітектура системи (наприклад, використання модульного підходу та брокера повідомлень) повинна теоретично дозволяти розширення на більшу кількість паркомісць або навіть декілька паркувальних зон у майбутньому

1.4 Обґрунтування вибору інструментальних засобів та технологій для реалізації проєкту

Для реалізації програмно-апаратного комплексу моніторингу паркувальних місць в рамках мінімально життєздатного продукту (MVP) було обрано набір інструментальних засобів та технологій, що найкраще відповідають вимогам до простоти розробки, швидкості прототипування та доступності ресурсів. Вибір ґрунтується на аналізі існуючих рішень та сформульованих функціональних і нефункціональних вимогах. Акцент зроблено на використанні найпростіших та найдоступніших інструментів для досягнення поставленої мети MVP.

Для розробки всіх програмних компонентів комплексу обрано мову програмування Python. Цей вибір зумовлений кількома ключовими перевагами: широкий набір бібліотек, простота синтаксису та швидкість прототипування та активна спільнота.

Для розпізнавання зайнятості паркомісць на основі аналізу статичних зображень (фото) будуть використані наступні прості та доступні інструменти:

Бібліотека OpenCV (Open Computer Vision): Це основна бібліотека для завдань, пов'язаних з обробкою зображень. OpenCV буде використовуватися для завантаження зображень, їх попередньої обробки (наприклад, зміна розміру, вирізання окремих паркувальних слотів, перетворення колірних просторів).

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

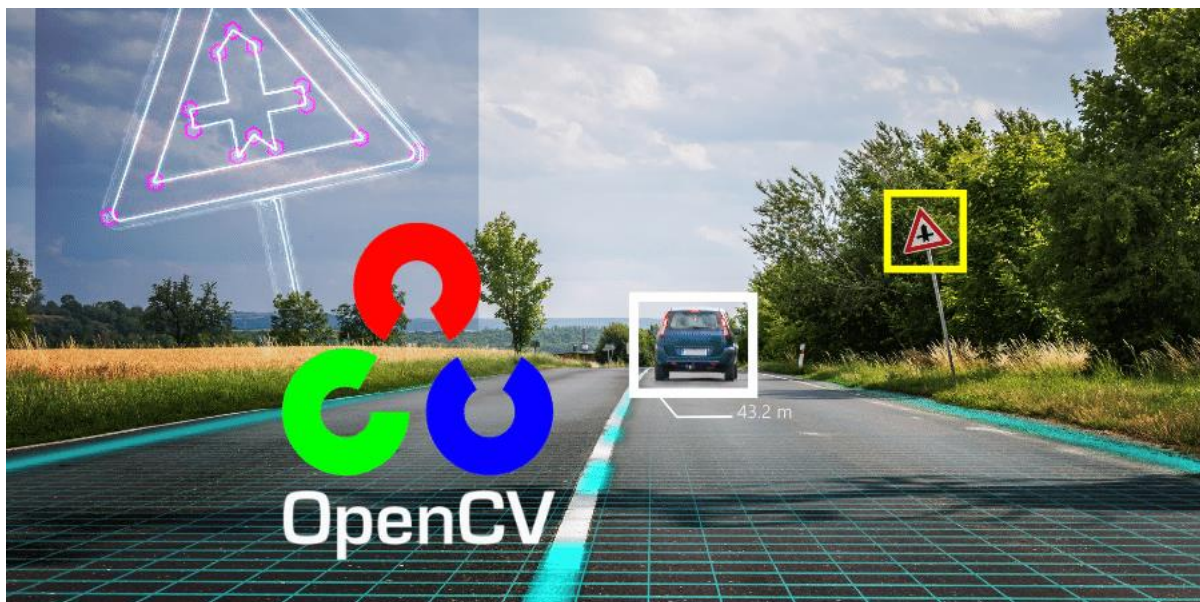


Рис. 1.6. Комп'ютерний зір з Python OpenCV

Фреймворк TensorFlow з Keras API: Для реалізації нейромережевого підходу, зокрема простої згорткової нейронної мережі (CNN) для класифікації стану паркомісця ("вільно"/"зайнято") та бібліотека NumPy.

Для MVP акцент робиться на роботі зі статичними зображеннями, що максимально спрощує розробку.

Для створення Telegram-бота, який взаємодіятиме з користувачами, обрано бібліотеку python-telegram-bot. Цей вибір обґрунтований її простотою використання, наявністю зрозумілої документації та достатньою функціональністю для реалізації базових команд (обробка запитів, надсилання повідомлень) в рамках MVP.

Враховуючи фокус на MVP, інтеграція модулів та зберігання даних буде реалізована за допомогою найпростіших підходів:

Система зберігання даних SQLite: Для зберігання інформації про статуси паркувальних місць буде використана вбудована база даних SQLite. Вона не потребує окремого серверу, легко інтегрується з Python через стандартний модуль sqlite3 і є оптимальною для невеликого обсягу даних в MVP.

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвєєв К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		24

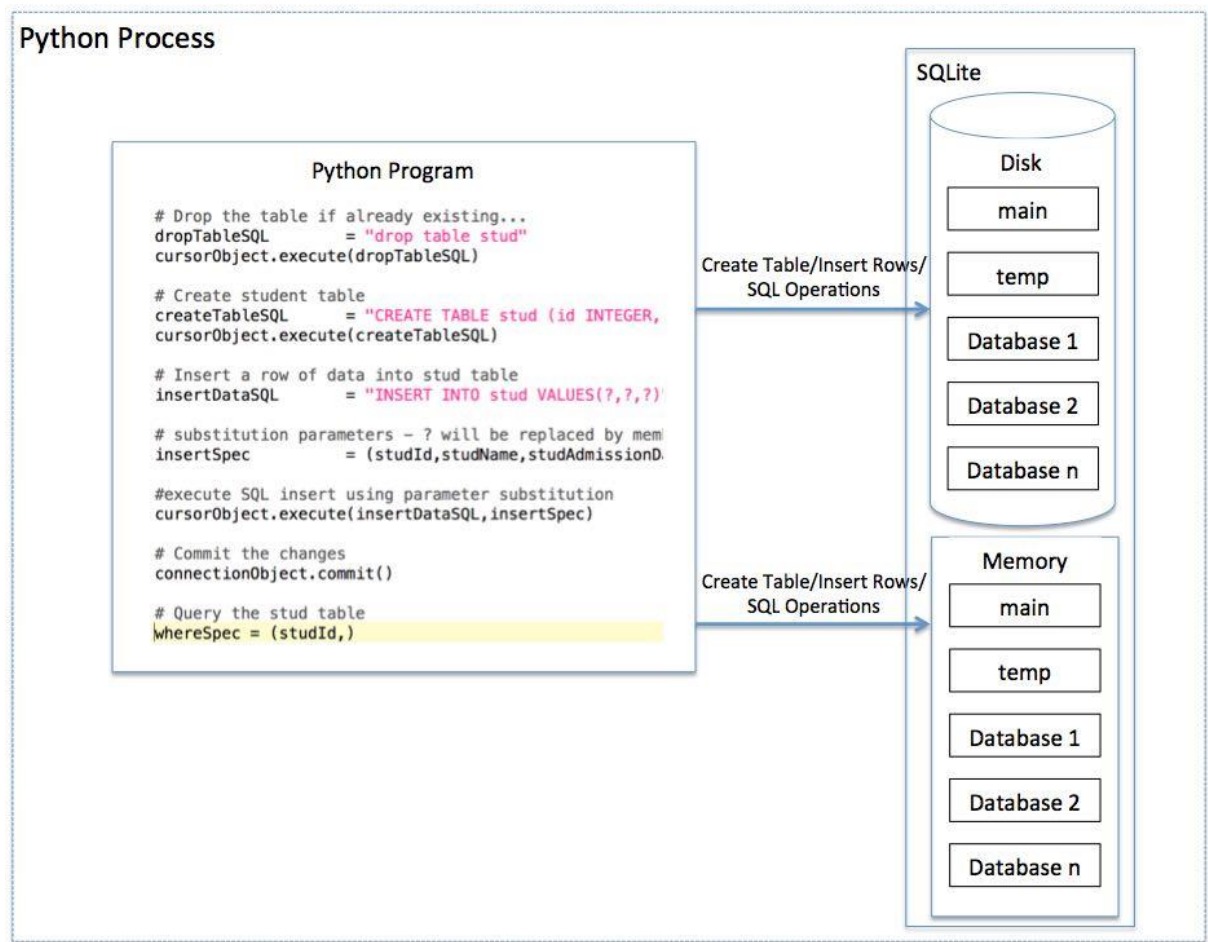


Рис. 1.7. Python and Sqlite

Обраний стек технологій та інструментів є максимально простим у використанні, що дозволяє зосередитись на реалізації ключового функціоналу MVP. Всі обрані інструменти мають низький поріг входження та забезпечують достатню гнучкість для демонстрації працездатності основної ідеї проекту з мінімальними зусиллями на налаштування та вивчення складних систем.

Висновок до першого розділу

У першому розділі було закладено теоретико-аналітичну базу для розробки програмно-апаратного комплексу моніторингу паркувальних місць.

Було проведено огляд існуючих технологій моніторингу, включаючи сенсорні, відеоаналітичні та IoT-рішення, що виявило їхні ключові переваги, недоліки та тенденцію до інтеграції інтелектуальних підходів. Поглиблений аналіз застосування нейронних мереж, зокрема згорткових архітектур (CNN), підтвердив їхню ефективність для розпізнавання зайнятості паркомісць на основі візуальних

даних та визначив важливість правильного вибору моделі, підготовки даних і метрик оцінки.

На основі цього аналізу було сформульовано функціональні та нефункціональні вимоги до мінімально життєздатного продукту (MVP), що включають автоматичне розпізнавання, взаємодію через Telegram-бот, візуалізацію статусів та концептуальне бронювання. Завершальним етапом стало обґрунтування вибору інструментальних засобів – Python, OpenCV, TensorFlow/Keras, python-telegram-bot та SQLite – з акцентом на простоті та швидкості прототипування для MVP.

Таким чином, перший розділ забезпечив необхідне підґрунтя для переходу до проєктування архітектури та подальшої реалізації системи.

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		26

РОЗДІЛ 2 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ КОМПОНЕНТІВ ПРОГРАМНО-АПАРАТНОГО КОМПЛЕКСУ

2.1 Розробка загальної архітектури програмно-апаратного комплексу

Загальна архітектура розроблюваного програмно-апаратного комплексу для моніторингу паркувальних місць проєктується з огляду на створення функціонального мінімально життєздатного продукту (MVP), а також для забезпечення можливості подальшого масштабування та впровадження розподіленої системи. Ключовим принципом є модульність, що дозволяє кожному компоненту виконувати чітко визначену роль та забезпечує гнучкість для інтеграції нових технологій, зокрема IoT-пристроїв та хмарних сервісів.

Для забезпечення ефективної роботи та можливості розширення системи, пропонується мережева архітектура, що включає локальні пристрої збору даних та центральний сервер обробки.

Локальні вузли збору даних (Edge-рівень):

На кожному паркувальному місці або для групи місць концептуально передбачається встановлення обчислювального модуля (наприклад, на базі мікрокомп'ютера типу Raspberry Pi).

Роль локального вузла це збір первинних даних про стан паркомісця за допомогою підключених сенсорів (наприклад, камер). Можлива первинна обробка даних (наприклад, детекція руху, базовий аналіз зображення) для зменшення навантаження на мережу та центральний сервер. Підготовлені дані або "сирі" дані передаються на центральний сервер.

Центральний сервер обробки та управління (Cloud-рівень):

Для основної логіки системи, зберігання даних та взаємодії з користувачами проєктується використання віртуального приватного сервера (VPS).

Роль VPS, розміщення всіх ключових серверних компонентів програмного комплексу, забезпечення їхньої взаємодії та надання доступу користувачам.

Основні функціональні компоненти системи та їхні проєктовані ролі в архітектурі:

Пристрої отримання даних (на локальних вузлах):

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		27

Забезпечення автоматичного збору візуальної або іншої сенсорної інформації про стан конкретного паркомісця.

Передають зібрані дані до брокера повідомлень на центральному сервері.

Брокер повідомлень (розгортається на VPS):

Центральний елемент для асинхронного та надійного обміну даними між численними локальними вузлами збору даних та серверними модулями обробки. Проєктується використання протоколу типу MQTT або аналогічного, що підходить для IoT-застосунків.

Приймає повідомлення від пристроїв отримання даних та маршрутизує їх до відповідних підписаних серверних компонентів.

Модуль агрегації та первинної обробки даних (розгортається на VPS):

Підписується на повідомлення від брокера, отримує дані від локальних вузлів. Здійснює агрегацію, фільтрацію та підготовку даних для передачі на аналіз модулю розпізнавання.

Отримує дані з брокера, передає підготовлені дані (наприклад, зображення слотів) модулю розпізнавання.

Модуль розпізнавання зайнятості паркомісць (ШІ-модель, розгортається на VPS):

Основний інтелектуальний компонент, що аналізує підготовлені дані (зображення) за допомогою алгоритмів машинного навчання (зокрема, згорткової нейронної мережі) для визначення статусу паркомісця ("вільно"/"зайнято") та рівня впевненості.

Отримує дані від модуля агрегації, повертає розпізнаний статус для оновлення в сховищі даних.

Сховище даних (База даних, розгортається на VPS):

Централізоване та структуроване зберігання всієї системної інформації: координати та актуальні статуси паркомісць, історія змін, дані користувачів, інформація про бронювання, (концептуально) платежі. Для MVP проєктується використання SQLite, з можливістю переходу на більш потужні СУБД (PostgreSQL, MySQL) при масштабуванні.

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		28

Обслуговує запити на читання та запис від усіх серверних модулів, що потребують доступу до даних.

Модуль візуалізації даних (Генератор карт, розгортається на VPS):

Створення актуального графічного представлення (карти) парковки, що відображає стан кожного місця, загальну статистику та легенду.

Отримує дані зі сховища даних; згенерована карта передається модулю взаємодії з користувачем.

Модуль взаємодії з користувачем (Telegram-бот, розгортається на VPS):

Надання користувачам та адміністраторам інтерфейсу для доступу до інформації про парковку, управління бронюваннями, отримання сповіщень та (концептуально) здійснення платежів.

Звертається до сховища даних, модуля візуалізації, (концептуально) платіжної системи. Отримує сповіщення від модуля агрегації/моніторингу (наприклад, через callback-механізм або підписку на брокер). Взаємодіє з користувачами через Telegram Bot API.

Платіжний модуль (концептуально, розгортається на VPS):

Обробка логіки, пов'язаної з концептуальною оплатою бронювань.

Проектована інтеграція компонентів та сценарії взаємодії:

У рамках MVP, де фізичне розгортання розподіленої системи моделюється, основні програмні модулі (розпізнавання, управління БД, логіка бота, візуалізація) розробляються для роботи в єдиному середовищі, імітуючи їхнє майбутнє розміщення на VPS.

- Локальний вузол (концептуальний Raspberry Pi з камерою) надсилає дані (зображення) до Брокера повідомлень.
- Модуль агрегації на VPS отримує ці дані з брокера.
- Модуль розпізнавання аналізує дані та оновлює статус у Сховищі даних.
- При зміні статусу або за запитом користувача, Модуль візуалізації генерує карту.

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				29
Змн.	Арк.	№ докум.	Підпис	Дата		

- Telegram-бот отримує інформацію зі Сховища даних та від Модуля візуалізації, а також може отримувати сповіщення про важливі події від Модуля агрегації (наприклад, через callback або підписку на спеціальні топіки брокера).

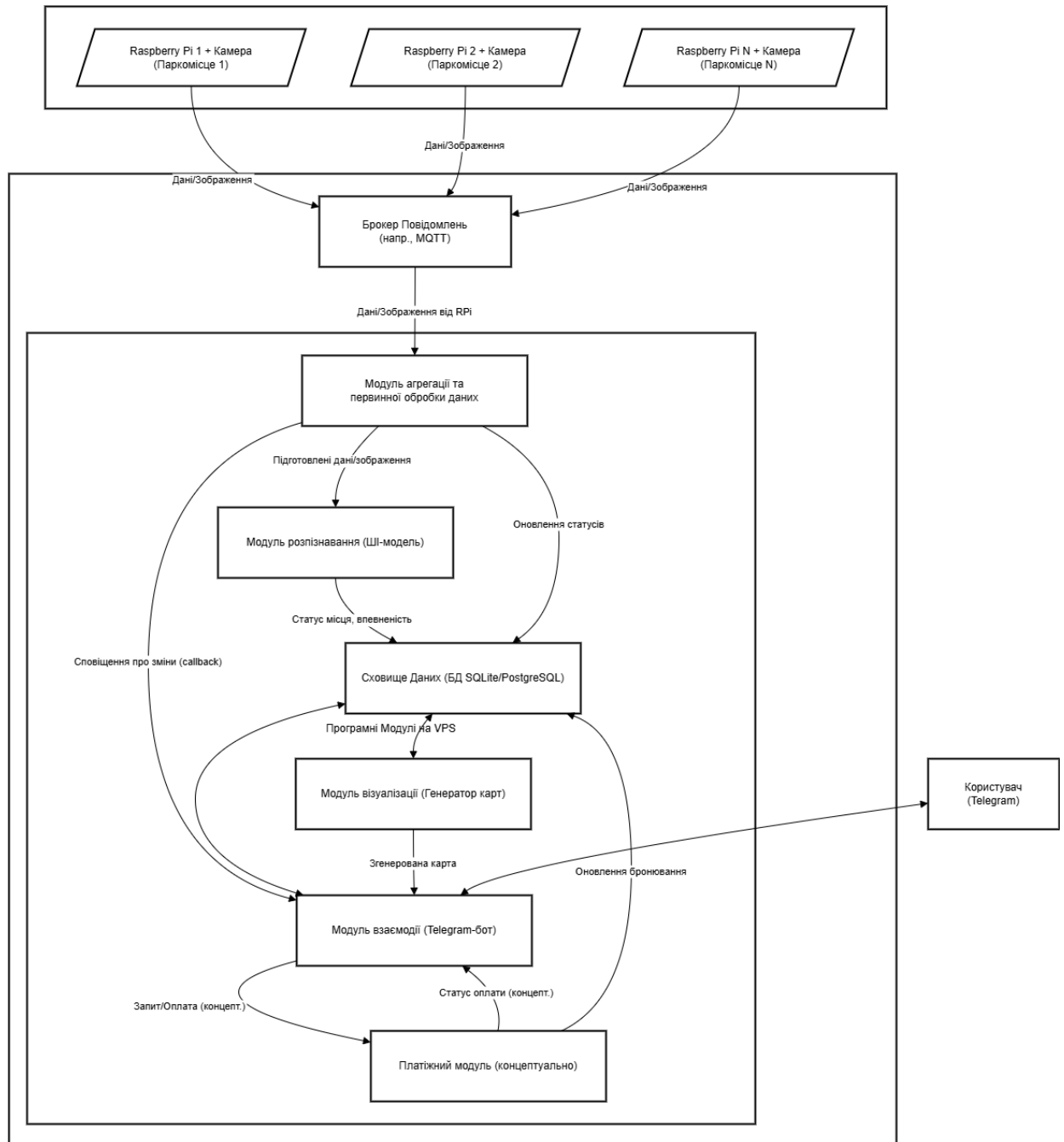


Рис. 2.1. Блок-схема, що відображає запроєктовану розподілену архітектуру

Запропонована архітектура програмно-апаратного комплексу є модульною та орієнтованою на масштабованість. Використання локальних обчислювальних вузлів для збору даних та центрального хмарного сервера (VPS) з брокером повідомлень для обробки та управління дозволяє створити гнучку систему. Чіткий розподіл ролей між компонентами спрощує проєктування, подальшу розробку та

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				30
Змн.	Арк.	№ докум.	Підпис	Дата		

підтримку. Такий підхід забезпечує ефективну реалізацію функціоналу MVP та закладає фундамент для подальшого розширення системи, включаючи інтеграцію різноманітних IoT-пристроїв та розгортання у промислових масштабах.

2.2 Проєктування модуля розпізнавання зайнятості паркомісць

Для реалізації MVP системи моніторингу паркувальних місць обрано підхід із класифікацією окремих, попередньо визначених паркувальних слотів. Такий підхід дозволяє уникнути складних задач детекції об'єктів на зображенні в реальному часі та зосередитись на бінарній класифікації: "вільно" чи "зайнято" для кожного слота. Координати паркомісць визначаються заздалегідь (наприклад, з файлу `parking_positions.pkl`, сформованого вручну або автоматично), що дає змогу вирізати відповідні області (ROI) з повного зображення парковки. Цей підхід добре вписується в загальну архітектуру системи, оскільки дозволяє швидко інтегрувати модуль розпізнавання у програмний комплекс і забезпечує достатню точність для MVP [14].

Для задачі бінарної класифікації зображень паркомісць розміром 48x48 пікселів використовується компактна згорткова нейронна мережа (CNN). Запланована архітектура містить такі шари:

- Вхідний шар: (48, 48, 3) – кольорове зображення слота.
- Згортковий шар 1: 32 фільтри, ядро 3x3, активація ReLU.
- MaxPooling 1: 2x2.
- Згортковий шар 2: 64 фільтри, ядро 3x3, активація ReLU.
- MaxPooling 2: 2x2.
- Згортковий шар 3: 64 фільтри, ядро 3x3, активація ReLU.
- MaxPooling 3: 2x2.
- Flatten: Перетворення у вектор.
- Dense: 64 нейрони, активація ReLU.
- Dropout: 0.5 (для запобігання перенавчанню).
- Вихідний шар: 1 нейрон, активація sigmoid (ймовірність зайнятості).

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		31

Така архітектура є оптимальною для задачі, оскільки поєднує простоту, швидкість навчання та достатню глибину для розпізнавання характерних ознак зайнятості місця [14].

Для навчання моделі планується використання датасету PKLot, а саме – його частини із сегментованими слотами (patches) [15]. Кожен патч відповідає окремому паркомісцю та має мітку "вільно"/"зайнято".

- Зміна розміру всіх зображень до 48x48 пікселів.
- Нормалізація значень пікселів у діапазон [0,.
- Аугментація даних для підвищення стійкості моделі: випадкові повороти, горизонтальні/вертикальні віддзеркалення, зсуви, масштабування, зміна яскравості та контрасту.

Дані діляться на тренувальну, валідаційну та тестову вибірки у співвідношенні 70/15/15 або 80/10/10 для забезпечення коректної оцінки якості моделі [16].

Параметри навчання:

- Оптимізатор: Adam (добре підходить для задач комп'ютерного зору).
- Функція втрат: Binary Crossentropy (для бінарної класифікації).
- Метрики: Accuracy (точність), а також Precision, Recall для детальної оцінки.
- Callbacks:
 - ModelCheckpoint (збереження найкращої моделі),
 - EarlyStopping (зупинка при відсутності покращення на валідації),
 - ReduceLROnPlateau (зменшення швидкості навчання при плато),
 - TensorBoard (моніторинг навчання)

Модель тренується до досягнення стабільного значення точності на валідаційній вибірці або до спрацювання EarlyStopping. Критерієм успішності є досягнення точності не нижче 95% на тестових даних, що відповідає результатам у літературі [17].

Проектування механізму використання моделі для розпізнавання (inference) запланований процес:

1. Отримання повного зображення парковки з камери (або тестового фото).

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		32

2. Вирізання окремих слотів за координатами з parking_positions.pkl.
3. Попередня обробка кожного слота (зміна розміру, нормалізація).
4. Подача зображення на вхід завантаженої моделі.
5. Отримання ймовірності зайнятості (output sigmoid):
Якщо ймовірність > 0.5 – місце вважається "зайнятим", інакше – "вільним".
6. Передача статусу паркомісця у базу даних для подальшої обробки іншими модулями системи.

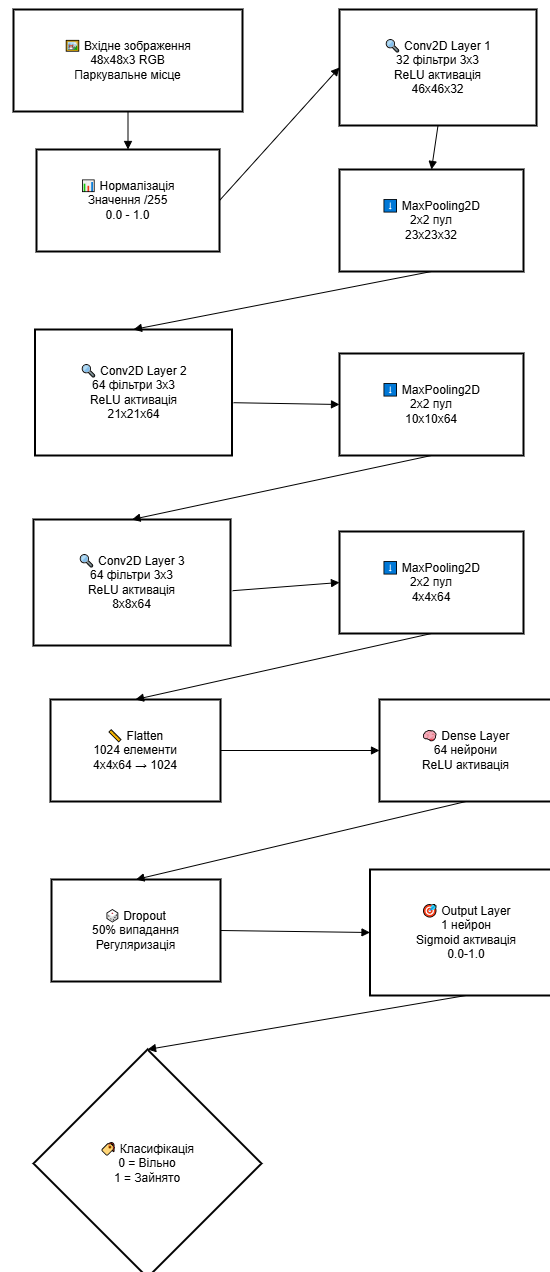


Рис. 2.2. Візуальна схему запланованої архітектури CNN

Успішним результатом проектування модуля розпізнавання вважається якщо досягнення точності класифікації не нижче 95% на тестових даних PKLot або

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				33
Змн.	Арк.	№ докум.	Підпис	Дата		

аналогічного датасету. Швидкість обробки одного зображення – не більше 0.2 секунди на сучасному ПК (достатньо для MVP). Стабільна робота моделі на зображеннях із різними погодними умовами та ракурсами. Можливість інтеграції модуля у загальну систему моніторингу паркувальних місць.

Такий підхід дозволяє створити простий, ефективний і масштабований модуль розпізнавання, який може бути легко адаптований для різних паркінгів і сценаріїв використання.

2.3 Проєктування модуля взаємодії з користувачем через Telegram-бот

Telegram-бот у нашому проєкті виступає основним інтерфейсом для взаємодії користувачів і адміністраторів із системою моніторингу паркувальних місць. Його використання дозволяє забезпечити швидкий доступ до актуальної інформації про стан парковки, бронювання місць та отримання сповіщень у зручному та звичному для більшості користувачів форматі.

Telegram-бот обраний як інтерфейс з таких причин:

Telegram є одним із наймасовіших месенджерів, що підтримується на всіх основних платформах (Android, iOS, Windows, macOS, Web), тому не потребує встановлення додаткових програм.

Бот може оперативно надсилати push-повідомлення про зміни статусу паркомісць, завершення бронювання, появу вільних місць тощо.

Інтерфейс бота інтуїтивно зрозумілий, а всі дії виконуються через прості команди або кнопки.

Легко реалізується розділення прав між звичайними користувачами та адміністраторами.

Telegram Bot API дозволяє швидко створювати інтерактивних ботів, а для MVP достатньо прямої роботи через HTTP-запити, що дає повний контроль над логікою.

Telegram-бот виступає центральною ланкою, яка координує обмін інформацією між користувачем і програмними модулями системи:

1. База даних (parking_database.py), зберігає статуси місць, інформацію про бронювання, статистику.

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		34

2. Модуль моніторингу (parking_monitor.py), інформує бота про зміни статусу місць через callback-механізм.

3. Платіжна система (payment_system.py) (концептуально), ініціює та перевіряє платежі за бронювання.

4. Генератор карт (parking_map_generator.py), створює актуальну схему парковки для надсилання користувачам.

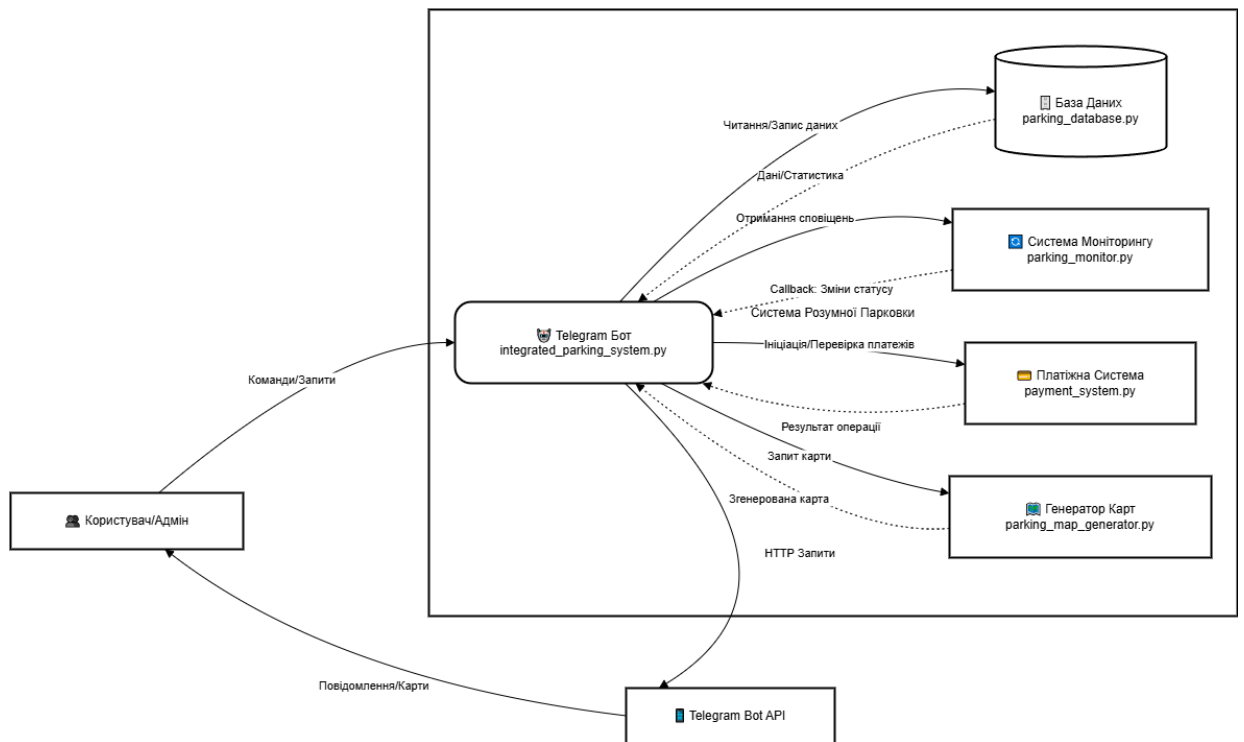


Рис. 2.3. Місце бота в архітектурі та основні потоки даних

Проектування функціоналу та команд Telegram-бота:

Користувацькі команди:

- /start — головне меню, привітання.
- /map — отримання карти парковки.
- /stats — детальна статистика для користувача.
- /my_bookings — перегляд активних бронювань.
- /subscribe — підписка на автоматичні сповіщення.
- /pay <номер_місця> — ініціація оплати/бронювання.
- /cancel <ID_бронювання> — скасування власного бронювання.
- (інші: швидке бронювання, допомога)

Адміністративні команди:

- /admin_bookings — управління всіма бронюваннями.
- /adminstats — розширена статистика.
- /admin_expired — перегляд прострочених бронювань.
- /admin_cancel <ID_бронювання> — адміністраторське скасування.
- (інші: розсилка, перегляд логів)

Для адміністративних команд бот перевіряє chat_id користувача у списку адміністраторів.

Автоматичні функції та сповіщення

При зміні статусу місця (наприклад, звільнення чи бронювання) модуль моніторингу викликає callback, а бот надсилає відповідне повідомлення підписаним користувачам.

Автоматичне оновлення статусу місць та інформування користувачів про завершення бронювання.

Для реалізації Telegram-бота планується використовувати прямі HTTP-запити до Telegram Bot API через бібліотеку requests, без застосування додаткових фреймворків. Це забезпечує повний контроль над логікою, мінімум залежностей і простоту інтеграції з іншими модулями системи.

Запроєктований Telegram-бот є зручним, гнучким і масштабованим інтерфейсом для користувачів та адміністраторів системи моніторингу паркувальних місць. Його архітектура дозволяє легко розширювати функціонал, інтегрувати нові сервіси та забезпечує ефективну взаємодію з усіма ключовими модулями комплексу.

2.4 Проєктування системи візуалізації та інтеграції даних про статуси паркомісць

Ефективна система візуалізації та інтеграції даних є ключовим елементом сучасних програмно-апаратних комплексів для моніторингу паркувальних місць. Вона забезпечує не лише зручне сприйняття інформації користувачами, а й надійну взаємодію між усіма модулями системи, що критично важливо для коректної роботи MVP та подальшого розвитку рішення.

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		36

Візуальне представлення стану парковки дозволяє користувачам і адміністраторам швидко оцінити ситуацію, знайти вільне місце або проконтролювати ефективність використання паркінгу. Для MVP обрано підхід генерації статичної карти у вигляді растрового зображення (PNG), оскільки це простий, наочний і технологічно доступний спосіб відображення інформації для широкого кола користувачів.

Карта генерується на основі даних із центральної бази даних SQLite (модуль `parking_database.py`). Для кожного паркомісця зберігається унікальний ідентифікатор (`space_id`), координати (отримані з файлу `parking_positions.pkl` і збережені в БД) та актуальний статус (`"free"`, `"occupied"`, `"reserved"`). Функція генерації карти (`generate_simple_map` у модулі `parking_map_generator.py`) виконує такі кроки:

- Зчитує всі паркомісця з БД.
- Визначає розміри карти та розташування слотів залежно від кількості місць і їх координат.
- Створює порожнє зображення-полотно потрібного розміру за допомогою бібліотеки OpenCV або Pillow.

Для інтуїтивного сприйняття на карті використовується кольорове кодування:

- Вільне місце (`free`)
- Зайняте місце (`occupied`)
- Заброньоване місце (`reserved`)

Кожне місце на карті відображається у вигляді прямокутника (або іншої фігури), всередині або поруч із яким розміщується його номер (`space_id`). Ця логіка реалізується у відповідній функції модуля генерації карти, де для кожного місця малюється фігура з потрібним кольором і наноситься текстова мітка.

Для підвищення інформативності на карту додається:

- Загальна статистика: кількість вільних, зайнятих, заброньованих місць.
- Поточний час генерації карти.
- Легенда з розшифровкою кольорів.

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвєєв К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		37

Весь текст наноситься на зображення з урахуванням підтримки кирилиці (наприклад, через Pillow із відповідним шрифтом).

Згенерована карта зберігається як файл (наприклад, current_parking_map.png). Telegram-бот (integrated_parking_system.py) при запиті користувача викликає функцію генерації карти, отримує шлях до файлу та надсилає його користувачу разом із короткою статистикою та підказками щодо подальших дій.

База даних SQLite (керована модулем parking_database.py) є основним джерелом істини для всієї системи. Усі модулі (моніторинг, візуалізація, Telegram-бот, платіжна система) взаємодіють із цією БД для отримання та оновлення актуальних статусів паркомісць, бронювань, історії змін тощо. Такий підхід гарантує цілісність і узгодженість даних у системі.

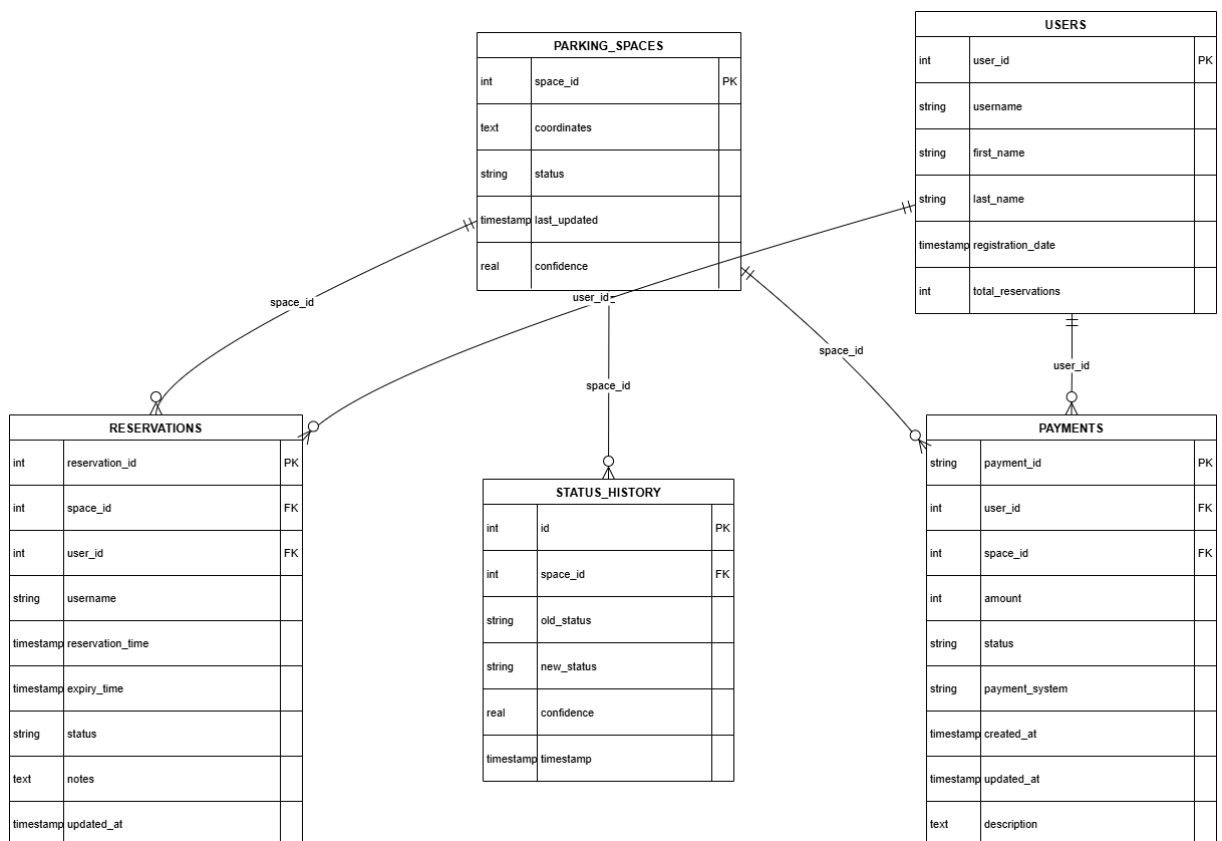


Рис. 2.4. ER-діаграма структури таблиць БД

Після аналізу кожного зображення ІІІ-моделлю модуль моніторингу викликає метод оновлення статусу місця (наприклад, update_space_status(space_id, new_status, confidence)), фіксуючи зміни у таблиці паркомісць та історії змін.

Модуль візуалізації отримує список усіх місць із їхніми статусами та координатами через метод `get_all_spaces()`, що дозволяє формувати актуальну карту.

Бот використовує методи БД для отримання інформації про місця, статистику, бронювання (`get_all_spaces()`, `get_statistics()`, `get_user_reservations()`), а також для запису нових бронювань чи скасування (`make_reservation()`, `cancel_reservation()`).

Для забезпечення оперативного інформування користувачів про зміни у системі реалізовано `callback`-механізм.

Всі методи, що виконують кілька операцій запису (наприклад, оновлення статусу місця та запис в історію), реалізовані з використанням транзакцій SQLite (`conn.commit()`, `conn.rollback()`). Це гарантує, що у разі виникнення помилок дані залишаються цілісними, а система — надійною.

У рамках MVP актуалізація даних ініціюється модулем моніторингу (`parking_monitor.py`), який у циклі або за розкладом (наприклад, через `threading.Timer`) обробляє нові зображення з папки камери. Після кожного аналізу, оновлюються статуси місць у БД, генерується нова карта, запускається система сповіщень через `callback`, що дозволяє Telegram-боту оперативно інформувати підписників.

Запроєктована система візуалізації та інтеграції даних поєднує простоту реалізації з гнучкістю та масштабованістю. Вона забезпечує зручне та інформативне відображення стану парковки, надійний обмін даними між модулями та оперативне інформування користувачів. Такий підхід дозволяє ефективно підтримати всі функціональні та нефункціональні вимоги MVP і закладає підґрунтя для подальшого розвитку системи.

Висновок до другого розділу

У другому розділі було комплексно спроектовано програмно-апаратний комплекс для моніторингу паркувальних місць. Розроблено загальну модульну архітектуру системи, що включає модуль збору даних, модуль розпізнавання на

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		39

базі CNN, Telegram-бот для взаємодії з користувачем, модуль візуалізації у вигляді статичної карти та сховище даних SQLite.

Детально спроектовано модуль розпізнавання, визначено архітектуру CNN для класифікації стану паркомісць, сплановано процес підготовки даних з датасету PKLot, а також процес навчання та використання моделі. Також розроблено функціонал та команди Telegram-бота, включаючи механізми взаємодії з іншими компонентами системи через базу даних та callback-механізм. На завершення, спроектовано систему візуалізації даних на статичній карті та принципи інтеграції даних для забезпечення їх актуальності та узгодженості.

Представлені проєктні рішення створюють міцну основу для подальшої реалізації та тестування MVP системи.

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		40

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОТОТИПУ СИСТЕМИ МОНІТОРИНГУ

3.1 Реалізація модуля розпізнавання зайнятості паркомісць та його функціональне тестування

Модуль розпізнавання зайнятості паркомісць є ключовим компонентом системи, що виконує автоматичну класифікацію окремих сегментованих зображень паркувальних слотів на категорії "вільно" та "зайнято". Його розробка базується на проєктних рішеннях, описаних у підрозділі 2.2, з акцентом на використанні згорткової нейронної мережі (CNN) та сучасних практик підготовки і тестування моделей комп'ютерного зору.

Розробка модуля здійснювалася мовою Python у середовищі PyCharm. Для реалізації були використані такі бібліотеки:

- TensorFlow (Keras API) — побудова та навчання нейронної мережі.
- OpenCV — обробка та підготовка зображень.
- NumPy — робота з багатовимірними масивами.
- Matplotlib, Seaborn — візуалізація даних, графіки навчання та аналізу результатів.
- Scikit-learn — метрики класифікації, побудова матриці помилок, звіт по класифікації.
- Pandas — зручна робота з анотаціями та метаданими датасету.

Для навчання та тестування моделі використовувався датасет PKLot (папка PKLotSegmented), що містить сегментовані зображення паркувальних місць із мітками "occupied"/"empty". Завантаження та підготовка даних реалізовано у скрипті load_pklot.py.

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		41

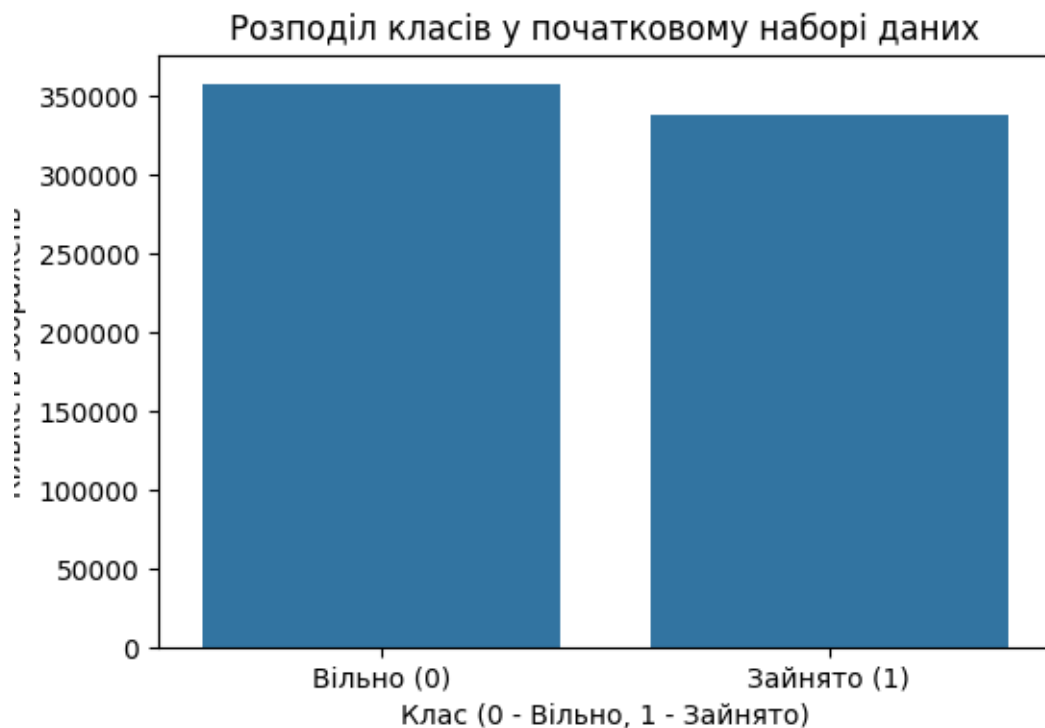


Рис. 3.1. Розподіл класів "вільно" та "зайнято" у початковому наборі даних
PKLotSegmented

Основні кроки підготовки:

- **Зміна розміру:** усі зображення приводяться до розміру 48x48 пікселів для уніфікації вхідних даних.
- **Нормалізація:** пікселі масштабуються в діапазон [0, для стабільної роботи моделі.
- **Розділення вибірки:** дані діляться на тренувальну, валідаційну та тестову вибірки у співвідношенні 70/15/15 за допомогою `train_test_split` зі збереженням балансу класів.
- **Аналіз класів:** для контролю балансу класів будуються діаграми розподілу усього набору та кожної вибірки (Matplotlib/Seaborn).

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				42
Змн.	Арк.	№ докум.	Підпис	Дата		

Приклади аугментації для одного зображення



Рис. 3.2. Приклади аугментації зображень з тренувальної вибірки

Аугментація даних здійснюється за допомогою ImageDataGenerator: випадкові повороти, зсуви, масштабування, віддзеркалення. Це підвищує стійкість моделі до різних ракурсів і умов зйомки. Приклади аугментованих зображень виводяться для візуального контролю.

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		43

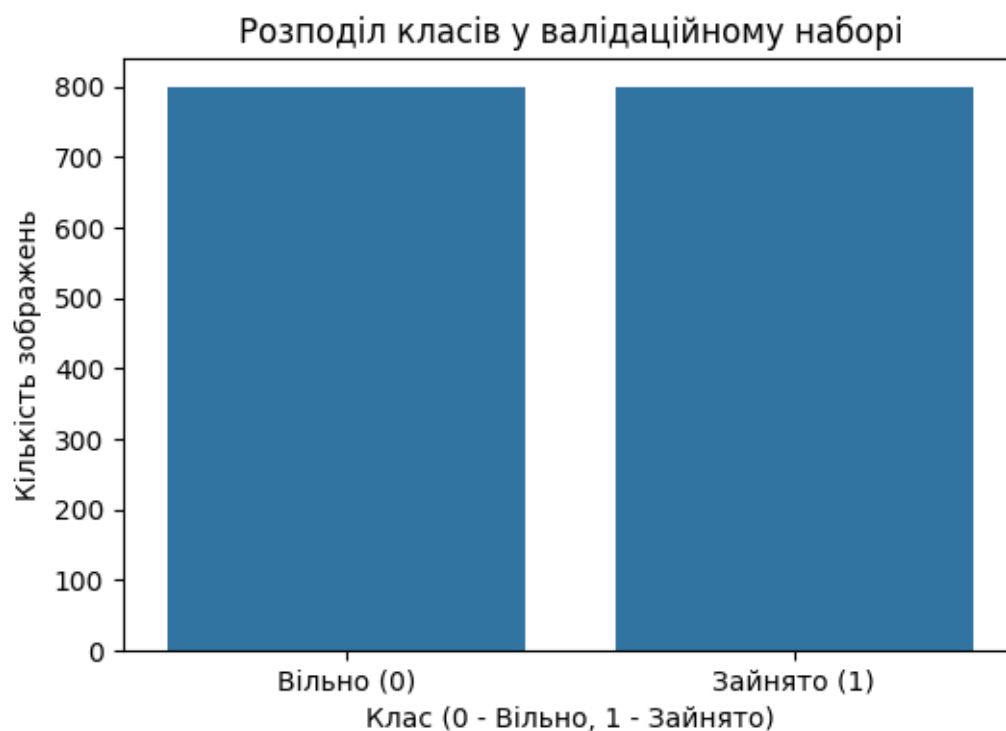


Рис. 3.3. Розподіл класів у валідаційній вибірці

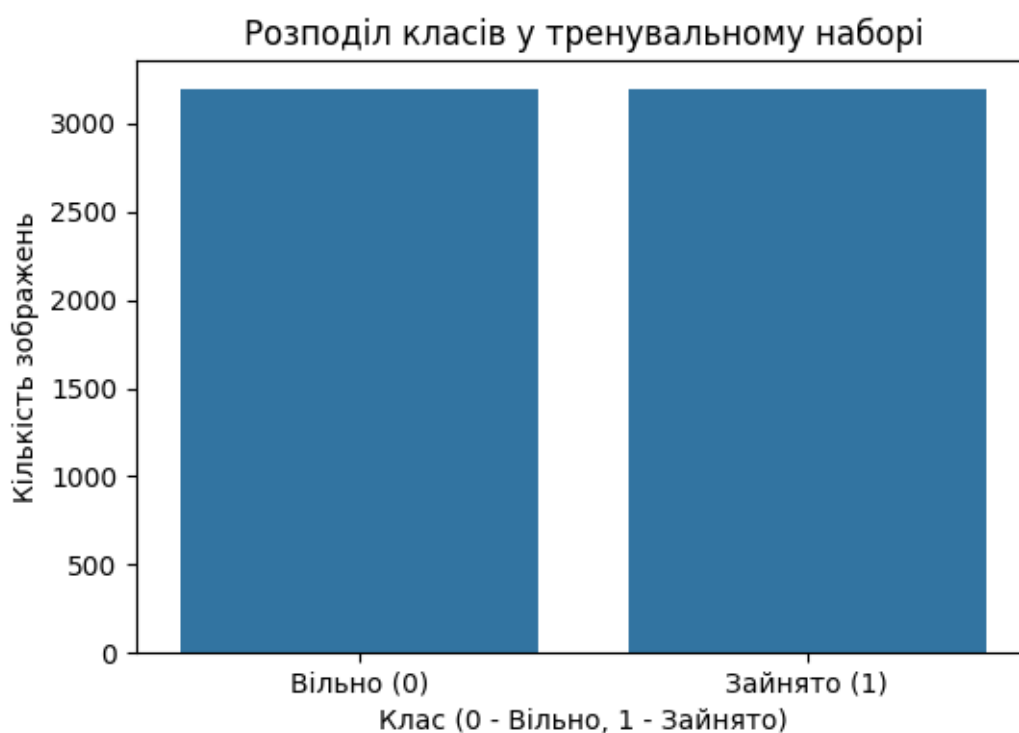


Рис. 3.4. Розподіл класів у тренувальній вибірці

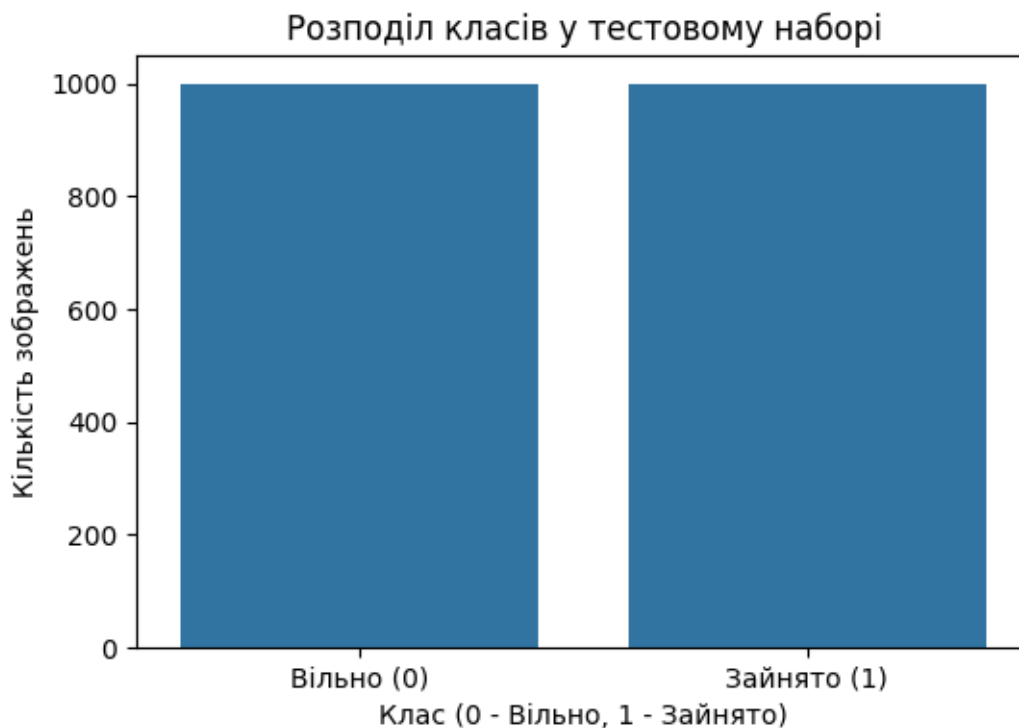


Рис. 3.5. Розподіл класів у тестовій вибірці

Архітектура згорткової нейронної мережі для класифікації паркувального слота реалізована у скрипті `load_pklot.py` та містить такі шари:

- Вхідний шар: (48, 48, 3)
- 3 згорткових шари (Conv2D) з кількістю фільтрів 32, 64, 64, ядро 3x3, активація ReLU
- Шари MaxPooling2D (2x2) після кожного згорткового шару
- Flatten
- Dense (64 нейрони, ReLU)
- Dropout (0.5)
- Вихідний Dense (1, sigmoid)

```
model = Sequential([
    Input(shape=(img_height, img_width, 3)),
    Conv2D(32, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(64, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])
```

Модель компілюється з оптимізатором Adam, функцією втрат Binary Crossentropy та метрикою Ассурасу. Навчання проводиться протягом 100-150 епох із розміром батча 32-64, із використанням callback-функцій:

```
Epoch 2/150
200/200 — 8s 41ms/step - accuracy: 0.9691 - loss: 0.0878 - val_accuracy: 0.9831 - val_loss: 0.0580 - learning_rate: 0.0010
Epoch 3/150
200/200 — 8s 40ms/step - accuracy: 0.9783 - loss: 0.0734 - val_accuracy: 0.9869 - val_loss: 0.0489 - learning_rate: 0.0010
Epoch 4/150
200/200 — 8s 41ms/step - accuracy: 0.9815 - loss: 0.0566 - val_accuracy: 0.9762 - val_loss: 0.0813 - learning_rate: 0.0010
Epoch 5/150
200/200 — 8s 40ms/step - accuracy: 0.9730 - loss: 0.0753 - val_accuracy: 0.9912 - val_loss: 0.0394 - learning_rate: 0.0010
Epoch 6/150
200/200 — 8s 41ms/step - accuracy: 0.9875 - loss: 0.0392 - val_accuracy: 0.9906 - val_loss: 0.0458 - learning_rate: 0.0010
Epoch 7/150
200/200 — 8s 40ms/step - accuracy: 0.9858 - loss: 0.0467 - val_accuracy: 0.9869 - val_loss: 0.0430 - learning_rate: 0.0010
Epoch 8/150
200/200 — 8s 41ms/step - accuracy: 0.9905 - loss: 0.0391 - val_accuracy: 0.9875 - val_loss: 0.0434 - learning_rate: 0.0010
Epoch 9/150
200/200 — 8s 40ms/step - accuracy: 0.9879 - loss: 0.0408 - val_accuracy: 0.9931 - val_loss: 0.0335 - learning_rate: 0.0010
Epoch 10/150
200/200 — 8s 40ms/step - accuracy: 0.9893 - loss: 0.0335 - val_accuracy: 0.9875 - val_loss: 0.0479 - learning_rate: 0.0010
Epoch 11/150
200/200 — 8s 40ms/step - accuracy: 0.9876 - loss: 0.0417 - val_accuracy: 0.9937 - val_loss: 0.0261 - learning_rate: 0.0010
Epoch 12/150
200/200 — 8s 41ms/step - accuracy: 0.9911 - loss: 0.0286 - val_accuracy: 0.9844 - val_loss: 0.0529 - learning_rate: 0.0010
Epoch 13/150
200/200 — 8s 40ms/step - accuracy: 0.9894 - loss: 0.0371 - val_accuracy: 0.9937 - val_loss: 0.0247 - learning_rate: 0.0010
Epoch 14/150
200/200 — 8s 40ms/step - accuracy: 0.9865 - loss: 0.0369 - val_accuracy: 0.9812 - val_loss: 0.0676 - learning_rate: 0.0010
Epoch 15/150
200/200 — 8s 40ms/step - accuracy: 0.9920 - loss: 0.0247 - val_accuracy: 0.9894 - val_loss: 0.0508 - learning_rate: 0.0010
```

Рис. 3.6. Процес навчання моделі

- ModelCheckpoint: збереження найкращої моделі за val_accuracy.
- EarlyStopping: зупинка при відсутності покращення val_loss.
- ReduceLROnPlateau: зменшення швидкості навчання при плато.
- TensorBoard: моніторинг процесу навчання.

Графіки точності та втрат на тренувальній і валідаційній вибірках демонструють стабільну збіжність моделі без ознак перенавчання (overfitting).

Оцінка моделі здійснюється за допомогою model.evaluate(test_generator), що повертає фінальні значення втрат та точності на тестовій вибірці. Для глибшого аналізу:

Обчислюється матриця помилок (confusion matrix), де:

- TP — правильно визначені "зайняті"
- TN — правильно визначені "вільні"
- FP — хибно визначені як "зайняті"
- FN — хибно визначені як "вільні"

Генерується classification report (precision, recall, F1-score) для обох класів.

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		46

Візуалізуються приклади передбачень моделі на тестових даних: як правильно, так і неправильно класифіковані зображення з коротким аналізом (наприклад, складні погодні умови або часткове перекриття авто).

Найкраща версія моделі зберігається у файл (наприклад, `parking_slot_classifier.keras`) за допомогою `ModelCheckpoint`. Це дозволяє використовувати модель для подальшого inference та інтеграції у систему моніторингу.

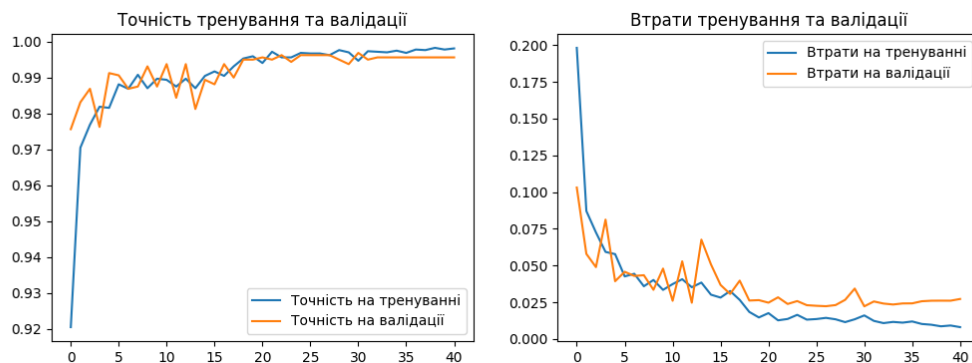


Рис. 3.7. Динаміка точності та функції втрат моделі на тренувальній та валідаційній вибірках протягом епох навчання

Для тестування функціоналу створено окрему функцію, яка завантажує збережену модель, приймає шлях до зображення паркувального слота, виконує попередню обробку (зміна розміру, нормалізація), подає зображення на вхід моделі, отримує ймовірність зайнятості та виводить результат у зручному форматі (наприклад, "Місце #15: Зайнято (98%)").

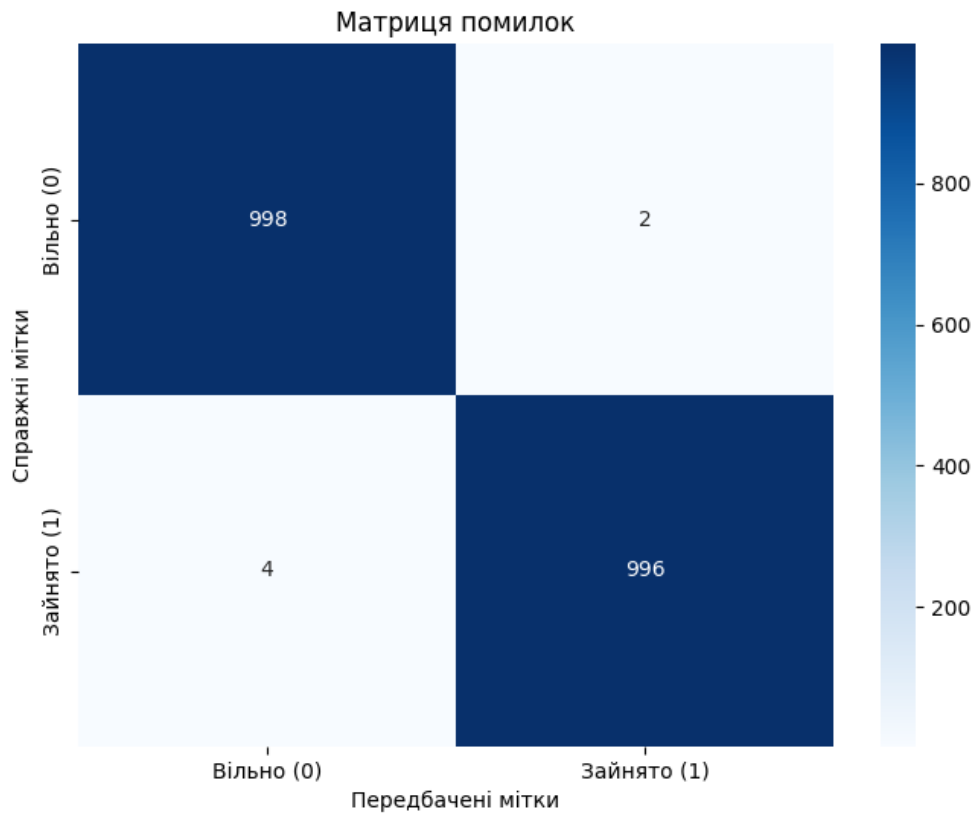


Рис. 3.8. Матриця помилок для оцінки моделі на тестовій вибірці

На кількох тестових зображеннях (які не використовувалися під час навчання) було проведено демонстрацію роботи модуля. Модель показала високу точність та швидкість класифікації (менше 0.1 секунди на зображення), що відповідає очікуванням, сформульованим у підрозділі 2.2. Помилки класифікації виявлялися переважно у випадках складних погодних умов або сильного перекриття авто, що є типовим для реальних задач комп'ютерного зору.

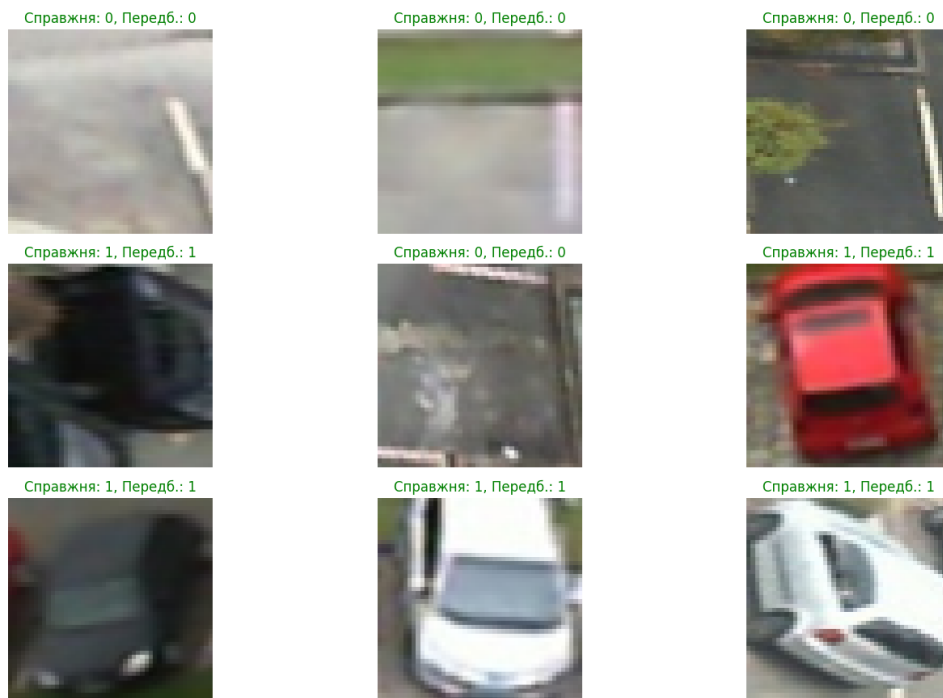


Рис. 3.9. Приклади класифікації паркомісць моделлю на тестових даних
(вказано справжню та передбачену мітку)

Реалізований модуль розпізнавання зайнятості паркомісць на основі згорткової нейронної мережі повністю відповідає поставленим функціональним вимогам MVP. Проведене тестування підтвердило його працездатність, високу точність та ефективність інтеграції у загальну систему моніторингу. Модуль може бути використаний як окремо, так і в складі комплексного рішення для автоматизованого моніторингу паркувальних місць.

3.2 Реалізація модуля Telegram-бота та його функціональне тестування

Telegram-бот у розробленій системі виступає основним інтерфейсом взаємодії користувачів та адміністраторів з комплексом моніторингу паркувальних місць. Його реалізація базується на проєктних рішеннях, визначених у підрозділі 2.3, та забезпечує повноцінний доступ до всіх функцій системи через зручний та інтуїтивний інтерфейс. Бот інтегрується з усіма ключовими компонентами: модулем розпізнавання, базою даних, системою візуалізації та платіжним модулем, забезпечуючи цілісну роботу всього програмно-апаратного комплексу.

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		49

Реалізація Telegram-бота здійснювалася мовою Python у середовищі PyCharm з використанням прямого API Telegram Bot без додаткових фреймворків. Основні бібліотеки, що використовувалися:

- requests — для HTTP-запитів до Telegram Bot API
- threading — для паралельної роботи бота та системи моніторингу
- json — для обробки JSON-даних від API та внутрішніх структур
- time, datetime — для роботи з часовими мітками та інтервалами
- os — для роботи з файловою системою та шляхами до зображень

Реалізація основної логіки бота

Основою роботи бота є метод `get_bot_updates()`, який використовує long polling для отримання оновлень від Telegram Bot API. Реалізація включає:

```
def get_bot_updates(self):
    """Отримує оновлення від Telegram"""
    url = f"{BASE_URL}/getUpdates"
    params = {'offset': self.last_update_id + 1, 'timeout': 30}

    try:
        response = requests.get(url, params=params, timeout=35)
        data = response.json()
        if data['ok']:
            return data['result']
        else:
            print(f"❌ Помилка getUpdates: {data.get('description', 'Unknown error')}")
            return []
    except Exception as e:
        print(f"❌ Помилка мережі getUpdates: {e}")
        return []
```

Головний цикл бота (`run_bot()`) постійно опитує API та обробляє нові повідомлення, забезпечуючи безперервну роботу системи.

Центральна функція обробки повідомлень функція `handle_message()` є ядром обробки всіх вхідних повідомлень. Вона парсить текст, визначає тип команди (основні, адміністративні, платіжні) та викликає відповідні обробники. Логіка включає підтримку понад 20 різних команд та типів повідомлень, включаючи швидке бронювання через надсилання номера місця.

Реалізовано надійну систему відправки повідомлень з fallback-механізмом:

```
def send_message(self, chat_id, text):
    if self.send_message_with_parse_mode(chat_id, text, 'HTML'):
        return True
    print("Fallback: надсилаємо без розмітки")
    return self.send_message_with_parse_mode(chat_id, text, None)
```

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				50
Змн.	Арк.	№ докум.	Підпис	Дата		

Реалізація користувацьких команд

Команда `/start` реалізує привітання користувача з детальним описом можливостей системи. Включає HTML-розмітку для красивого відображення та автоматично розрізняє звичайних користувачів та адміністраторів.

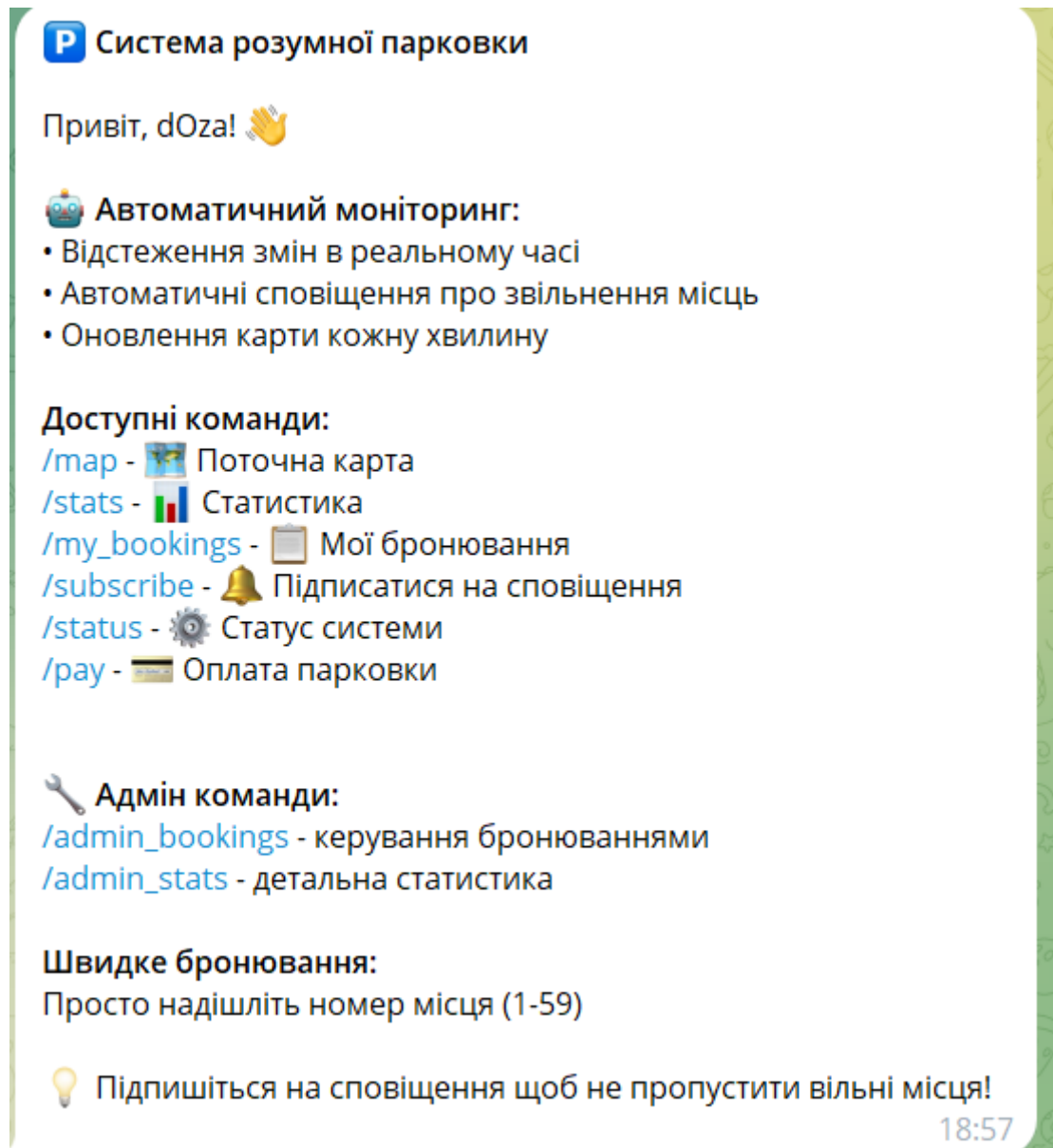


Рис. 3.10. Оформлення головного меню

Команда `/map` найбільш важлива функція системи, що генерує та надсилає актуальну карту парковки.

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				51
Змн.	Арк.	№ докум.	Підпис	Дата		

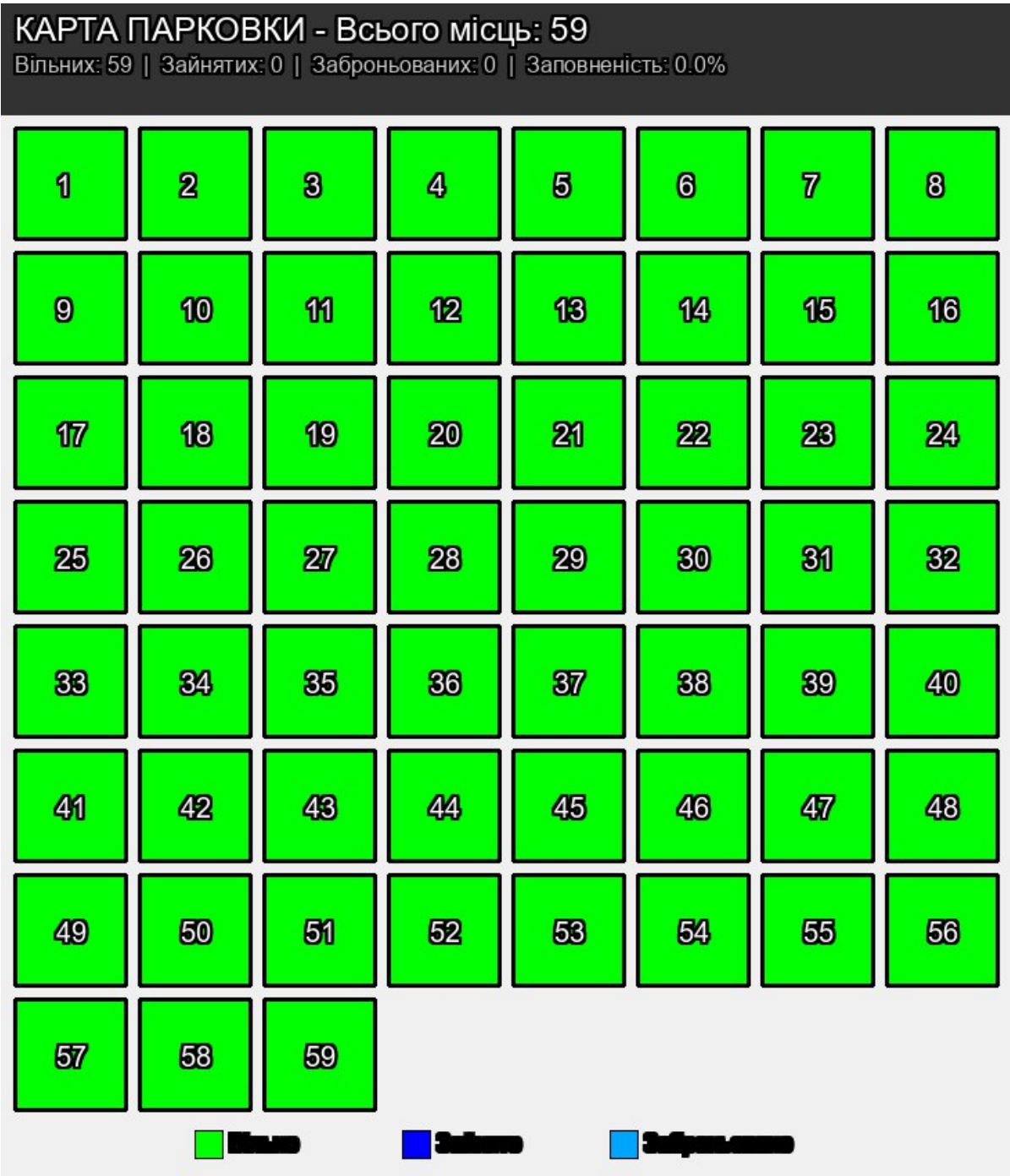


Рис. 3.11. Оформлення головного меню

Команда /stats надає детальну статистику з інформацією про поточний стан парковки, включаючи список вільних місць та статус моніторингу.

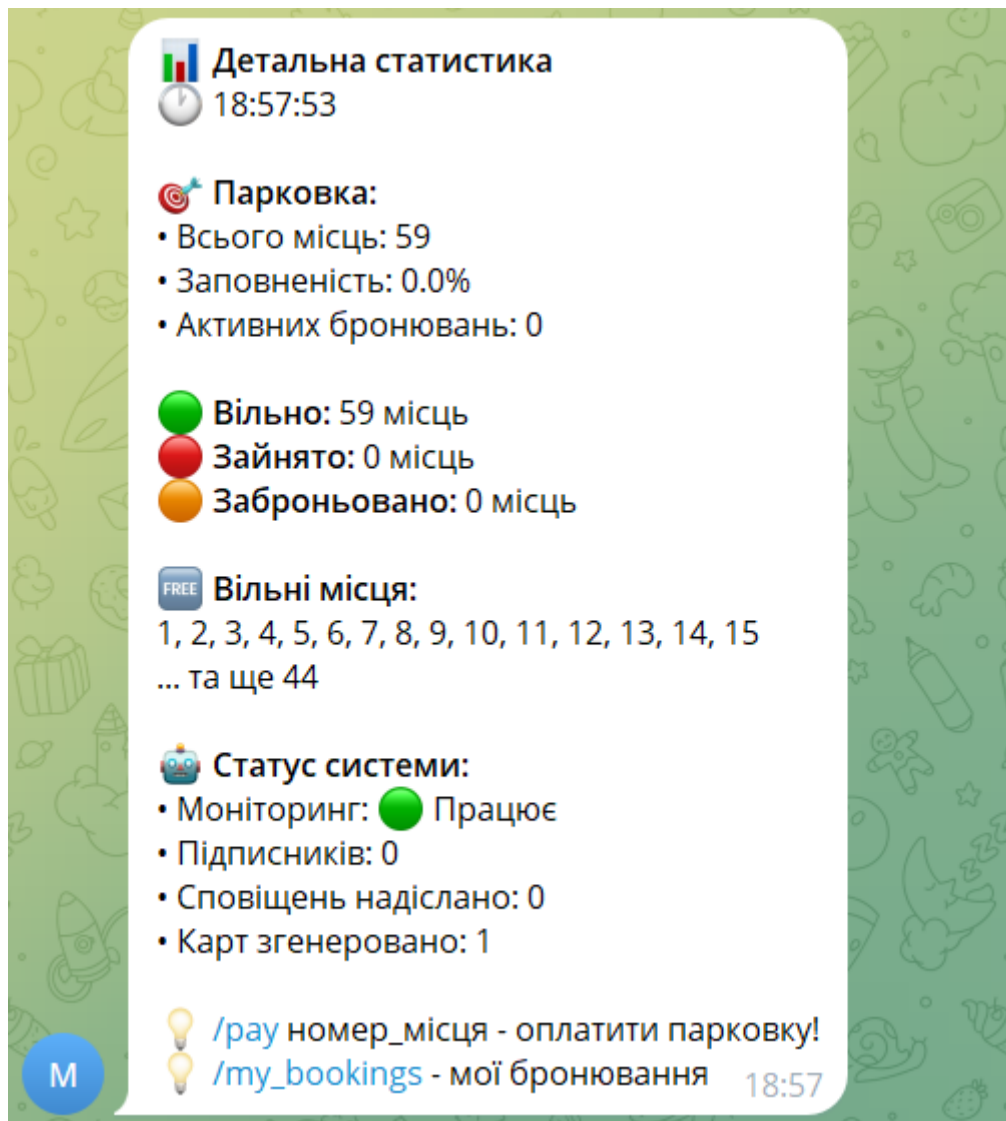


Рис. 3.12. Отримання статистики

Система підписки (`/subscribe`) реалізує механізм підписки на автоматичні сповіщення. Користувачі додаються до множини `self.subscribers` та отримують push-сповіщення про всі зміни в системі.

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				53
Змн.	Арк.	№ докум.	Підпис	Дата		

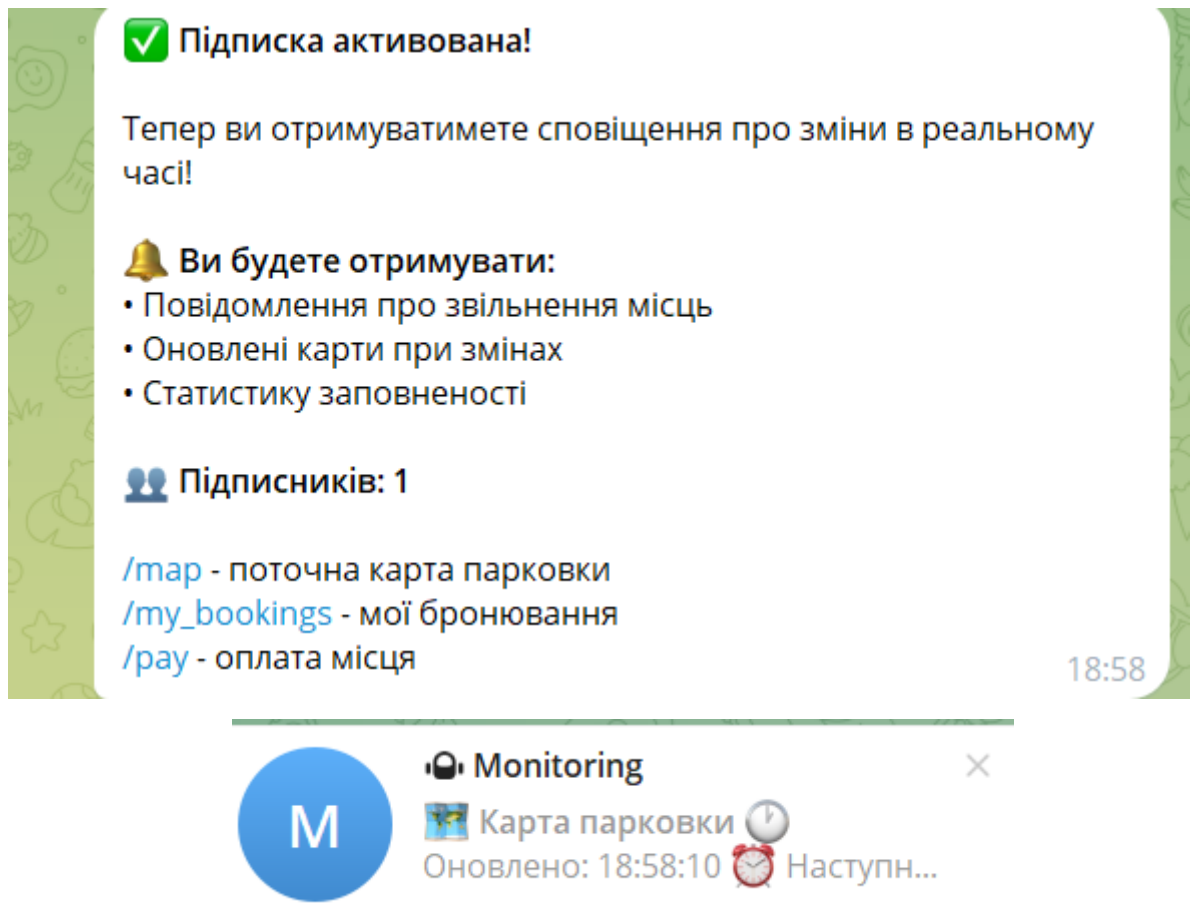


Рис. 3.13-14. Приклад отримання автоматичного сповіщення

Швидке бронювання через номер місця, функція дозволяє користувачам просто надіслати номер місця (1-59) для отримання опцій бронювання:

Реалізовано просту але ефективну систему перевірки прав.

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвєєв К.І.				54
Змн.	Арк.	№ докум.	Підпис	Дата		

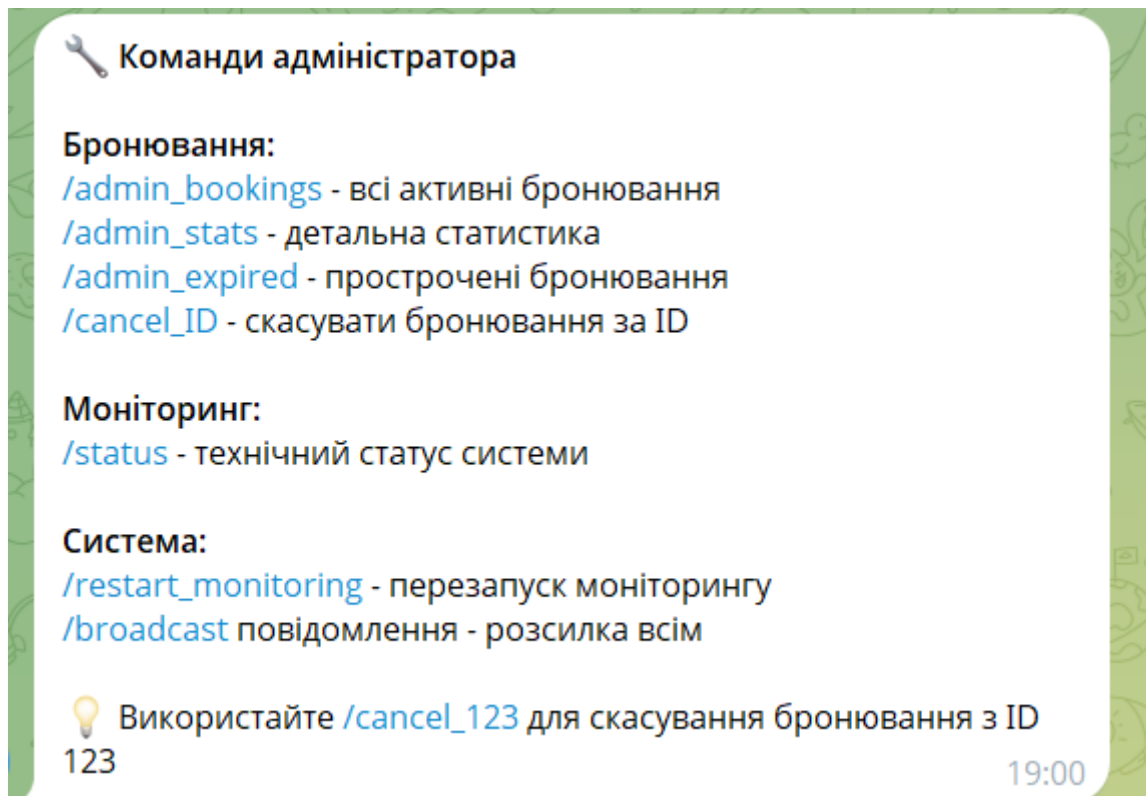


Рис. 3.15. Приклад роботи адміністративної команди (частина 2)

Команда /admin_bookings надає повний контроль над усіма бронюваннями з можливістю перегляду, групування за статусами та швидкого скасування:

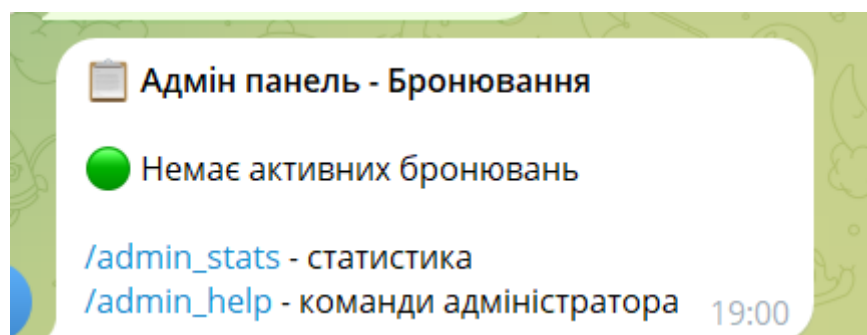


Рис. 3.16. Приклад роботи адміністративної команди (частина 2)

Команда /admin_stats розширена статистика для адміністраторів включає інформацію про платежі, ефективність системи, користувачів та технічні метрики.

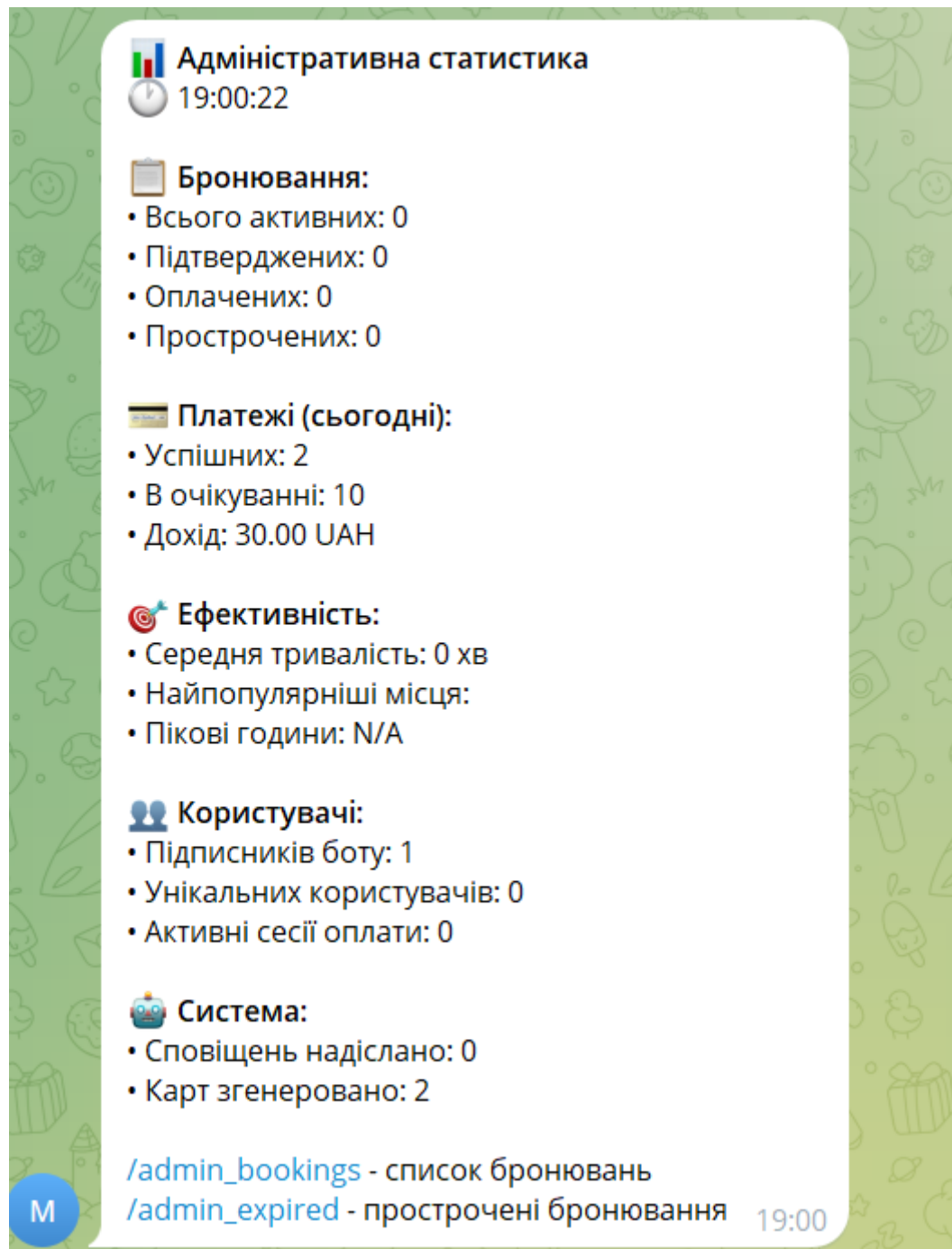


Рис. 3.17. Приклад роботи адміністративної команди (частина 3)

Підписники зберігаються у множині `self.subscribers`, що забезпечує швидкий доступ та автоматичне видалення дублікатів. Система автоматично видаляє неактивних користувачів при помилках доставки.

Реалізований Telegram-бот повністю відповідає поставленим вимогам та забезпечує комплексний інтерфейс для взаємодії з системою моніторингу паркувальних місць. Функціональне тестування підтвердило коректність роботи

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		56

всіх основних функцій, надійність системи сповіщень та ефективність інтеграції з іншими модулями комплексу.

Розроблений бот може слугувати основою для повномасштабного впровадження системи розумної парковки, забезпечуючи зручний та функціональний інтерфейс для всіх категорій користувачів.

3.3 Реалізація статичної системи візуалізації паркувальних місць та її інтеграція

Модуль статичної візуалізації паркувальних місць (`parking_map_generator.py`) відповідає за створення наочного зображення поточного стану всієї парковки у вигляді PNG-карти. Така карта використовується як основний візуальний інтерфейс для користувачів Telegram-бота, адміністраторів і в процесі автоматичних сповіщень. Реалізація модуля повністю відповідає проектним рішенням, викладеним у підрозділі 2.4, і забезпечує інтеграцію з базою даних та іншими компонентами системи.

Для генерації карти використовуються: NumPy — для створення полотна (фонового зображення) через `np.ones()`. OpenCV (`cv2`) — для малювання прямокутників, ліній, базового тексту. Pillow (PIL) — для нанесення тексту українською мовою, підтримки кирилиці та роботи зі шрифтами

Координати паркомісць визначаються вручну за допомогою окремого модуля `parking_space_picker.py`. Оператор відкриває зображення парковки, виділяє мишкою області для кожного місця, і ці координати зберігаються у файл `parking_positions.pkl`.

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				57
Змн.	Арк.	№ докум.	Підпис	Дата		



Рис. 3.18. Візуалізація як працює створення місць

Під час ініціалізації системи (наприклад, у класі `ParkingMonitor` або `ParkingDatabase`, метод `init_parking_spaces`) ці координати завантажуються з .pkl-файлу:

```
try:
    with open(self.positions_path, 'rb') as f:
        positions_data = pickle.load(f)

    if positions_data and isinstance(positions_data[0], list):
        print("🔄 Конвертація полігонів в прямокутники...")
        self.positions = []
        for polygon in positions_data:
            x_coords = [point[0] for point in polygon]
            y_coords = [point[1] for point in polygon]
            x1, x2 = min(x_coords), max(x_coords)
            y1, y2 = min(y_coords), max(y_coords)
            self.positions.append((x1, y1, x2, y2))
    else:
        self.positions = positions_data

    print(f"✅ Завантажено {len(self.positions)} паркувальних місць")
    self.database.init_parking_spaces(self.positions)
except FileNotFoundError:
    print(f"❌ Файл позицій не знайдено: {self.positions_path}")
    return False
```

Далі, у методі `init_parking_spaces` класу `ParkingDatabase`, кожне місце заноситься у БД з координатами у форматі JSON.

Створення полотна для карти та малювання слотів:

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				58
Змн.	Арк.	№ докум.	Підпис	Дата		

```
spaces = self.database.get_all_spaces()
cv2_image = np.ones((map_height, map_width, 3), dtype=np.uint8) * 240
current_space_idx = 0
for row in range(rows):
    for col in range(cols):
        if current_space_idx >= len(spaces):
            break
        space = spaces[current_space_idx]
        x_draw = col * (square_size + spacing) + spacing +
        y_draw = row * (square_size + spacing) + spacing + 80

        color = self.colors.get(space['status'], (128, 128, 128))
        cv2.rectangle(cv2_image, (x_draw, y_draw),
                      (x_draw + square_size, y_draw + square_size), color, -1)
        cv2.rectangle(cv2_image, (x_draw, y_draw),
                      (x_draw + square_size, y_draw + square_size), (0, 0, 0), 2)
        current_space_idx += 1
```

Нанесення номерів місць, статистики, легенди (з використанням PIL для кирилиці):

```
stats = self.database.get_statistics()
title_text = f"КАРТА ПАРКОВКИ - Всього місць: {stats['total_spaces']}"
self.add_text_pil(pil_image, title_text, (10, 5), font_size=24, color=(255, 255, 255))
stats_text = f"Вільних: {stats['free_spaces']} | Зайнятих: {stats['occupied_spaces']} | Забр.: {stats['reserved_spaces']}"
self.add_text_pil(pil_image, stats_text, (10, 35), font_size=16, color=(200, 200, 200))
```

Збереження згенерованої карти:

```
final_cv2_image = self.pil_to_cv2(pil_image)
cv2.imwrite(output_path, final_cv2_image)
```

У файлі `integrated_parking_system.py` при обробці команди `/map` бот викликає генерацію карти:

```
try:
    map_path = self.map_generator.generate_simple_map("current_telegram_map.png")
    if map_path:
        stats = self.database.get_statistics()
        caption = f"📍 Карта парковки\n" \
                  f"🟩 Вільно: {stats['free_spaces']}\n" \
                  f"🟡 Зайнято: {stats['occupied_spaces']}\n" \
                  f"🔴 Заброньовано: {stats['reserved_spaces']}"
        self.send_photo(chat_id, map_path, caption)
except Exception as e:
    self.send_message(chat_id, "❌ Помилка генерації карти")
```

Тобто бот отримує шлях до згенерованого PNG-файлу і надсилає його користувачу разом із підписом.

У `parking_monitor.py` після оновлення статусів паркомісць викликається оновлення карти у методі `update_parking_map`:

```
timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
map_filename = f"parking_map_{timestamp}.png"
map_path_history = os.path.join(self.maps_folder, map_filename)
generated_path = self.map_generator.generate_simple_map(map_path_history)
```

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		59

```

if generated_path:
    shutil.copy2(generated_path, self.current_map_path)
    self.last_map_update = time.time()
    return generated_path
return None

```

Ця згенерована карта (або шлях до неї) потім може бути використана для сповіщень через callback-механізм, як описано у `process_status_changes` та `notify_subscribers`.

Модуль візуалізації (`parking_map_generator.py`) використовує методи `ParkingDatabase`: `get_all_spaces()` — отримує список словників, де кожен словник містить `space_id`, `coordinates` (як JSON-рядок, який потрібно розпарсити), `status`, `get_statistics()` — отримує агреговану статистику.

Приклад отримання та використання даних у `ParkingMapGenerator`:

```

spaces = self.database.get_all_spaces()
stats = self.database.get_statistics()

```

Координати кожного місця — це список `[x1, y1, x2, y2]`, що зберігається у БД як JSON.

Такий підхід забезпечує просту, наочну і гнучку систему візуалізації, повністю інтегровану з усіма модулями комплексу.

3.4 Комплексне тестування взаємодії модулів MVP, опис користувацького інтерфейсу та ключові аспекти для демонстрації працездатності

Метою цього підрозділу є перевірка коректної спільної роботи всіх модулів MVP — від ШІ-модуля до Telegram-бота, а також детальний опис користувацького інтерфейсу та демонстрація ключових сценаріїв, що підтверджують працездатність і відповідність системи функціональним вимогам.

Telegram-бот реалізований через прямі HTTP-запити до Telegram Bot API (методи `get_bot_updates`, `handle_message`, `send_message`, `send_photo`), що забезпечує повний контроль над логікою та швидку реакцію на запити користувачів.

Для покращення читабельності всі повідомлення бота містять HTML-розмітку: заголовки, підсвічування статусів, емодзі для швидкого сприйняття.

Тестування проводилося за сценаріями, що охоплюють повний цикл роботи системи: від надходження нового зображення до оновлення карти, бази даних і

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		60

надсилання сповіщень користувачам. Перевірялися всі ключові ланцюжки передачі даних між модулями: ParkingMonitor → ParkingDatabase → ParkingMapGenerator → IntegratedParkingSystem (Telegram-бот).

Вибрані тестові сценарії:

- Сценарій 1: Повний цикл моніторингу та сповіщення.
- Сценарій 2: Бронювання місця користувачем із концептуальною оплатою.
- Сценарій 3: Скасування бронювання (користувачем та адміністратором).

Сценарій 1:

1. У папці camera 1 зберігаються зображення парковки.

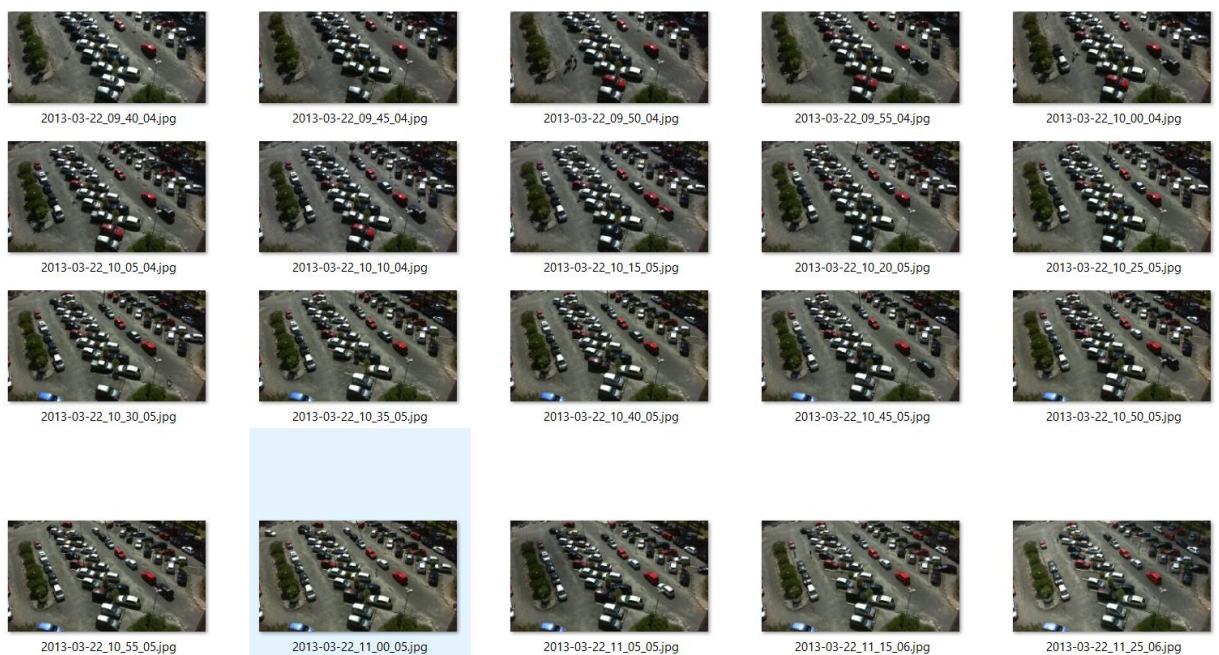


Рис. 3.19. Вміст папки camera1

2. ParkingMonitor аналізує зображення, визначає зміни статусів місць, оновлює їх у ParkingDatabase.

3. Генерується нова карта (parking_map_generator.py).

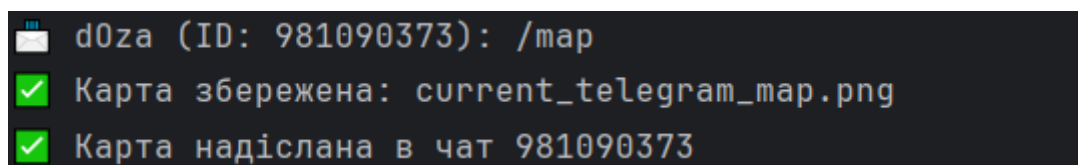


Рис. 3.20. Реагування на команду /map (частина 1)

4. Через callback IntegratedParkingSystem користувач отримує сповіщення



Рис. 3.21. Реагування на команду /map (частина 2)

Сценарій 2: Бронювання місця користувачем (з оплатою)

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвєєв К.І.				62
Змн.	Арк.	№ докум.	Підпис	Дата		

1. Користувач надсилає команду /pay 7.

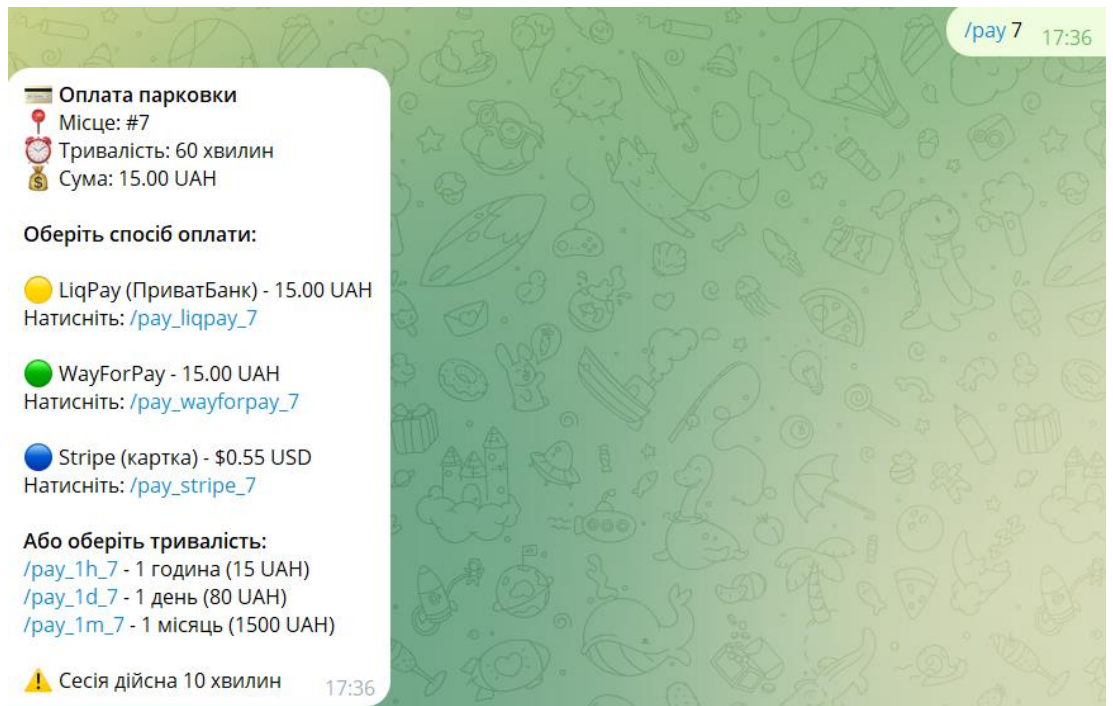


Рис. 3.22. Реагування на бронювання (частина 1)

2. Бот пропонує вибір платіжної системи та тривалості.

3. Користувач отримує "посилання" та payment_id.

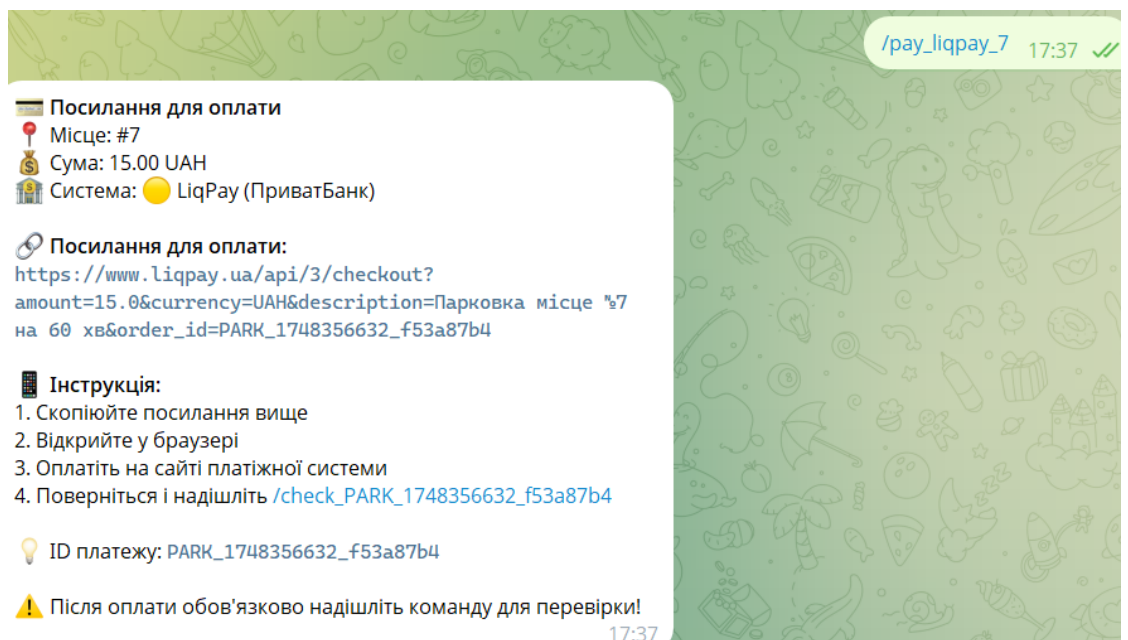


Рис. 3.23. Реагування на бронювання (частина 2)

4. Після імітації оплати користувач надсилає команду для перевірки (/check_<payment_id>).

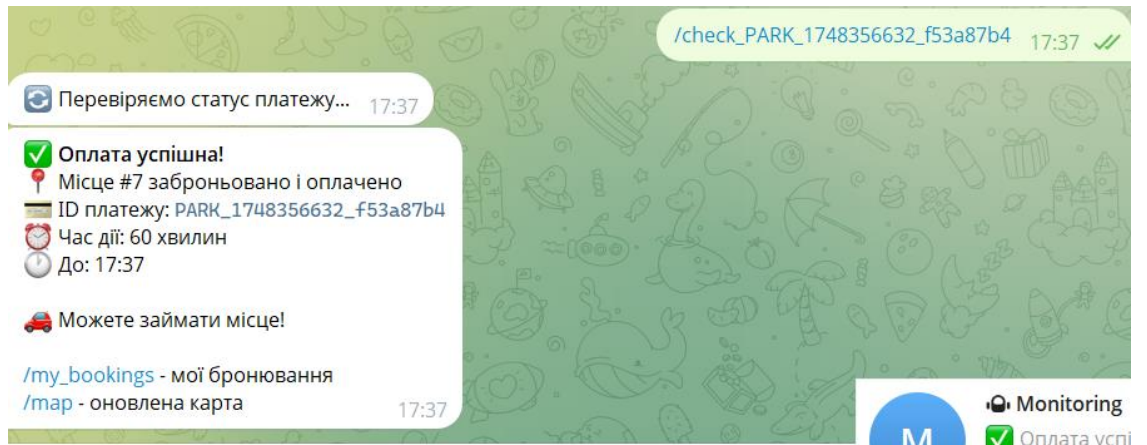


Рис. 3.24. Реагування на бронювання (частина 3)

5. SimplePaymentSystem імітує успішну оплату, ParkingDatabase оновлює статус місця на "reserved"/"paid".

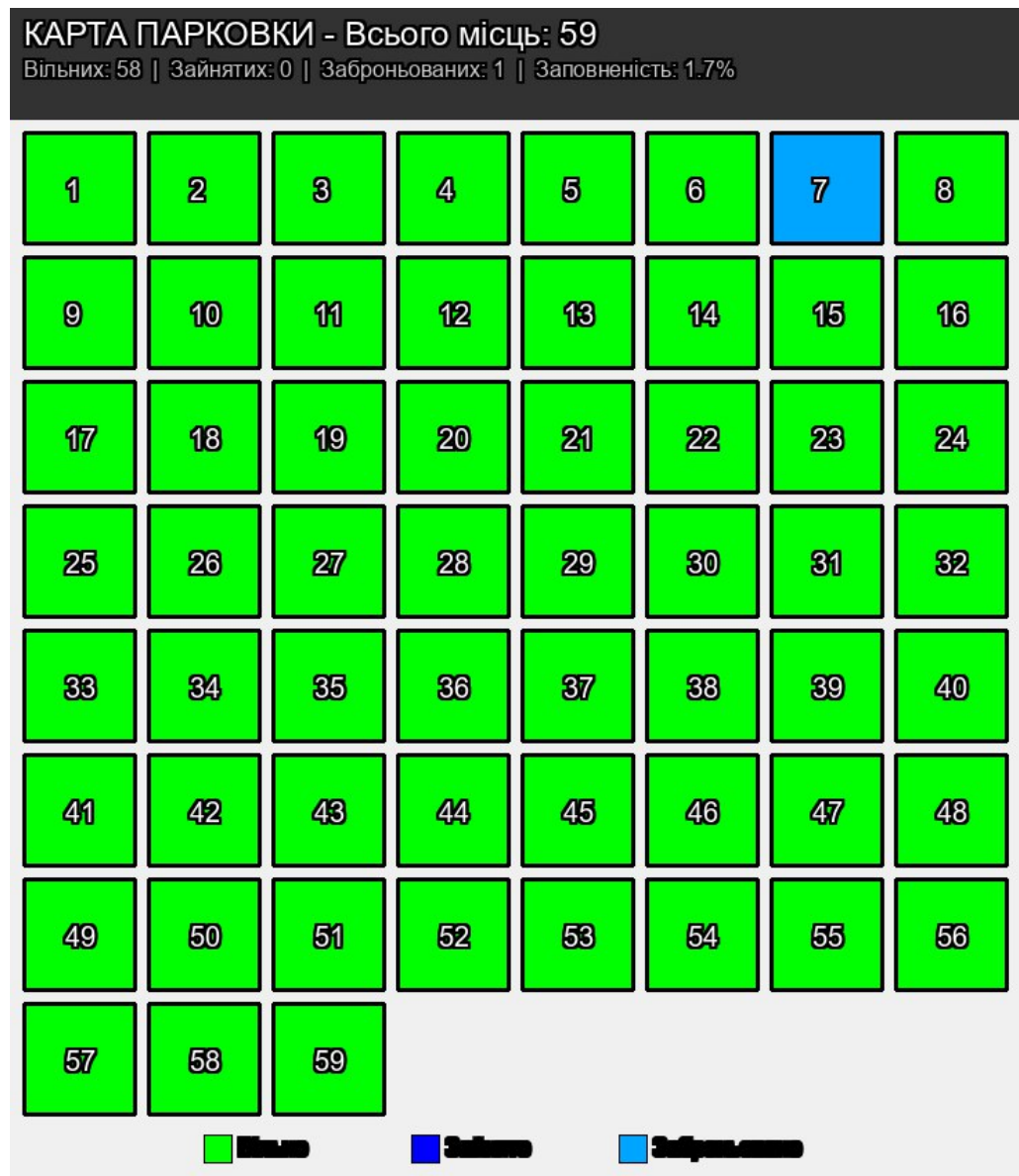


Рис. 3.25. Вивід карти парковки

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				64
Змн.	Арк.	№ докум.	Підпис	Дата		

6. Бот надсилає підтвердження користувачу

Сценарій 3: Скасування бронювання

1. Користувач надсилає `/cancel <ID_бронювання>`, адміністратор — `/admin_cancel <ID_бронювання>`.

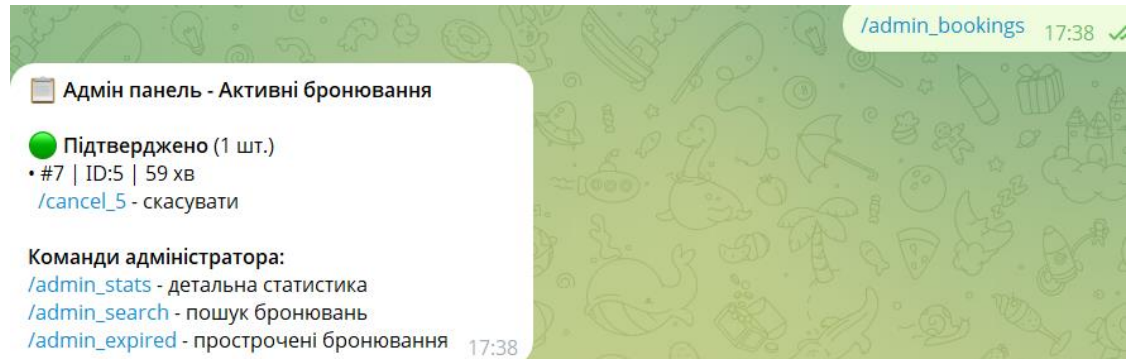


Рис. 3.26. Тестування адмін панелі

2. ParkingDatabase змінює статус бронювання та місця.

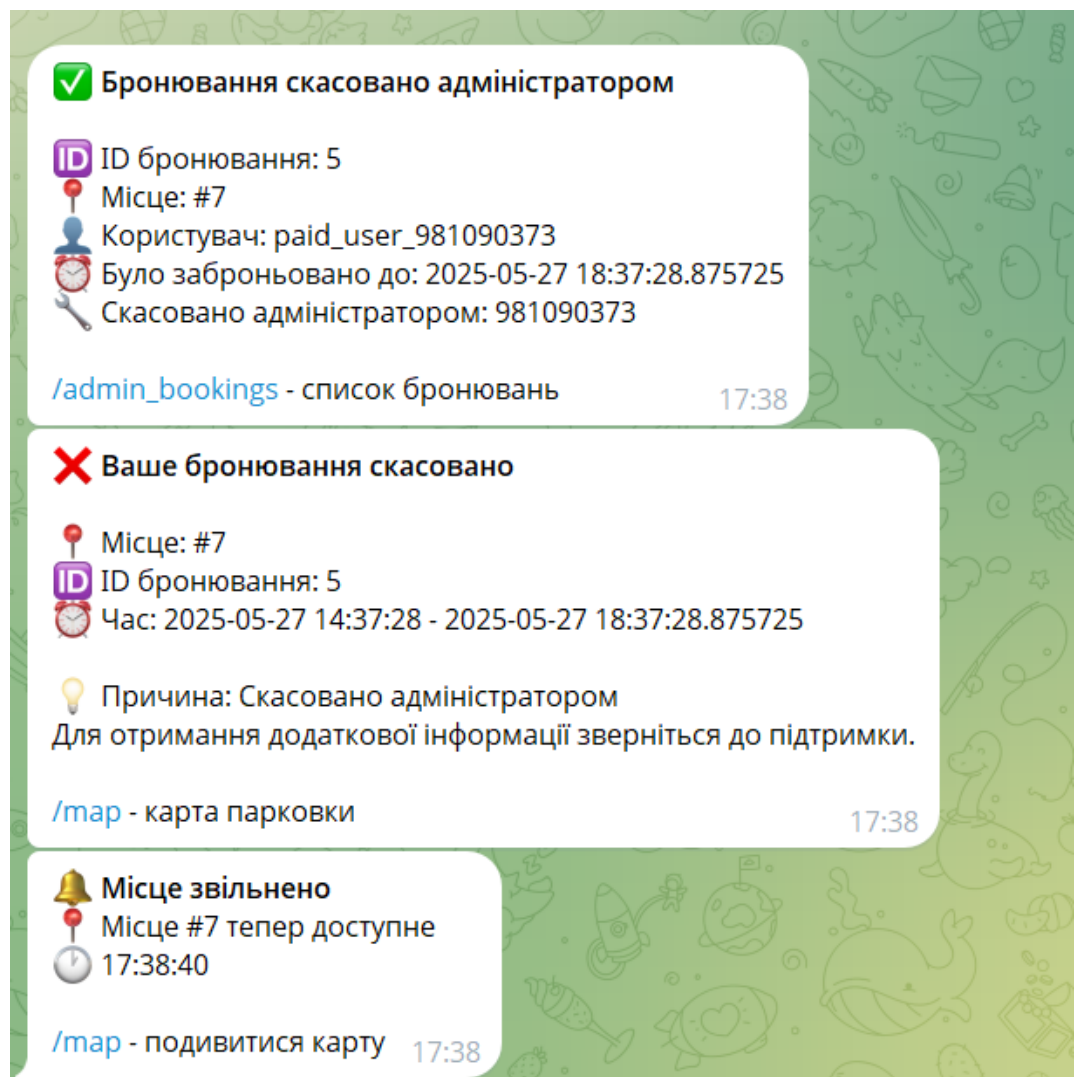


Рис. 3.27. Успішне скасування бронювання

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвєєв К.І.				65
Змн.	Арк.	№ докум.	Підпис	Дата		

3. Бот надсилає підтвердження ініціатору та сповіщення всім підписникам.

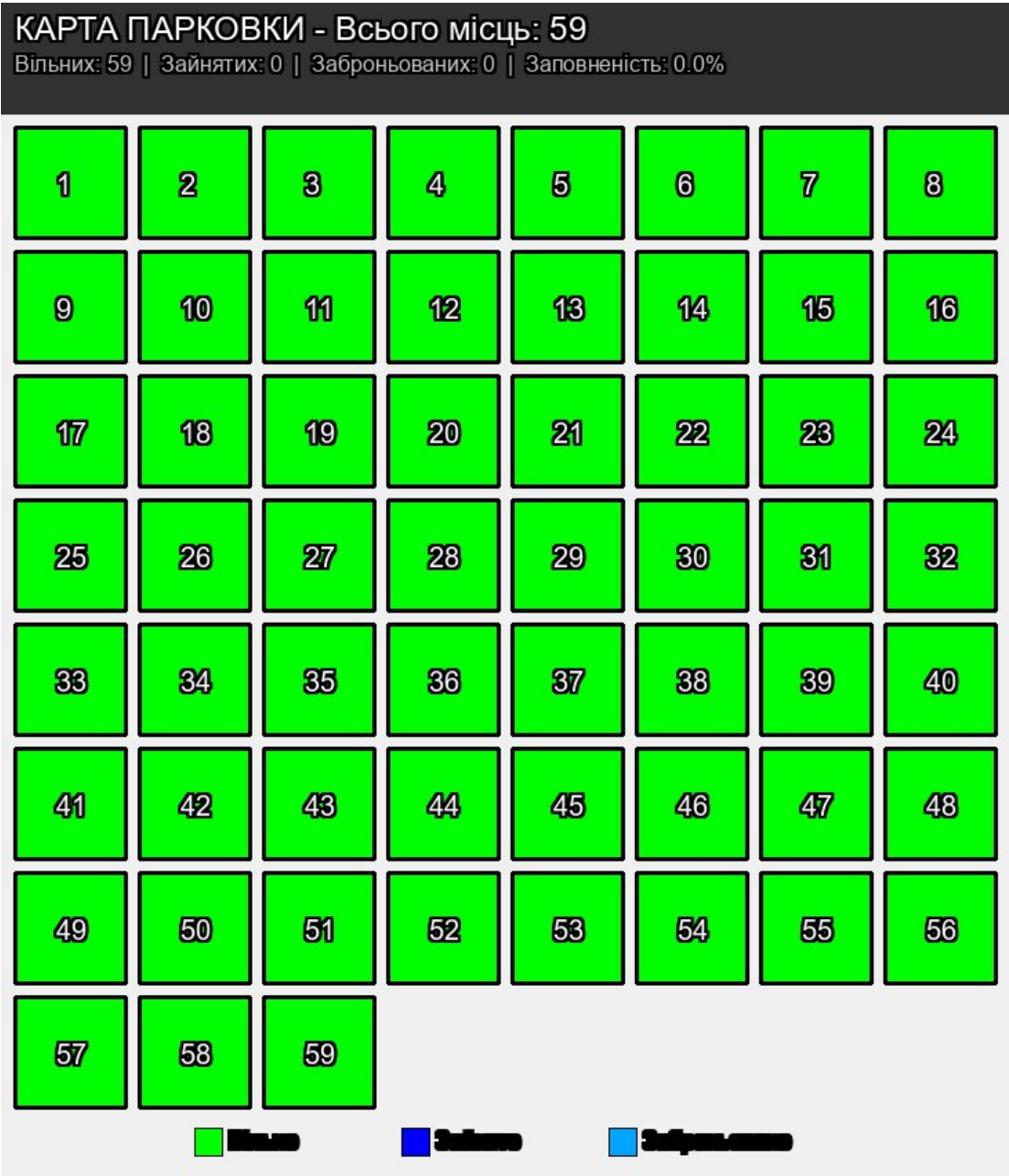


Рис. 3. 28. Карта парковки після видалення бронювання

У всіх сценаріях перевірено коректність передачі даних, швидкість реакції системи (1–3 секунди), відсутність зависань чи втрат інформації.

Проведене комплексне тестування підтвердило, що всі модулі MVP коректно взаємодіють між собою, а система повністю відповідає функціональним і нефункціональним вимогам. Інтерфейс Telegram-бота є зручним, інформативним і підтримує всі необхідні сценарії для користувачів та адміністраторів. Реалізована архітектура забезпечує стабільність, надійність і гнучкість для подальшого

розвитку системи, а також дозволяє ефективно продемонструвати основні функції на захисті.

Висновок до третього розділу

У третьому розділі було представлено програмну реалізацію та результати тестування розробленого прототипу системи моніторингу паркувальних місць. Ключовим етапом стала реалізація модуля розпізнавання зайнятості паркомісць на основі згорткової нейронної мережі, яка після навчання на датасеті PKLot та відповідної підготовки даних продемонструвала належну точність класифікації стану окремих слотів. Процес навчання та оцінки моделі, включаючи використання технік аугментації та callback-функцій, був детально описаний та підкріплений візуальними результатами.

Було реалізовано модуль взаємодії з користувачем через Telegram-бот, що забезпечує основний інтерфейс до функціоналу системи. Описано логіку обробки команд користувачів та адміністраторів, а також механізми сповіщень, реалізовані через пряму взаємодію з Telegram Bot API. Функціональне тестування бота підтвердило коректність обробки запитів та взаємодії з іншими компонентами.

Також описано реалізацію системи статичної візуалізації паркувальних місць у вигляді карти, що генерується на основі актуальних даних з бази SQLite. Показано інтеграцію цього модуля з Telegram-ботом для надання користувачам наочної інформації про стан парковки.

На завершення, проведено комплексне тестування взаємодії всіх модулів MVP через ключові сценарії використання, такі як автоматичне оновлення стану парковки та сповіщення, а також процес бронювання місця. Тестування підтвердило коректну спільну роботу розроблених компонентів, відповідність системи основним функціональним вимогам та готовність прототипу до демонстрації. Реалізовані програмні рішення створюють міцну основу для подальшого вдосконалення та розвитку системи.

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				67
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

У рамках даної кваліфікаційної роботи було успішно розроблено та досліджено прототип програмно-апаратного комплексу для автоматизованого моніторингу зайнятості паркувальних місць, що відповідає поставленій меті. Система базується на аналізі візуальної інформації за допомогою згорткової нейронної мережі та забезпечує взаємодію з користувачами через Telegram-бот.

Протягом роботи було виконано наступні ключові завдання: проведено аналіз існуючих технологій моніторингу та обґрунтовано вибір методів для реалізації; розроблено загальну архітектуру системи, що включає концепцію масштабованого розгортання; спроектовано та реалізовано основні програмні модулі. Зокрема, створено модуль розпізнавання на основі CNN, який після навчання на датасеті PKLot продемонстрував ефективність у класифікації стану паркомісць. Розроблено функціональний Telegram-бот для взаємодії з користувачами, що підтримує перегляд карти парковки, статистики, а також концептуальне бронювання та оплату. Реалізовано модуль генерації статичної карти парковки та механізми інтеграції даних через базу SQLite та callback-систему.

Практичне значення роботи полягає у створенні прототипу, що демонструє можливість автоматизації моніторингу парковок та надання користувачам зручного доступу до актуальної інформації, сприяючи оптимізації використання паркувального простору. Комплексне тестування підтвердило коректну взаємодію реалізованих модулів MVP та відповідність системи функціональним вимогам.

Результати роботи можуть слугувати основою для подальшого розвитку системи, зокрема, для інтеграції з реальними камерами, розширення функціоналу (наприклад, розпізнавання номерних знаків), впровадження більш просунутих ШІ-моделей та повноцінного розгортання у запроєктованій масштабованій хмарній архітектурі.

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		68

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Chen X., Qian Z., Rajagopal R., Stiers T., Flores C., Kavalier R., Williams F. III. Smart Parking Sensing and Information System: Design, Implementation and Evaluation // arXiv preprint arXiv:1712.02741. – 2017. – URL: <https://arxiv.org/pdf/1712.02741>
2. Krzebietke S., et al. Effect of manure and mineral fertilisers on the content of PAHs in soil // Scientific Reports. – 2025. – URL: <https://www.nature.com/articles/s41598-025-86441-w>
3. Sophisticated parking sensor Urbiotica // ITS International. – 2024. – URL: <https://www.itsinternational.com/daily-news/intertraffic-amsterdam-2024/sophisticated-parking-sensor-urbiotica>
4. Car Parking Management Solution // OPTEX ASEAN. – URL: <https://optex-asean.com/solution/car-parking-management-solution/>
5. China Mobile Smart Parking Internet of Things Case Study // GSMA. – URL: https://www.gsma.com/solutions-and-impact/technologies/internet-of-things/gsma_resources/china-mobile-smart-parking-internet-of-things-case-study/
6. Smart Parking System // TIMES24 Malaysia. – URL: <https://www.times-parking.com.my/smart-parking-system/>
7. Al-Turjman F., Malekloo A. Smart Parking Systems: A Comprehensive Review // Sensors. – 2021. – URL: <https://pmc.ncbi.nlm.nih.gov/articles/PMC8464435/>
8. Zhang J., Wang Y., et al. Parking Lot Occupancy Detection Based on Deep Learning // arXiv preprint arXiv:2412.01983. – 2024. – URL: <https://arxiv.org/html/2412.01983v1>
9. Machine Learning Based Parking Space Classification Using R-CNN and Faster R-CNN FPN Architecture // MIDI 2024. – URL: <https://midi2024.opi.org.pl/wp-content/uploads/2024/12/M1-08-Machine-Learning-Based-Parking-Space-Classification-Using-R-CNN-and-Faster-R-CNN-FPN-Architecture.pdf>

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				69
Змн.	Арк.	№ докум.	Підпис	Дата		

10. Kovalchuk S., et al. Automated Parking Lot Occupancy Detection Using Computer Vision // CEUR Workshop Proceedings. – 2018. – URL: <https://ceur-ws.org/Vol-2087/paper5.pdf>
11. CNRPark-EXT Dataset // CNRPark. – URL: <http://cnrpark.it/>
12. Monitoring Software // ParkSol. – URL: <https://parksol.lt/solutions/monitoring-software/>
13. Parking Lot Occupancy Detection // Viso Suite. – URL: <https://viso.ai/applications/parking-lot-occupancy-detection/>
14. Stanford CS231n Course Project Report: Parking Lot Occupancy Detection // Stanford University. – 2016. – URL: https://cs231n.stanford.edu/reports/2016/pdfs/280_Report.pdf
15. Kurniawan R., et al. Deep Learning Based Parking Space Detection: A Review // SSRN. – 2023. – URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4475267
16. Train-Test Split in Machine Learning // Built In. – URL: <https://builtin.com/data-science/train-test-split>
17. Kovalchuk S., et al. Parking Lot Occupancy Detection Using Deep Learning // CEUR Workshop Proceedings. – 2022. – URL: <https://ceur-ws.org/Vol-3156/paper21.pdf>
18. PKLot Dataset on Roboflow // Roboflow Universe. – URL: <https://public.roboflow.com/object-detection/pklot>
19. PKLot Dataset ReadMe // Federal University of Paraná. – URL: <https://www.inf.ufpr.br/lesoliveira/download/pklot-readme.pdf>
20. Parking Lot Occupancy Detection // GitHub Repository: deep-parking. – 2018. – URL: <https://github.com/fabiocarrara/deep-parking/blob/master/pklot.py>

ДОДАТКИ

		Зіневич Н.Я.			ККІКБ.КР.Б – 123 – 4 – ПЗ	Арк.
		Матвеев К.І.				71
Змн.	Арк.	№ докум.	Підпис	Дата		

