Міністерство освіти і науки України
Криворізький національний університет
Кафедра моделювання та програмного забезпечення

**КВАЛІФІКАЦІЙНА РОБОТА**

**на здобуття ступеня вищої освіти магістра**

зі спеціальності 121 – Інженерія програмного забезпечення

На тему: Дослідження та розробка програмного модуля керування ігровими стратегіями

Засвідчую, що в цій кваліфікаційній роботі немає запозичень із праць інших авторів без відповідних посилань.
Студент гр. ІПЗ-23-2м
_____ /О. І. Ісаєнко /

Керівник
кваліфікаційної роботи _____ / Н. Х. Саітгареєв /
Економіко-організаційна
частина _____ / О. В. Шамрай /
Нормоконтроль _____ / _____/
Завідувач кафедри _____ / А. М. Стрюк /

Кривий Ріг
2024

Криворізький національний університет

Факультет: Інформаційних технологій

Кафедра: Моделювання та програмного забезпечення

Ступінь вищої освіти: магістр

Спеціальність: 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. кафедри

_____ А.М. Стрюк

«__» _____ 20__ р.

**ЗАВДАННЯ**

**на кваліфікаційну роботу**

студенту групи ІПЗ-19-2 Ісаєнко Олександру Ігоровичу

1. Тема: Дослідження та розробка програмного модуля керування ігровими стратегіями затверджено наказом по КНУ № __ від «_» _____ 2024 р.
2. Термін подання студентом закінченої роботи: «_» ____2024 р.
3. Вихідні дані по роботі: розроблюваний програмний модуль повинен працювати з скріншотами різної якості та надавати релевантну інформацію в залежності від результатів обробленого скріншоту.
4. Зміст пояснювальної записки (перелік питань, що їх треба розробити): виконати аналіз предметної області та існуючих аналогів зі схожим функціоналом, змоделювати сценарії застосування програмного комплексу, змоделювати базу даних, здійснити програмну реалізацію програмного модулю, провести тестування розробленого модулю, виконати аналіз економічної ефективності розроблюваного комплексу.
5. Перелік ілюстративного матеріалу: діаграма застосування, блок-схеми розроблених алгоритмів, діаграма бази даних, знімки екранних форм.

Календарний план:

| № | Найменування етапів кваліфікаційної роботи | Термін виконання етапів роботи |
|---|---|---|
| 1 | Ознайомлення з літературою за тематикою роботи та збір даних | 23.12.2023 – 22.01.2024 |
| 2 | Проведення аналізу предметної області та існуючих аналогів програмних модулів | 23.01.2024 – 15.02.2024 |
| 3 | Формулювання актуальності та завдань роботи | 16.02.2024– 25.02.2024 |
| 4 | Підготовка та оформлення матеріалів першого розділу | 26.02.2024 – 20.03.2024 |
| 5 | Моделювання сценарію взаємодії програмного модулю та кінцевого користувач | 21.03.2024 – 29.04.2024 |
| 6 | Розробка алгоритмів та функціональних схем роботи програмного модулю | 30.04.2024– 09.05.2024 |
| 7 | Підготовка та оформлення матеріалів другого розділу | 10.05.2024– 30.06.2024 |
| 8 | Розробка Discord чат-боту та моделі комп’ютерного зору | 01.07.2024– 09.08.2024 |
| 9 | Підготовка та оформлення матеріалів третього розділу | 10.08.2024 – 14.09.2024 |
| 10 | Економічний аналіз та оцінка ефективності роботи | 15.09.2024 – 29.10.2024 |
| 11 | Підготовка та оформлення матеріалів четвертого розділу | 30.10.2024– 24.11.2024 |
| 12 | Остаточне оформлення пояснювальної записки | 25.11.2024 – 05.12.2024 |

Дата видачі завдання: «_»_____ 2024 р.

Студент _____ / О. І. Ісаєнко /

Керівник роботи _____ / Н. Х. Саітгареєв /

# РЕФЕРАТ

РОЗПІЗНАВАННЯ ОБ’ЄКТІВ, КОМП’ЮТЕРНИЙ ЗІР, МАШИННЕ НАВЧАННЯ, CLASH OF CLANS, DISCORD-БОТ, ROBOFLOW, АНАЛІЗ ЗОБРАЖЕНЬ, ІГРОВІ СТРАТЕГІЇ, АВТОМАТИЗАЦІЯ АНАЛІЗУ.

Пояснювальна записка: 73 с., 3 табл., 37 рис., 20 джерел.

Метою кваліфікаційної роботи є розробка багатофункціонального програмного комплексу, що буде у нагоді мільйонам гравців по всьому світу в багатокористувацьку мобільну стратегію «Clash of Clans», що мають бажання підвищити свій рівень планування і підготовки до атак, тобто найважливішого ігрового процесу.

Методами дослідження роботи є модель, що була навчена у середовищі RoboFlow із використанням хмарної інфраструктури, попередньо навчена нейронна мережа YOLOv5 для розпізнавання будівель на зображеннях, обробка даних із застосуванням бібліотеки OpenCV для підготовки зображень до аналізу, інтеграція з Discord API для забезпечення інтерактивної взаємодії з користувачами.

Основними характеристиками роботи є: автоматичне розпізнавання будівель за скріншотом бази, інференс у хмарному середовищі з мінімальними затримками, інтеграція з платформою Discord для забезпечення доступу до функцій модуля.

Результатом роботи є створений програмний модуль, який об'єднує сучасні методи комп'ютерного зору для автоматичного аналізу ігрових баз.

Результати роботи, а саме програмний модуль рекомендовано використовувати цільовій аудиторії багатокористувацької гри Clash of Clans, тобто її гравцям, котрі хочуть грати з більшою ефективністю і задоволенням від ігрового процесу.

У майбутньому, за потреби, функціонал програмного модуля можна збільшувати, додаючи нові можливості та розширюючи підтримувані рівні ратуші для сканування.

# ABSTRACT

OBJECT RECOGNITION, COMPUTER VISION, MACHINE LEARNING, CLASH OF CLANS, DISCORD BOT, ROBOFLOW, IMAGE ANALYSIS, GAME STRATEGIES, ANALYSIS AUTOMATION.

Explanatory note: 73 p., 3 tab., 37 fig., 20 references.

The goal of the qualification work is to develop a multifunctional software package that will be useful to millions of players around the world in the multiplayer mobile strategy game "Clash of Clans" who want to improve their level of planning and preparation for attacks, i.e. the most important gameplay.

The research methods of the work are a model trained in the RoboFlow environment using cloud infrastructure, a pre-trained YOLOv5 neural network for recognizing buildings in images, data processing using the OpenCV library to prepare images for analysis, and integration with the Discord API to provide interactive interaction with users.

The main features of the work are: automatic recognition of buildings from a screenshot of the database, inference in the cloud environment with minimal delays, integration with the Discord platform to provide access to the module's functions.

The result of the work is a software module that combines modern methods of computer vision for automatic analysis of game bases.

The results of the work, namely the software module, are recommended for use by the target audience of the multiplayer game Clash of Clans, i.e. its players who want to play with greater efficiency and enjoyment of the game.

In the future, if necessary, the functionality of the software module can be increased by adding new features and expanding the supported levels of the town hall for scanning.

**ЗМІСТ**

# ВСТУП

Протягом декількох останніх років чат-боти стали нашими невід’ємними компаньйонами у житті. Завдяки ним та мережі Інтернет кожен з нас може миттєво подивитися графіки стабілізаційних відключень електроенергії, передати показання лічильника газу, зв’язатися зі службою підтримки майже будь-якої компанії, ну і, звісно, хоч якимось чином зайняти наші думки, щоб хоч трохи відволіктися від усього, що відбувається навколо.

Стратегічні ігри, такі як Clash of Clans, набули популярності завдяки тактичному плануванню та змагальним елементам. Успіх у грі вимагає аналізу баз противника й оптимального прийняття рішень, що потребує часу та досвіду. Основними викликами для гравців є: варіативність розташувань баз – мільйони комбінацій ускладнюють пошук ефективної стратегії атаки; відсутність швидких інструментів аналізу – гравці витрачають багато часу на пошук відповідних відео або стратегій; комп'ютерний зір – незважаючи на розвиток технологій, його застосування у стратегічних іграх є обмеженим, але має потенціал для автоматизації аналізу й пошуку рішень.

Світові тенденції демонструють активний розвиток технологій комп'ютерного зору, штучного інтелекту і автоматизації у вирішенні завдань, пов'язаних з аналізом стратегічних ігор. Розробка Discord-бота, що автоматизує процес розпізнавання баз у Clash of Clans і пошук релевантних відеостратегій на YouTube, повністю відповідає сучасним викликам і запитам ігрових спільнот. Цей проєкт має великий потенціал для подальшого розвитку в напрямку автоматизації ігрових рішень.

Питання про актуальність використання технологій комп’ютерного зору у 2024 році напевно є риторичним, бо комп’ютерний зір зараз застосовується починаючи від сфери медицини, де аналізують медичні зображення для діагностики хвороб, сфери безпеки, де системи відеоспостереження вміють розпізнавати осіб, їхні обличчя та навіть підозрілу поведінку, безпілотному транспорті, де буквально уся обстановка на дорозі сприймається за допомогою

камер, радарів та лідарів і до ігрових технологій, де за допомогою комп’ютерного зору можна розпізнавати будь-які потрібні об’єкти, їх патерни, оптимізовувати графіку та використовувати розширену реальність (AR).

Об’єктом дослідження кваліфікаційної роботи є процес планування ігрових стратегій у мобільній грі Clash of Clans. А предметом дослідження є максимальне покращення результатів гравця під час атаки шляхом аналізу ігрової стратегії перед атакою у грі.

Метою кваліфікаційної роботи є розробка багатофункціонального програмного комплексу, що буде у нагоді мільйонам гравців по всьому світу в багатокористувацьку мобільну стратегію «Clash of Clans», що мають бажання підвищити свій рівень планування і підготовки до атак, тобто найважливішого ігрового процесу. Галуззю застосування такого програмного модулю є багатомільйонна цільова аудиторія мобільної гри Clash of Clans.

Для досягнення мети дослідження використано такі методи:

Аналіз літератури та існуючих рішень – вивчено наукові статті й технічну документацію для оцінки стану проблеми та обґрунтування актуальності підходу.

Метод комп'ютерного зору – розроблено нейронні мережі для автоматичного розпізнавання будівель на скріншотах із Clash of Clans, застосовуючи CNN для точного й швидкого аналізу.

Інтеграція з YouTube – автоматизовано пошук відео стратегій на основі ідентифікованих об'єктів.

Модульний підхід – створено незалежні модулі для обробки зображень, пошуку відео, інтеграції з Discord і взаємодії з користувачем.

Емпіричне тестування – перевірено працездатність модулів на реальних даних для оцінки точності та релевантності результатів.

Порівняльний аналіз – зіставлено результати автоматизованого підходу з ручним пошуком, що підтвердило ефективність і практичну користь запропонованого рішення.

# 1 АНАЛІТИЧНИЙ ОГЛЯД І ПОСТАНОВКА ЗАВДАННЯ

## 1.1 Огляд актуальних рішень у сфері комп’ютерного зору

На даний момент часу можна виділити кілька основних напрямків розвитку комп’ютерного, в яких відбувається дуже активний рух від світових IT-гігантів, серед яких:

- **Генеративний штучний інтелект та моделі дифузії**

Генеративні моделі, такі як GAN (Generative Adversarial Networks) та моделі дифузії, займають важливе місце у сучасних дослідженнях. Вони використовуються для створення синтетичних зображень та відео, що можуть бути застосовані у різних сферах, включаючи розваги, дизайн та медицину.

Приклади та використання:

- **DALL-E від OpenAI**: - це нейронна мережа, розроблена OpenAI, яка створює зображення з текстових підписів. Вона може створювати різноманітні зображення, включаючи антропоморфізовані версії тварин та об’єктів, поєднуючи непов’язані концепції правдоподібними способами, візуалізуючи текст та застосовуючи трансформації до існуючих зображень. DALL·E базується на архітектурі Transformer, як і GPT-3, і отримує як текст, так і зображення у вигляді єдиного потоку даних, що містить до 1280 токенів. Ця процедура навчання дозволяє DALL·E не лише генерувати зображення з нуля, але й регенерувати будь-яку прямокутну область існуючого зображення, що розширюється до нижнього правого кута, таким чином, щоб це відповідало текстовій підказці. DALL·E здатний:
    - Контролювати атрибути: Змінювати колір, розмір, кількість об'єктів тощо.
    - Малювати кілька об'єктів: Розміщувати об'єкти відносно один одного, складати їх, контролювати кілька атрибутів одночасно.

- Візуалізувати перспективу та тривимірність: Керувати точкою зору сцени, стилем рендеринга, створювати анімації обертання об’єкта, застосовувати оптичні спотворення, генерувати відображення.
- Візуалізувати внутрішню та зовнішню структуру: Рендерити внутрішню структуру за допомогою поперечних перерізів та зовнішню структуру за допомогою макрофотографій.
- Виводити контекстуальні деталі: Доповнювати опис відсутніми деталями, змінювати стиль, оточення та час, малювати той самий об'єкт у різних ситуаціях, додавати текст на зображення.
- Поєднувати непов’язані концепції: Створювати зображення реальних та уявних речей, поєднуючи різні ідеї, наприклад, переносячи якості з одних концепцій на інші, або проектуючи продукти, натхненні несподіваними джерелами.
- Проводити візуальні міркування без попереднього навчання (zero-shot reasoning): Виконувати завдання з перетворення зображень на основі текстових інструкцій, аналогічно GPT-3.
- Демонструвати географічні та часові знання: Створювати зображення, що відображають знання про географічні факти, пам'ятки, околиці та історичні періоди [1]. Ось приклади того, що DALL·E 3 зробив за наступними промптами:

«Ілюстрація авокадо, що сидить у кріслі терапевта і каже: "Я відчуваю таку порожнечу всередині", з діркою розміром з кісточку в центрі. Терапевт-ложка, робить нотатки.» (див. рис. 1.1);

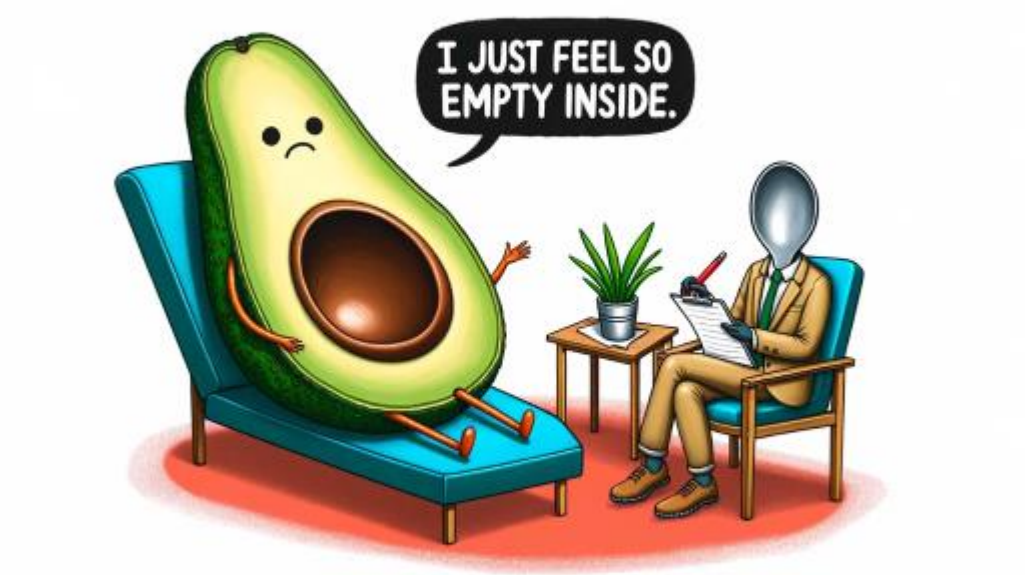


Рисунок 1.1 - Зображення, згенероване за допомогою DALL·E 3

«На глибокому чорному тлі зображена постать середніх років, з насиченою і сяючою тонганською шкірою, в центрі вихору, кучеряве волосся розвівається за її спиною, як буря. Її вбрання нагадує вихор мармурових та порцелянових уламків. Освітлена блиском розсипаних порцелянових уламків, що створюють казкову атмосферу, танцівниця виглядає фрагментарною, проте зберігає гармонійну та плавну форму.» (див. рис. 1.2);

Рисунок 1.2 - Зображення, згенероване за допомогою DALL·E 3

«Перед глядачем розгортається величезний пейзаж, повністю зроблений з різних видів м'яса. ніжні, соковиті пагорби ростбіфа, дерева курячих гомілок, річки бекону і валуни шинки створюють сюрреалістичну, але апетитну картину. небо прикрашають сонце з пепероні і хмари з салямі.» (див. рис. 1.3);

Рисунок 1.3 - Зображення, згенероване за допомогою DALL·E 3

- **BigGAN**: це модель генеративно-змагальних мереж (GAN), розроблена для створення високоякісних та різноманітних зображень. BigGAN демонструє покращені показники Inception Score (IS) та Frechet Inception Distance (FID) порівняно з попередніми моделями, що свідчить про високу якість згенерованих зображень.
- **Моделі дифузії**: Покращують якість зображень шляхом поступової модифікації випадкових шумів до чітких зображень.
    - **Виявлення об'єктів у реальному часі та їх відстеження:**

Цей напрямок важливий для автономних транспортних засобів, систем відеоспостереження та доповненої реальності. Алгоритми, такі як YOLO, забезпечують швидке та точне виявлення об'єктів.

Приклади та використання:

- **YOLOv7**: це сучасна система виявлення об'єктів у реальному часі, яка відзначається високою точністю та швидкістю. Вона здатна працювати з різними розмірами зображень, забезпечуючи оптимальний баланс між швидкістю та точністю. YOLOv7 досягає вражаючих результатів точності, перевершуючи деякі моделі Faster R-CNN, що традиційно вважаються більш точними.

- **SSD (Single Shot MultiBox Detector)**: це метод виявлення об'єктів на зображеннях, який використовує єдину глибоку нейронну мережу. На відміну від методів, які потребують окремого етапу формування пропозицій щодо розташування об’єктів, SSD усуває необхідність генерації пропозицій та подальшого перерахунку пікселів або ознак, інкапсулюючи всі обчислення в одній мережі. Працює SSD наступним чином: спочатку дискретизує простір - SSD розбиває вихідний простір обмежуючих рамок на набір стандартних рамок (default boxes) з різними співвідношеннями сторін та масштабами для кожного місця на карті ознак; виконує прогнозування - під час передбачення мережа генерує бали для наявності кожної категорії об’єктів у кожній стандартній рамці та вносить коригування в рамку, щоб краще відповідати формі об’єкта; та нарешті займається комбінуванням прогнозів - SSD комбінує прогнози з кількох карт ознак з різною роздільною здатністю, щоб природним чином обробляти об'єкти різних розмірів.

До переваг SSD можна віднести: простоту - SSD простіше в навчанні та інтеграції в системи, які потребують компонента виявлення, порівняно з методами, що вимагають пропозицій щодо об'єктів; швидкість - SSD набагато швидший, ніж методи з етапом пропозицій щодо об'єктів, забезпечуючи при цьому уніфіковану рамку як для навчання, так і для висновків; точність - SSD має точність, порівнянну з методами, які використовують додатковий крок пропозицій щодо об'єктів. На вхідному зображенні розміром 300x300 SSD досягає 72,1% mAP на тесті VOC2007 зі швидкістю 58 FPS на Nvidia Titan X, а для вхідного зображення 500x500 SSD досягає 75,1% mAP, перевершуючи аналогічну сучасну модель Faster R-CNN [2].

- **3D реконструкція та розуміння сцен**

Реконструкція 3D середовищ з 2D-зображень дозволяє створювати точні тривимірні моделі, що корисно в AR/VR та робототехніці. Сучасні методи включають нейронні радіальні поля (NeRF) та Structure from Motion (SfM).

Приклади та використання:

- **Instant NeRFs від NVIDIA**: це революційна технологія, що дозволяє швидко і легко перетворювати набір 2D зображень на якісні 3D сцени, використовуючи нейронні поля випромінювання (NeRF) (див. рис. 1.4).

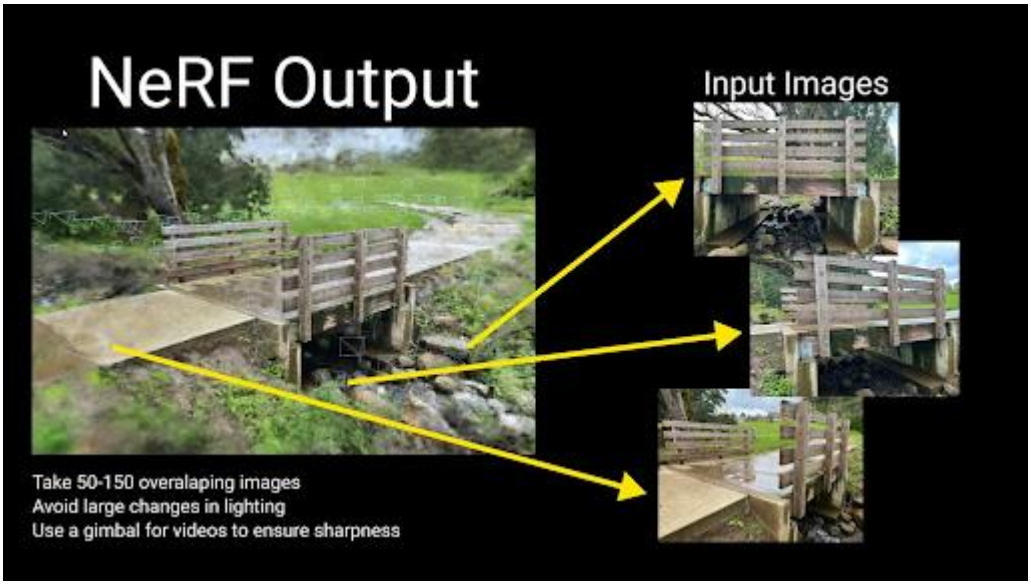


Рисунок 1.4 - Приклад роботи NVIDIA Instant NeRF

- **COLMAP**: це універсальний програмний інструмент, призначений для створення 3D-моделей на основі набору 2D-зображень. Він реалізує повний пайплайн структури з руху (SfM) та стереозріння з кількох камер (MVS). COLMAP використовується в NVIDIA Instant NeRFs для визначення позицій камер на основі вхідних зображень.
    - **Edge AI**

Розгортання моделей комп'ютерного зору на периферійних пристроях зменшує затримки та покращує продуктивність, дозволяючи обробку даних безпосередньо на пристрої.

Приклади та використання:

- **NVIDIA Jetson**: це сімейство модулів, призначених для прискорення ШІ на периферійних пристроях. Вони пропонують високу продуктивність ШІ в енергоефективному та компактному форматі. Разом із NVIDIA

JetPack SDK ці модулі дозволяють розробляти та впроваджувати інноваційні продукти в різних галузях (див. рис. 1.5).

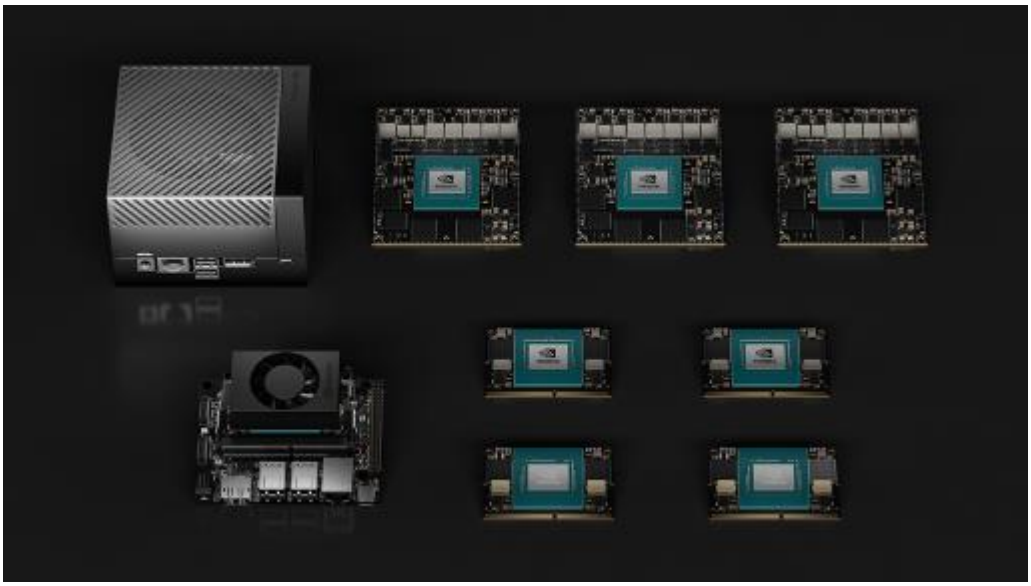

Рисунок 1.5 - Сімейство модулів NVIDIA® Jetson™

- **Google Coral**: це платформа для локального ШІ, яка пропонує повний набір інструментів для розробки продуктів з використанням локального ШІ. Її можливості висновку на пристрої дозволяють створювати продукти, які є ефективними, приватними, швидкими та працюють в автономному режимі.

- **Самостійне навчання**

Самостійне навчання (Self-supervised learning) зменшує потребу у великих позначених наборах даних, дозволяючи моделям навчатися на непозначених або мало позначених даних.

Приклади та використання:

- **SimCLR**: це фреймворк, представлений у статті "Простий фреймворк для контрастного навчання візуальних представлень". Він спрощує алгоритми самоконтрольованого навчання, запропоновані раніше, без потреби у спеціалізованих архітектурах або банках пам'яті. Основна ідея SimCLR полягає в навчанні моделі розрізняти позитивні пари (два різні доповнені перегляди одного зображення) від негативних пар (доповнені

перегляди різних зображень). Це робиться шляхом мінімізації контрастної втрати, яка змушує представлення позитивних пар бути ближчими один до одного, ніж до представлень негативних пар. Ключовими особливостями SimCLR є: простота - SimCLR не потребує спеціалізованих архітектур або банків пам'яті, що робить його легким для реалізації та масштабування; ефективність - SimCLR досяг вражаючих результатів в задачах самоконтрольованого та напівконтрольованого навчання на ImageNet. Основними висновками дослідження є наступні твердження: композиція доповнень даних відіграє ключову роль у визначенні ефективних задач прогнозування; введення нелінійного перетворення між представленням та контрастною втратою суттєво покращує якість отриманих представлень; контрастне навчання виграє від більших розмірів пакетів та більшої кількості кроків навчання порівняно з контрольованим навчанням [3].

- **BYOL (Bootstrap Your Own Latent)**: це новий метод самоконтрольованого навчання представлення зображень, представлений у статті "Bootstrap your own latent: A new approach to self-supervised Learning" BYOL працює з двома нейронними мережами: онлайн-мережею, яка навчається та цільовою мережею, яка служить своєрідним "вчителем".

  Основна ідея BYOL полягає в наступному: Онлайн-мережа отримує на вхід зображення, яке було модифіковано за допомогою аугментації даних. Онлайн-мережа навчається передбачати представлення цього ж зображення, але модифікованого іншим способом, які були отримані цільовою мережею. Цільова мережа оновлюється за допомогою повільно рухомого середнього значення ваг онлайн-мережі.

  Ключові особливості BYOL є: відсутність негативних пар - На відміну від багатьох інших методів самоконтрольованого навчання, BYOL не вимагає використання негативних пар, що спрощує процес навчання та дозволяє уникнути деяких проблем, пов'язаних з їх вибором; та висока

ефективність: BYOL демонструє результати, які перевершують або, принаймні, не поступаються найсучаснішим методам у задачах переносу навчання та напівконтрольованого навчання. Наприклад, BYOL досягає точності 74,3% для top-1 на ImageNet при лінійній оцінці з архітектурою ResNet-50 та 79,6% з більшим ResNet [4] (див. рис. 1.6).

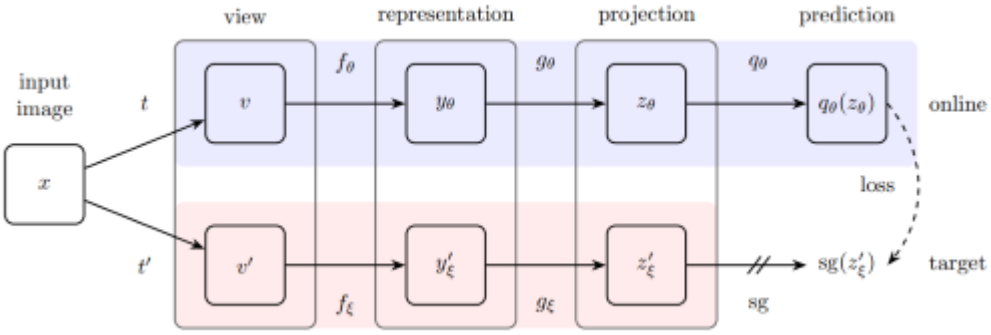


Рисунок 1.6 - Архітектура BYOL

### 1.2 Актуальність теми, основна ідея та мета кваліфікаційної роботи

Актуальність дослідження та розробки програмного модуля керування ігровими стратегіями обумовлена кількома ключовими факторами:

Розвиток індустрії відеоігор: Сучасні ігри стають все більш складними та реалістичними, вимагаючи просунутих алгоритмів для керування персонажами та середовищем. Програмні модулі, що забезпечують керування ігровими стратегіями, дозволяють розробникам створювати більш захоплюючий та інтерактивний геймплей.

Штучний інтелект та машинне навчання: Інтеграція AI в ігри покращує можливості NPC (неігрових персонажів) адаптуватися до дій гравця. Це підвищує реалістичність гри та забезпечує унікальний досвід для кожного користувача.

Крос-галузеві застосування: Технології, розроблені для ігрових стратегій, можуть бути застосовані в інших сферах, таких як симуляції, навчання, військова підготовка та бізнес-моделювання. Це розширює сферу впливу розробок та підвищує їх комерційну цінність.

Конкурентоспроможність на ринку: Компанії, які впроваджують інноваційні рішення в своїх іграх, мають перевагу над конкурентами. Розробка унікальних модулів керування стратегіями може стати визначальним фактором успіху продукту.

Покращення користувацького досвіду: Інтелектуальні ігрові стратегії підвищують задоволеність гравців, утримують їх увагу та стимулюють повторні сеанси гри, що важливо для комерційного успіху проекту.

### 1.3 Перелік задач кваліфікаційної роботи

Ціллю роботи є дослідження технологій машинного зору і розробки програмного модулю на їх основі, що значно полегшить процес стратегічного планування перед атаками у мобільній грі Clash of Clans у вигляді чат-боту для платформи обміну миттєвими повідомленнями та цифрового розповсюдження інформації з функціями VoIP, а саме Discord.

Для досягнення мети роботи визначені наступні задачі:

- Виконати огляд актуальних рішень у сфері комп’ютерного зору;
- Провести критичний аналіз існуючих аналогів та виявити їх недоліки;
- Сформулювати сценарії взаємодії користувача з чат-ботом;
- Змоделювати алгоритм, функціональні схеми та базу даних для програмного модулю;
- Навчити систему комп’ютерного зору розпізнавати потрібні об’єкти, оброблювати їх та інтерпретувати у потрібному вигляді;
- Розробити інтерфейс програмного модулю;
- Реалізувати функціональну складову програмного модулю;
- Протестувати правильність роботи чат-боту.

Тобто, в результаті кваліфікаційної роботи має бути реалізовано програмний модуль, що буде значно полегшувати та прискорювати процес планування атак та керування кланом для гравця Clash of Clans.

## 1.4 Аналіз існуючих аналогів у професійній сфері з виявленням їх недоліків

Серед аналогів розроблюваного програмного модулю можна визначити два ресурси, це:

- https://www.burntbase.com/;
- https://findthisbase.com/search.

А тепер більш детально про кожен з додатків, які можна знайти у Discord App Directory, скоріш за все Ви спитаєте що таке Discord App Directory і що з ним робити? Так от, Discord App Directory дозволяє вам шукати, переглядати і вивчати тисячі програм для налаштування вашого сервера безпосередньо у Discord! За допомогою програм ви можете налаштувати і персоналізувати свій сервер, автоматизувати корисні завдання, щоб звільнити час, а також додати веселощів і захоплення у вашу спільноту. Нові програми додаються регулярно, оскільки розробники дозволяють відкривати свої програми, тож слідкуйте за тим, щоб не пропустити наступну круту програму, яка покращить ваш сервер [5].

Discord-бот BurntBase: Твій помічник у Clash of Clans

**BurntBase** - це спеціальний бот для платформи Discord, створений спеціально для гравців популярної мобільної стратегії Clash of Clans. Його основна функція - допомагати гравцям захищати свої бази та покращувати свої атакуючі стратегії [6].

Як працює BurntBase?

Аналіз баз: Бот використовує технологію розпізнавання зображень, щоб проаналізувати скріншоти ваших баз або баз ваших супротивників.

Пошук відео: Після аналізу, бот шукає у своїй базі даних відео на YouTube, де ці бази вже були атаковані та отримали три зірки.

Надання результатів: Якщо такі відео знаходяться, бот надає вам посилання на них, щоб ви могли вивчити стратегії атаки та з'ясувати, як зміцнити свою оборону.

Чому BurntBase корисний?

Економія часу: Замість того, щоб годинами шукати відео на YouTube, ви можете отримати всю необхідну інформацію за лічені секунди.

Покращення оборони: Вивчаючи невдалі атаки на схожі бази, ви можете виявити слабкі місця своєї оборони та зміцнити їх.

Розробка ефективних атак: Аналізуючи успішні атаки на бази супротивників, ви можете розробити власні ефективні стратегії атаки (див. рис. 1.7).

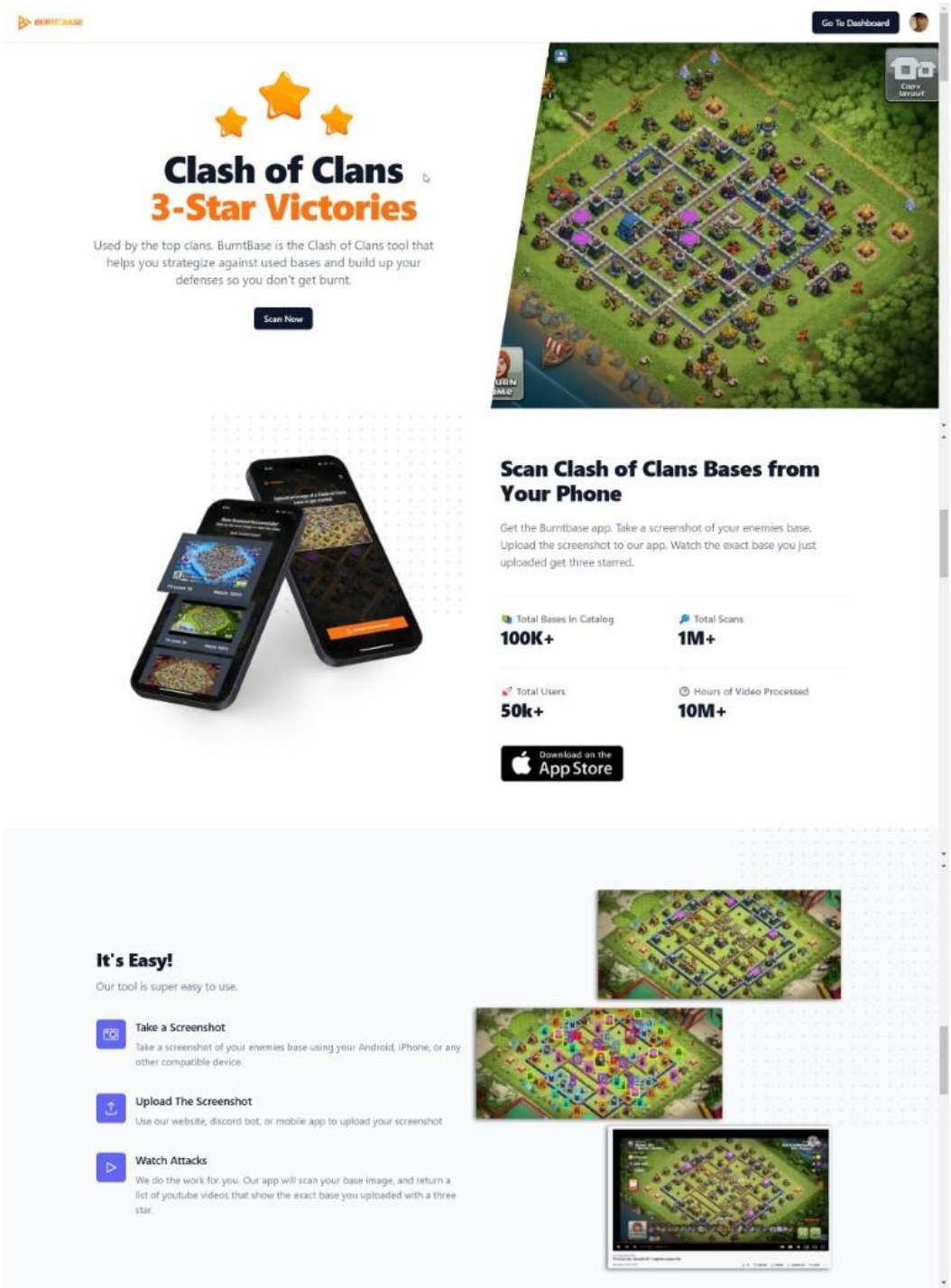


Рисунок 1.7 - Головна сторінка сайту https://www.burntbase.com/

Основні функції BurntBase:

Швидке сканування скріншотів баз: Завдяки великій базі даних, бот знаходить відповідні відео майже миттєво (див. рис. 1.8).

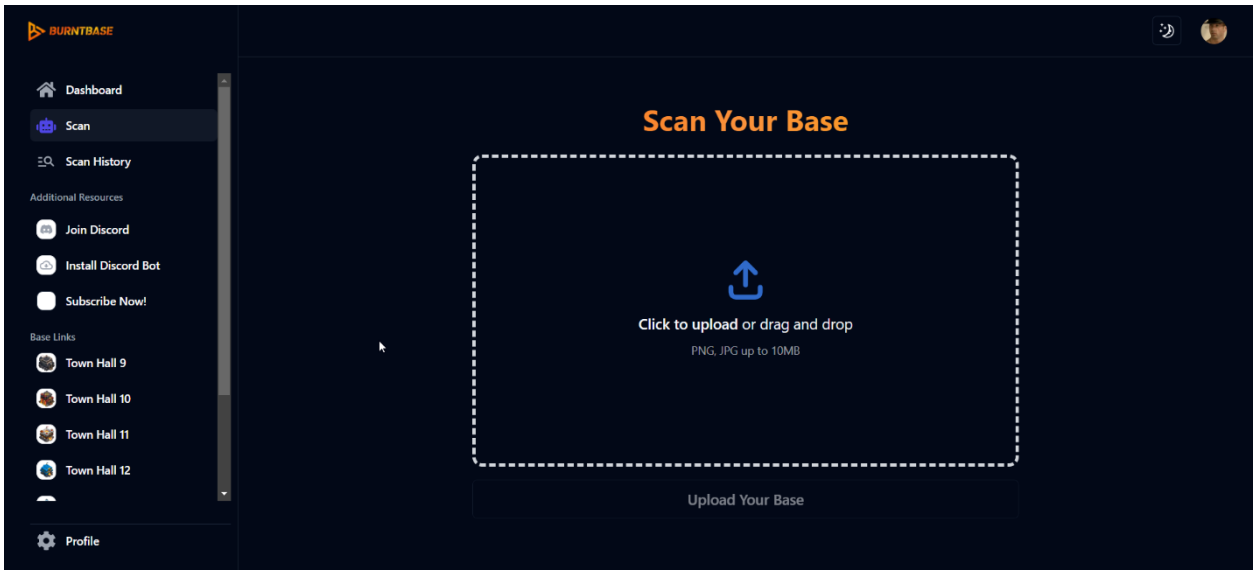


Рисунок 1.8 - Сторінка сканування сайту https://www.burntbase.com/scan

Інтуїтивний інтерфейс: Сайт має водночас стильний і зрозумілий інтерфейс, що дозволяє користуватися ним навіть новачкам (див. рис. 1.9).

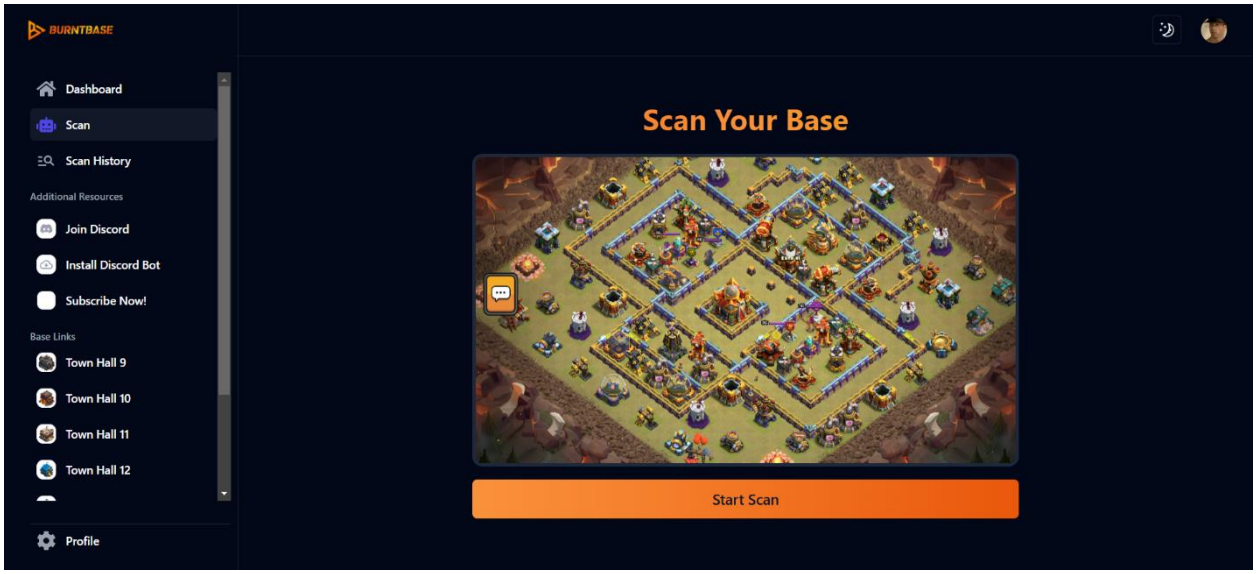


Рисунок 1.9 - Процес сканування бази на сайті https://www.burntbase.com/scan

Зручна та інформативна сторінка результатів пошуку (див. рис. 1.10).

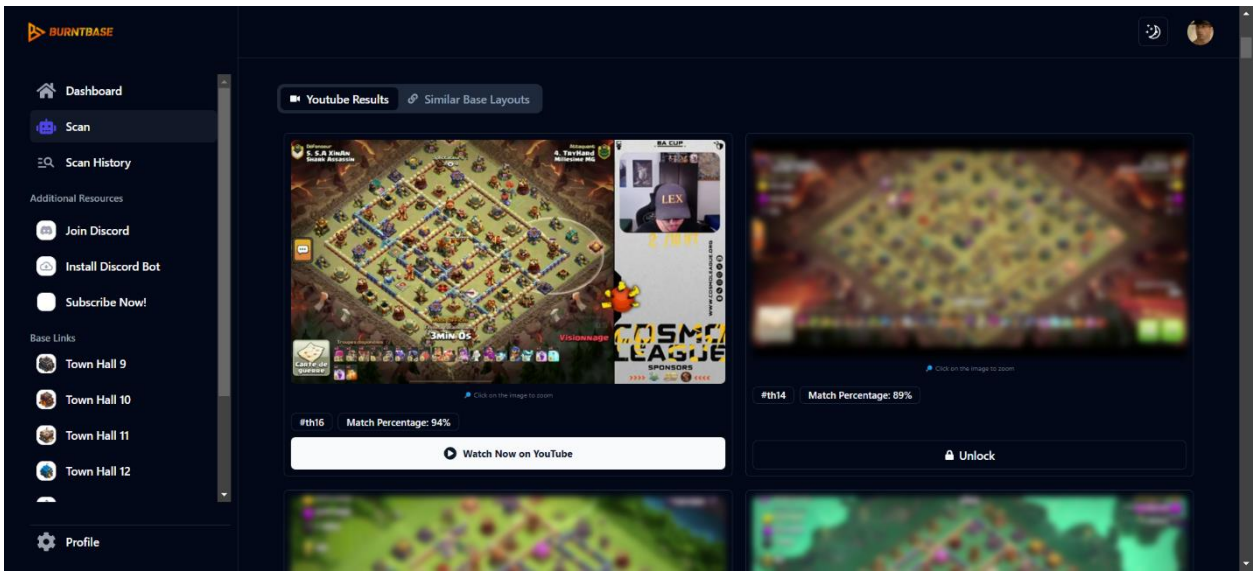


Рисунок 1.10 - Результат сканування бази на сайті https://www.burntbase.com/scan

Історія попередніх сканувань (див. рис. 1.11).


Рисунок 1.11 - Історія попередніх сканувань на сайті https://www.burntbase.com/history

Регулярні оновлення: База даних бота постійно оновлюється, щоб ви завжди мали доступ до найактуальнішої інформації після внутрішньоігрових оновлень контенту, як от нещодавнє оновлення від 25 листопада, що

вважається найбільшим оновленням Clash of Clans цього року – «Ратуша 17 та купа інших нових доповнень вже тут... почнемо!» [7]

Підтримка спільноти: BurntBase має дуже велику та досить активну спільноту Discord-користувачів, що налічує 19500 користувачів, де ви можете обмінюватися досвідом та отримати допомогу (див. рис. 1.12) [8].

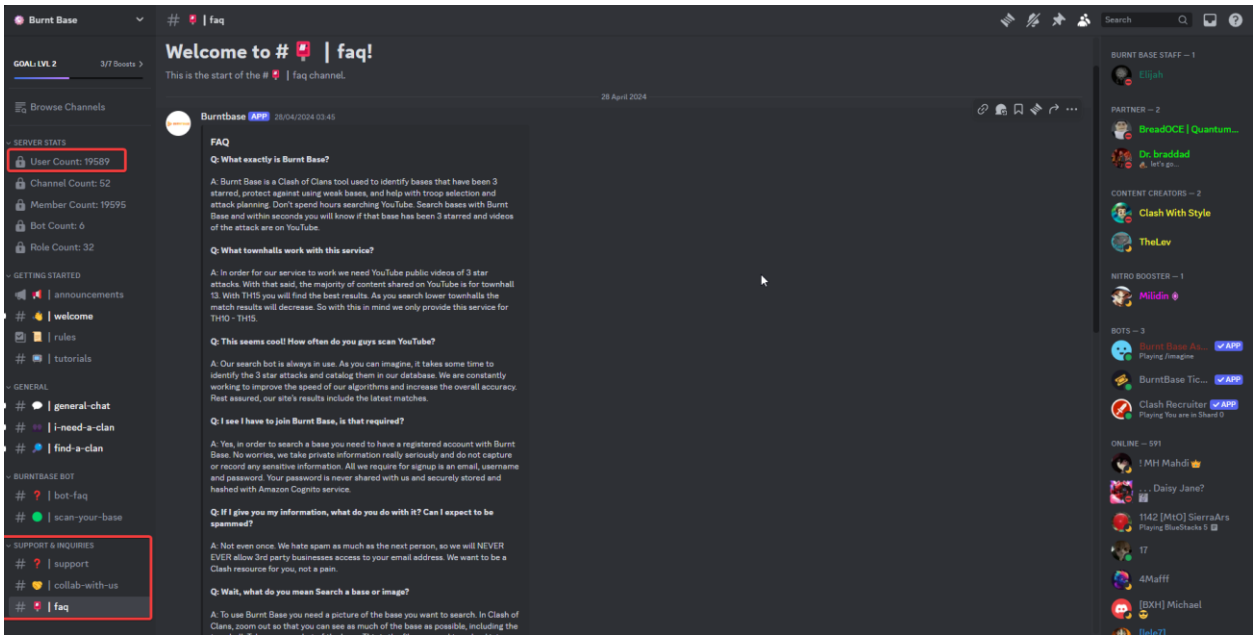


Рисунок 1.12 - Розділ ЧаПи у Discord-спільності Burnt Base

Наявність партнерської програми: якщо ви контент-кріейтор, або ж просто людина, яка має авдиторію, якій можуть бути цікаві платні можливості на BurntBase, то ви можете на цьому непогано так заробити (див. рис. 1.13)[9].


Рисунок 1.13 - Умови партнерської програми BurntBase Affiliate

Якщо Ви граєте в Clash of Clans і хочете підняти свою гру на новий рівень, то Discord-бот BurntBase - це те, що вам потрібно!

А тепер перейдемо до другого аналогу, Discord-боту Find This Base, твого персонального розвідника у Clash of Clans.

**Find This Base** – це потужний інструмент для гравців Clash of Clans, створений спеціально для того, щоб полегшити пошук ефективних стратегій атаки. Цей бот працює на платформі Discord і пропонує ряд корисних функцій, які допоможуть тобі стати кращим гравцем у Clash of Clans [10].

Як працює Find This Base?

Завантаження зображення бази: Ти завантажуєш скріншот бази, яку хочеш атакувати або захистити на головній сторінці сайту (див. рис. 1.14).

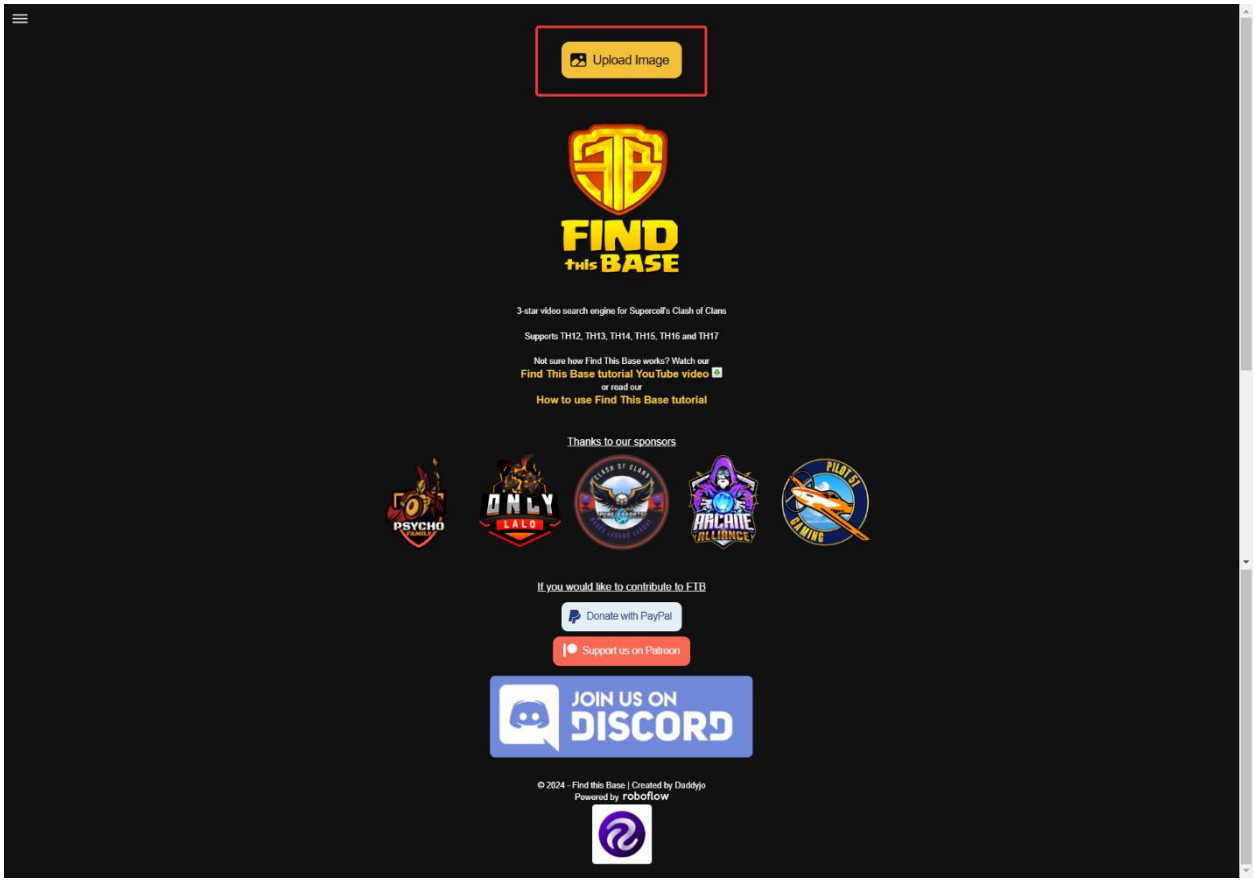


Рисунок 1.14 - Головна сторінка сайту https://findthisbase.com/search

Пошук відео: Бот аналізує завантажене зображення і шукає у своїй великій базі даних відео на YouTube, де були успішно атаковані схожі бази.

Зазвичай цей процес не займає багато часу і супроводжується повідомленнями статусу (див. рис. 1.15):

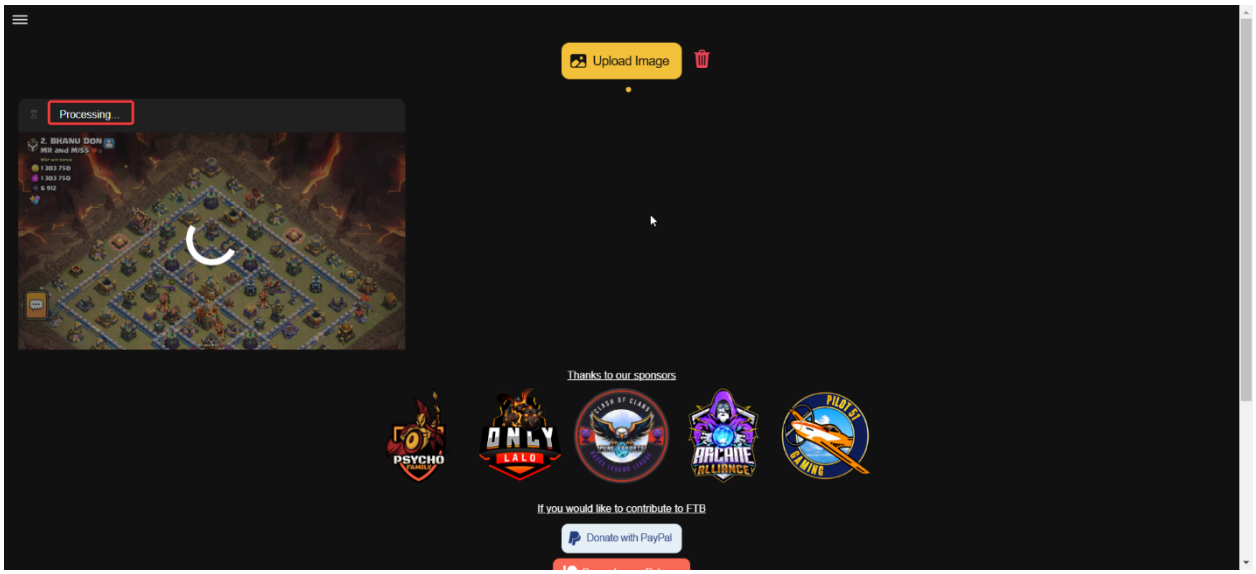


Рисунок 1.15 - Процес обробки зображення бази на сайті https://findthisbase.com/search

Надання результатів: Після пошуку бот надає тобі посилання на відповідні відео, де ти можеш детально вивчити стратегії атаки, які можуть бути ефективними проти твоєї цілі, але іноді трапляються ситуації, коли результатів пошуку немає (див. рис. 1.16), але зазвичай отримуємо досить розгорнуті і релевантні результати пошуку (див. рис. 1.17):

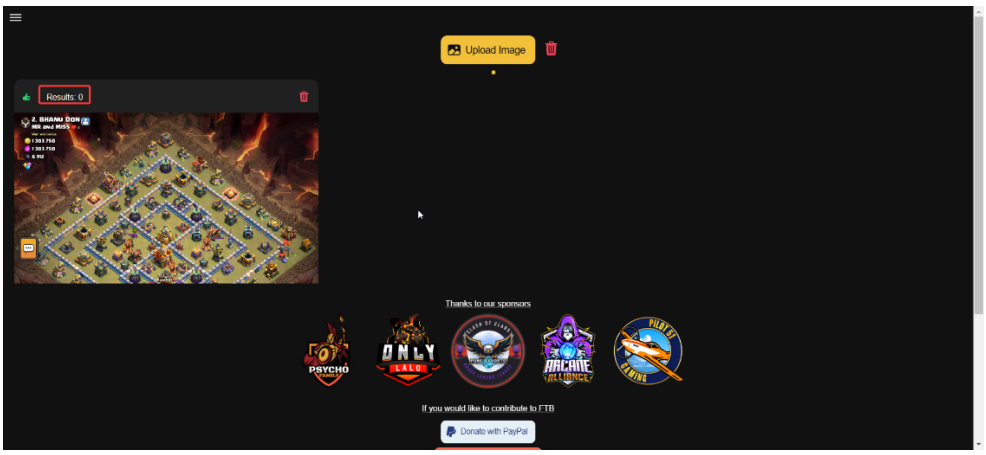


Рисунок 1.16 - Випадок безрезультатного пошуку бази на сайті https://findthisbase.com/search

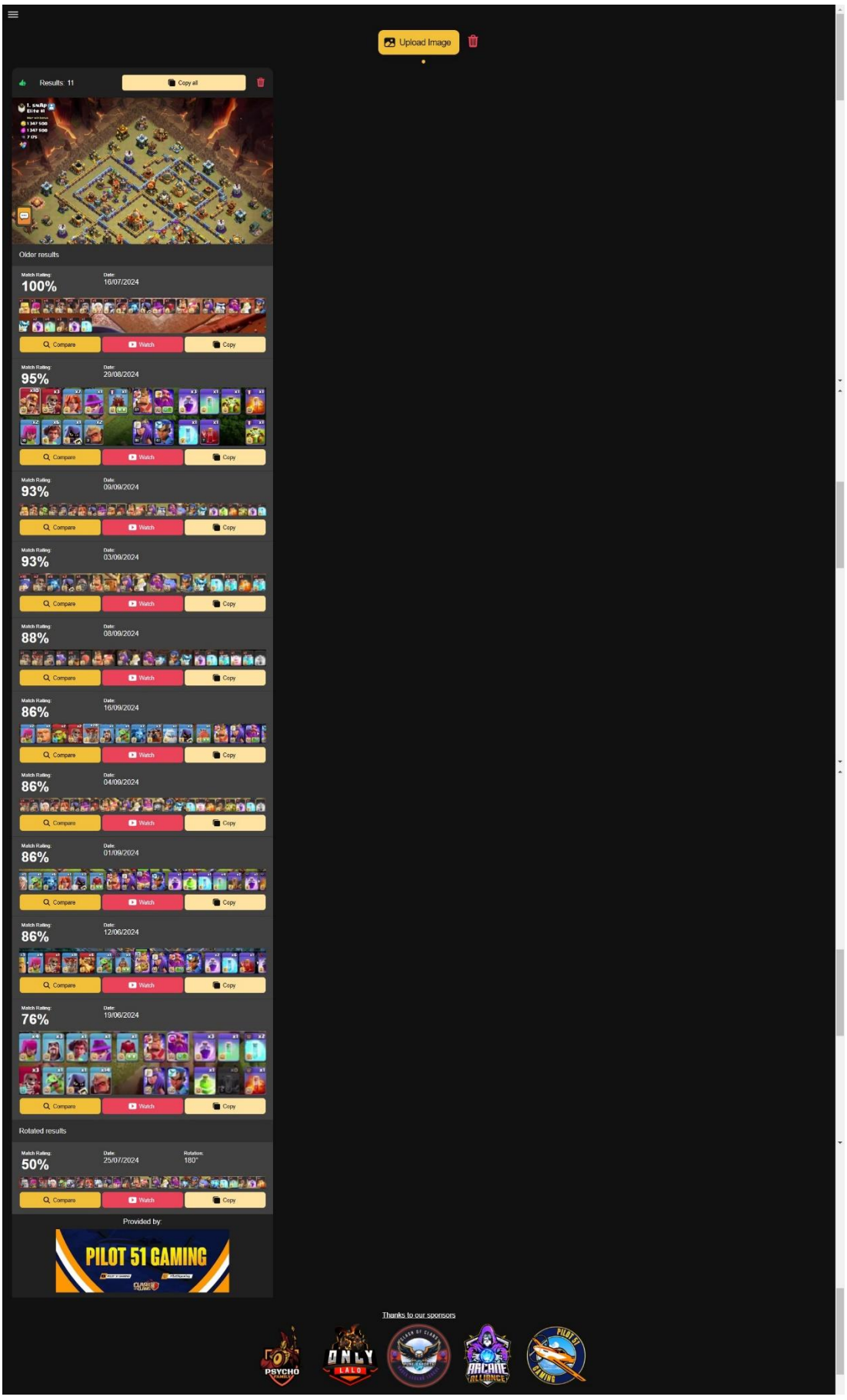


Рисунок 1.17 - Результат пошуку бази на сайті https://findthisbase.com/search

Під кожним результатом пошуку Find This Base дає ще 3 кнопки: «Compare», «Watch» та «Copy». Натиснувши кнопку «Compare» можна переконатися в тому, що бот знайшов вдалу спробу атаки саме на ту базу, що зображена на скріншоті (див. рис. 1.18):

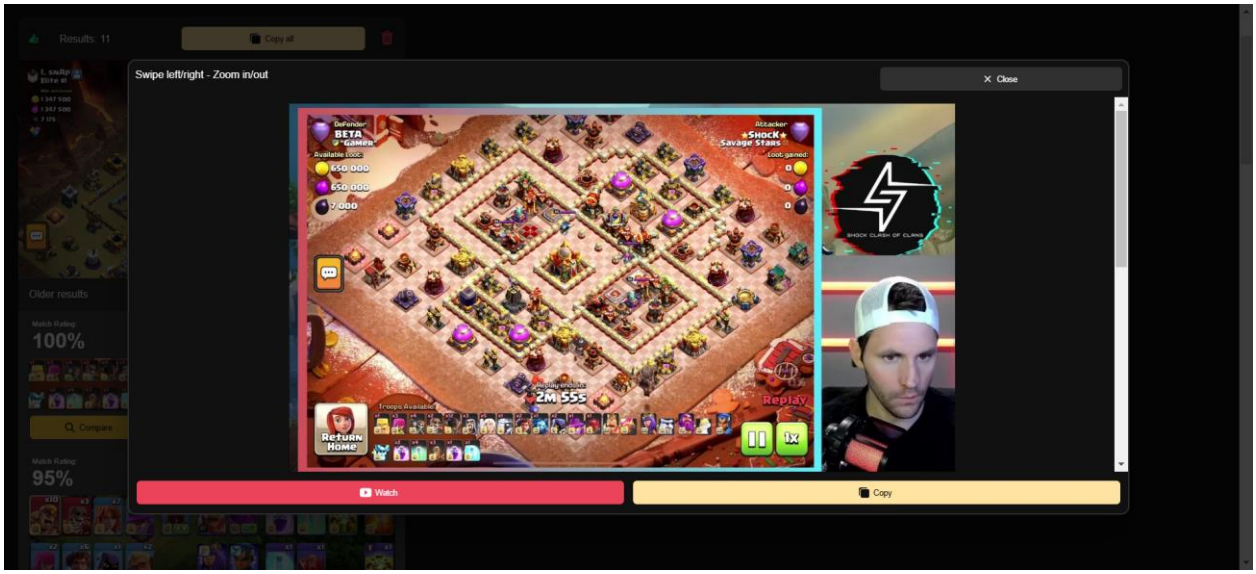


Рисунок 1.18 - Вікно порівняння баз в результатах пошуку на сайті https://findthisbase.com/search

Натиснувши кнопку «Watch» у новому вікні браузеру відкривається відео на YouTube з відповідним тайм-кодом, на якому починається атака на ту базу, що зображена на скріншоті (див. рис. 1.19), а натиснувши кнопку «Copy» в буфер обміну скопіюється інформація наступного формату:

«Match Rating: 100%

Date: 16/07/2024

Used army:

https://findthisbasestorage.blob.core.windows.net/ftb2test/20240716113940244_142_UsedArmy.jpg

Video: https://www.youtube.com/watch?v=OI7Bhwm-s98&t=764s», на випадок, якщо результатами пошуку потрібно поділитися з другом, або іншим членом клану.

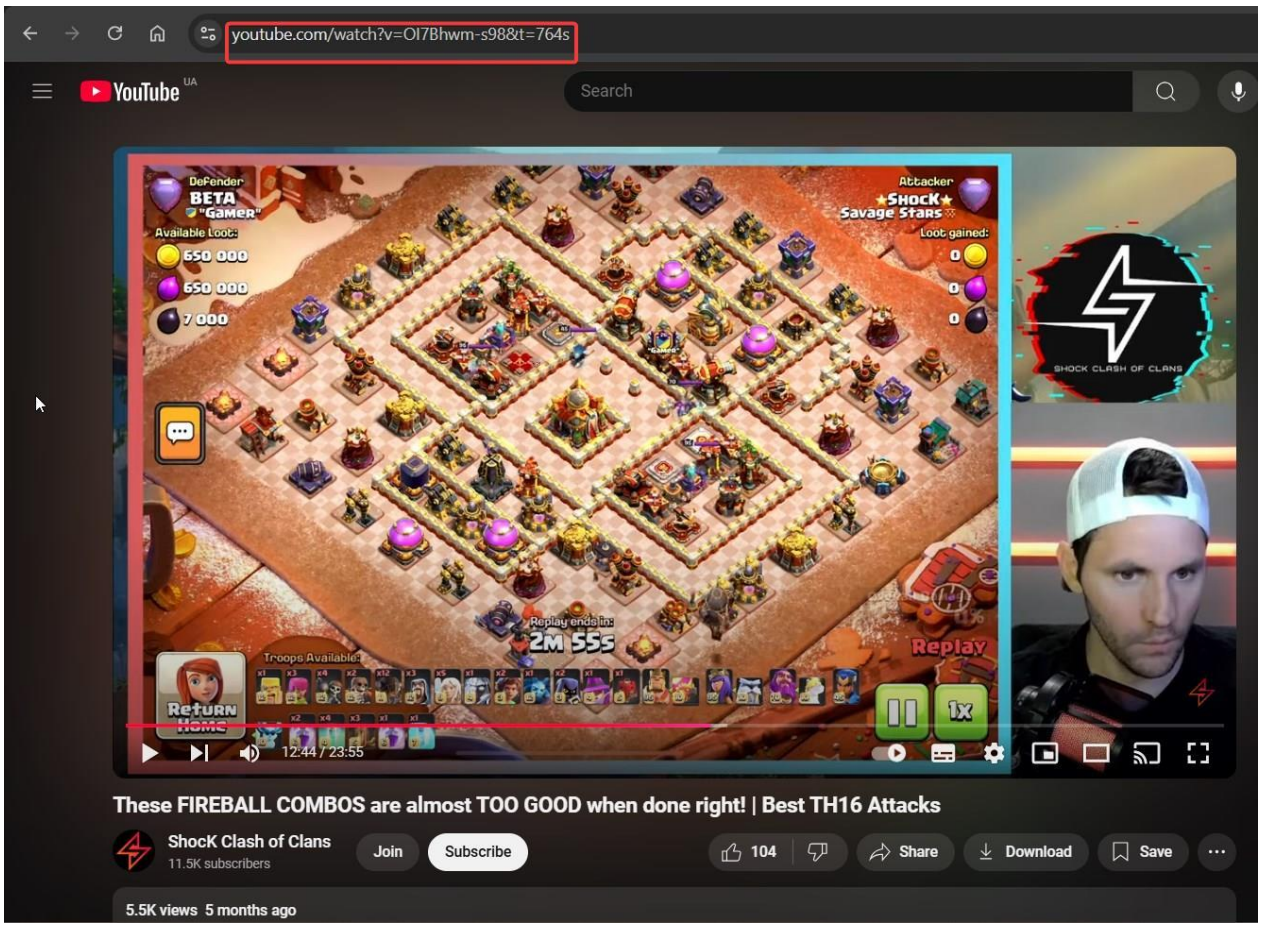


Рисунок 1.19 - Результат натискання кнопки «Watch» у результатах пошуку на сайті https://findthisbase.com/search

Чому Find This Base корисний?

Економія часу: Замість того, щоб годинами переглядати безліч відео на YouTube, ти отримуєш підібрані варіанти за лічені секунди.

Покращення стратегій атаки: Вивчаючи успішні атаки інших гравців, ти можеш розробити власні ефективні стратегії і збільшити свої шанси на перемогу.

Аналіз вразливостей бази: Переглядаючи невдалі атаки на схожі бази, ти можеш виявити слабкі місця своєї оборони і зміцнити їх.

Основні функції Find This Base:

Швидкий пошук відео: Бот має велику базу даних і працює дуже швидко.

Інтуїтивний інтерфейс: Навіть новачок зможе легко розібратися в роботі бота.

Регулярні оновлення: База даних бота постійно доповнюється новими відео, щоб ти завжди мав доступ до актуальної інформації.

Підтримка різних типів баз: Бот працює з різними типами баз, включаючи бази для фарму, для кланових війн та гібридні.

Якщо ти хочеш підняти свою гру в Clash of Clans на новий рівень і стати більш успішним атакувальником, то Discord-бот Find This Base – це незамінний інструмент для тебе!

**Недоліки ботів для Clash of Clans: Find This Base та BurntBase**

Хоча боти для Clash of Clans, такі як Find This Base та BurntBase, є цінними інструментами для гравців, вони мають свої обмеження та недоліки. Ось деякі з них, до загальних можна віднести наступне:

Залежність від якості зображення: Ефективність роботи бота значною мірою залежить від якості завантаженого зображення бази. Розмиті, нечіткі або неповні зображення можуть призвести до неправильної ідентифікації бази та неточних результатів пошуку.

Обмежена база даних: Незважаючи на великі бази даних, боти можуть не містити відео для всіх можливих конфігурацій баз. Це особливо стосується нестандартних або недавно розроблених баз.

Зміни в грі: З кожним оновленням Clash of Clans змінюються баланс військ, будівель та стратегії атаки. Боти можуть не встигати за цими змінами, що робить деякі відео менш актуальними.

Неможливість передбачити нестандартні стратегії: Боти аналізують вже існуючі відео, тому вони можуть не знайти ефективних стратегій для нестандартних або незвичайних баз.

Вартість використання ботів: обидва ресурси можна використовувати безкоштовно, але в обмеженому форматі. На сайті BurntBase, наприклад, дає при скануванні лише переглянути перший результат пошуку без підписки (див. рис. 1.20), та не дає доступу до історії сканувань (див. рис. 1.21).

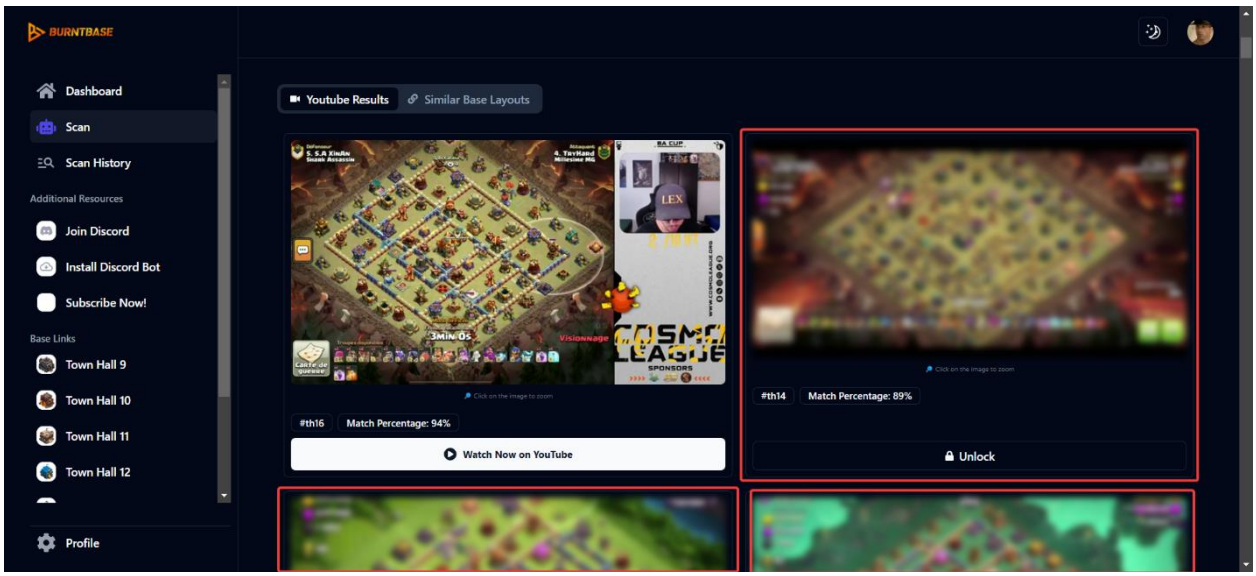


Рисунок 1.20 - Обмежений безкоштовною версією результат пошуку на сайті https://www.burntbase.com/scan


Рисунок 1.21 – Неможливість переглянути історію сканів з безкоштовною версією на сайті https://www.burntbase.com/history

При цьому підписка на BurntBase коштує досить немало, в особливості для нашого регіону, а саме 9.99$ на місяць (див. рис. 1.22). За цю суму користувач отримує безлімітний доступ до усього функціоналу ресурсу та можливість користуватися мобільними застосунками на iOS та Android (див. рис. 1.23).

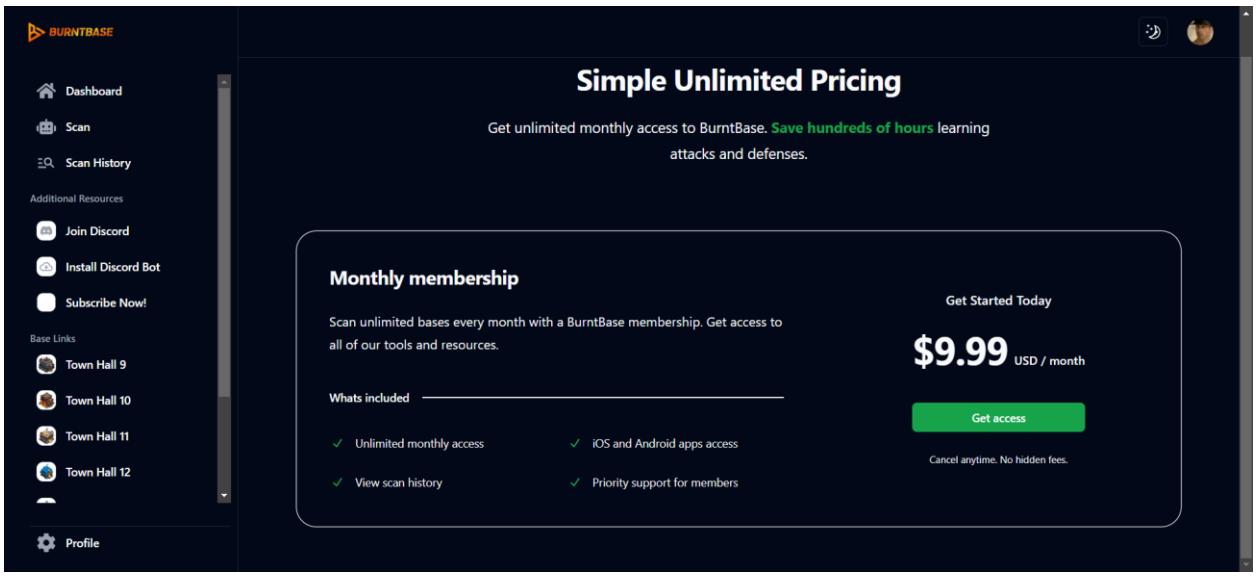


Рисунок 1.22 - Повна інформація про платну підписку на сайті https://www.burntbase.com/

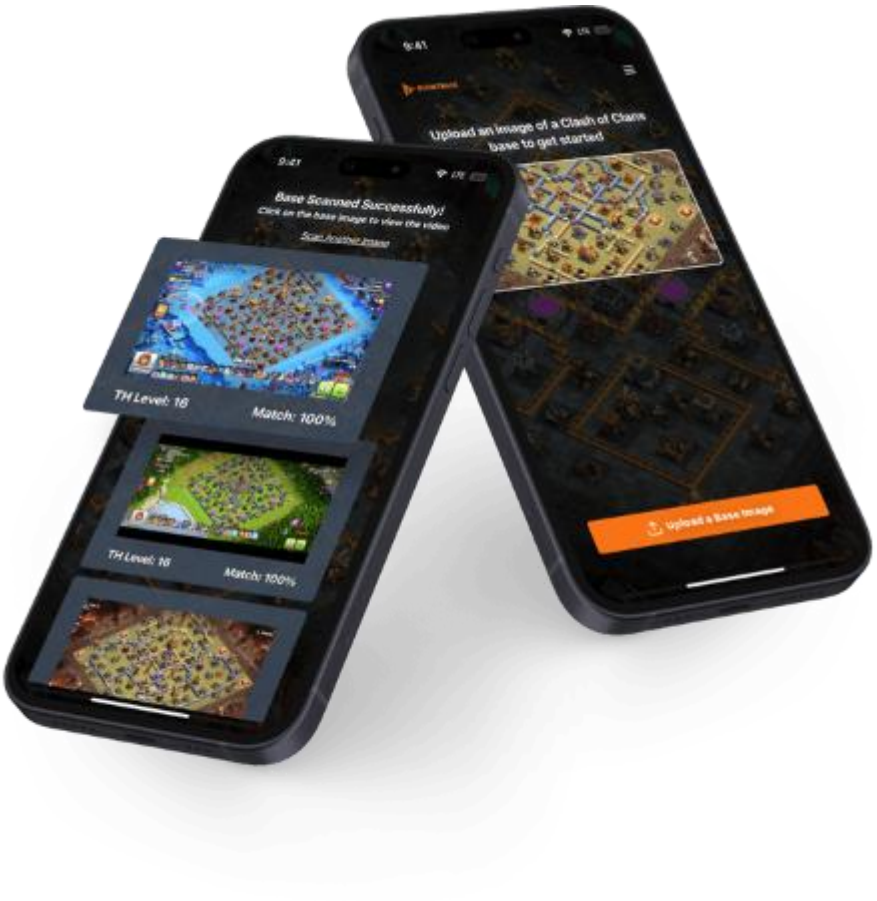


Рисунок 1.23 - Демонстрація мобільного застосунку BurntBase

Хоча в той самий час, на Discord-сервері BurntBase можна користуватися функцією сканування (див. рис. 1.24) і отримувати повні результати, але в трохи іншому, властивому для Discord форматі (див. рис. 1.25).

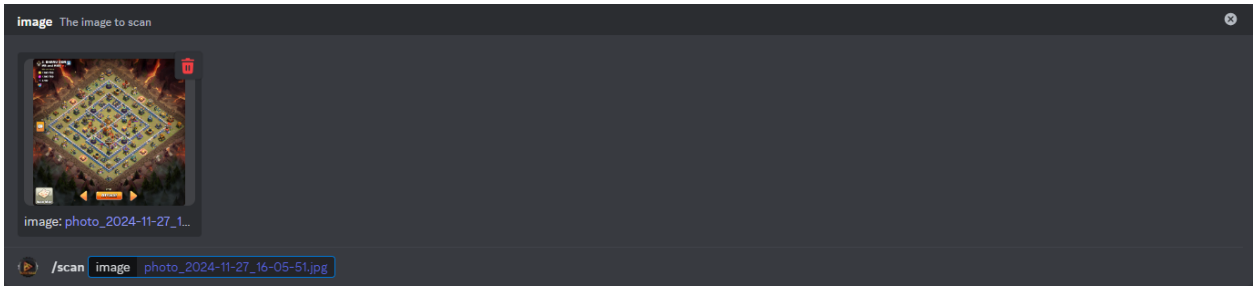


Рисунок 1.24 - Процес запиту на сканування бази за допомогою Discord-бота BurntBase


Рисунок 1.25 - Результати сканування бази за допомогою Discord-бота BurntBase

Щодо платної складової Find This Base має трохи інший підхід: Ви можете користуватися безкоштовно сайтом цього ресурсу і ботом на офіційному Discord-сервері Find This Base [11], але якщо Ви хочете додати Find This Base бота на свій Discord-сервер, наприклад це сервер Вашого клану, то Вам потрібно буде купувати Find This Base Scantokens у офіційному Discord-сервері Find This Base (див. рис 1.26), які будуть дійсні протягом 30 днів (див. рис. 1.27). Якщо після закінчення цих 30 днів на сервері залишаться токени, їх можна активувати за допомогою нової покупки. Новопридбані токени будуть додані до загальної кількості токенів на сервері[12] (див. рис. 1.28).

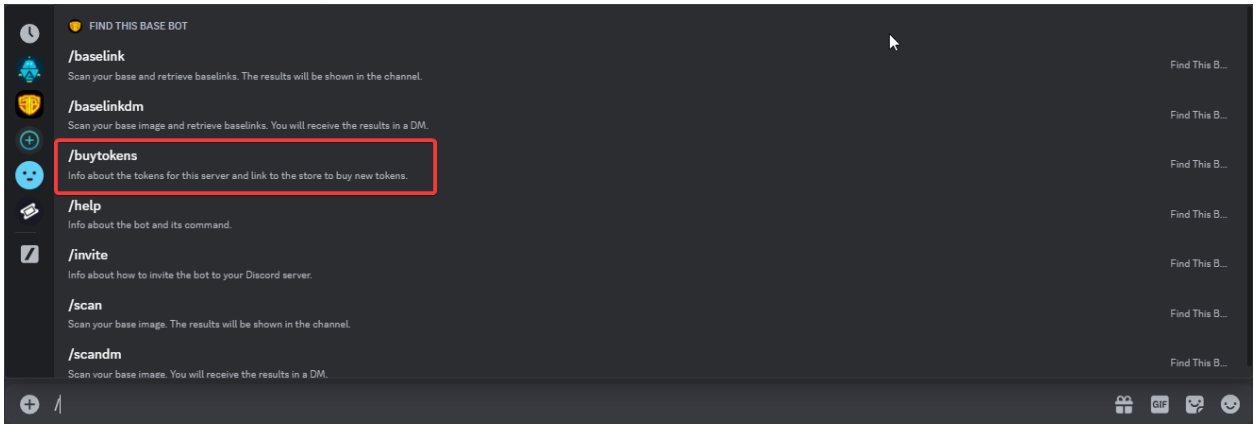


Рисунок 1.26 - Спосіб купівлі Find This Base Scantokens у офіційному Discord-сервері Find This Base

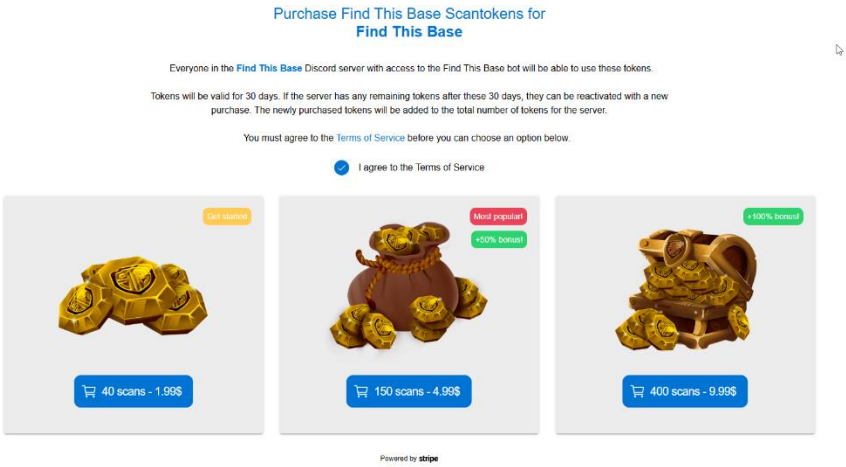


Рисунок 1.27 - Ціни та опції купівлі Find This Base Scantokens на сайті https://findthisbase.com/

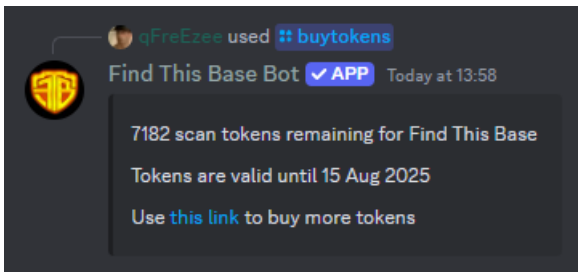


Рисунок 1.28 - Залишок Find This Base Scantokens і дата їх доступності у бота Find This Base Bot

До специфічних недоліків для кожного бота можна віднести:

Find This Base:

Може бути менш ефективним для пошуку стратегій для захисту бази, оскільки основний акцент робиться на атакуючих стратегіях.

Може мати обмеження щодо розміру та складності баз, які може проаналізувати.

BurntBase:

Може бути більш спеціалізований на пошуку відео для певних типів баз, що обмежує його універсальність, іноді може видавати взагалі неправильні результати при пошуку бази, наприклад при надсиланні скріншоту з рівнем ратуши 16 рівня через сайт – бот видавав єдиний результат і це була база з 13 рівнем ратуші, що не має взагалі нічого спільного з початковим запитом.

Може мати меншу базу даних порівняно з іншими ботами.

Додаткові міркування:

Етичні аспекти: Використання ботів може порушити баланс між гравцями, надаючи деяким гравцям несправедливу перевагу. Але це не суперечить Політиці щодо Фан-Контенту від Supercell [13]

Залежність від технологій: Робота ботів залежить від стабільності серверів та алгоритмів розпізнавання зображень.

Витрати часу: Навіть з використанням ботів, гравці все одно повинні витрачати час на аналіз запропонованих відео та розробку власних стратегій.

Висновок:

Боти для Clash of Clans можуть бути корисними інструментами для навчання та покращення своїх навичок, але вони не є панацеєю. Важливо розуміти їхні обмеження та використовувати їх як допоміжний інструмент, а не як основне джерело інформації.

## 1.5 Аналітичний огляд бізнес-процесів професійної області і виявлення проблеми

1. Аналіз ігрової індустрії та стратегії в контексті Clash of Clans

Clash of Clans є однією з найпопулярніших мобільних стратегічних ігор у жанрі MMORTS (масова багатокористувацька онлайн стратегія в реальному часі). Гра передбачає:

Побудову бази: Гравці розміщують будівлі, укріплення й пастки для захисту ресурсів.

Атаку на інші бази: Гравець планує стратегію атаки, вибирає війська і тактично використовує їх для захоплення ресурсів чи перемоги.

Глобальна спільнота гри складається з мільйонів користувачів, що створює складне середовище для пошуку ефективних стратегій атак. Професійна область, пов’язана з аналітикою ігрових стратегій, включає такі бізнес-процеси:

Створення гайдів і навчального контенту: Блогери, стримери та YouTube-креатори створюють відеоконтент із демонстрацією успішних атак.

Аналіз розташування баз: Професійні гравці та аналітики вручну аналізують ідентичні або схожі бази для вибору оптимальних стратегій атак.

Поширення стратегій: Стратегії поширюються на платформах YouTube, Discord, форумах та в ігрових спільнотах.

Розглянемо детальніше бізнес-процеси та проблеми, які виникають у цій професійній області:

Таблиця 1.1 - Огляд ключових бізнес-процесів та проблеми

| Бізнес-процес | Опис | Проблеми |
|---|---|---|
| Пошук стратегій атак | Гравці шукають відео або статті, де показано успішні атаки на схожі бази. | - Трудомісткий процес, що займає багато часу.<br>- Відсутність автоматизованих інструментів. |
| Створення навчальних матеріалів | Блогери створюють відео для демонстрації атак. | - Не всі гравці мають час для вивчення відео.<br>- Важко знайти саме релевантну інформацію. |
| Розпізнавання розташування будівель | Аналітики вручну визначають ключові особливості бази. | - Відсутність автоматизованого аналізу.<br>- Візуальний аналіз потребує великого досвіду. |
| Поширення стратегій у Discord | Спільноти використовують Discord для обговорення ігор та стратегій. | - Discord-боти для аналізу баз не розвинені.<br>- Відсутність інструментів для інтеграції аналітики. |

3. Виявлення проблеми

На основі аналізу можна виділити основні проблеми в професійній області, які вирішує розробка твого програмного модуля:

Відсутність автоматизованого інструменту для пошуку стратегій атак

Гравці витрачають значний час на пошук релевантних відео.

Процес часто є неефективним, оскільки вибір стратегії залежить від точності ідентифікації розташування будівель.

Недостатнє застосування технологій комп'ютерного зору

Існуючі рішення не використовують комп'ютерний зір для автоматичного розпізнавання об'єктів на скріншотах.

Ручний аналіз баз є повільним і схильним до помилок.

Відсутність інтеграції у середовищі Discord

Discord є популярною платформою для комунікації гравців, однак бракує ботів, що допомагають автоматизувати процес аналізу та пошуку стратегій атак.

Більшість Discord-ботів зосереджені лише на базових функціях, таких як статистика або команди для спілкування.

4. Актуальність та значення вирішення проблеми

Розробка Discord-бота, що використовує комп'ютерний зір для розпізнавання будівель і API YouTube для пошуку стратегій, є актуальним рішенням для таких завдань:

Автоматизація процесу аналізу баз: Швидке й точне розпізнавання об'єктів на скріншотах.

Ефективний пошук стратегій: Автоматичний підбір релевантних відео з YouTube.

Інтеграція зі спільнотами Discord: Наявність інструменту безпосередньо в середовищі, де спілкуються гравці, підвищує зручність і ефективність використання.

Таке рішення підвищить ефективність ігрових бізнес-процесів, заощадить час для гравців і надасть конкурентну перевагу в аналізі ігрових стратегій.

# 2 ВІДОМОСТІ ПРО ПРЕДМЕТ (ОБ’ЄКТ) ДОСЛІДЖЕННЯ

## 2.1 Формулювання предмету (об'єкту) дослідження

Об'єктом дослідження є процеси автоматизації аналізу ігрових стратегій у стратегічних онлайн-іграх на прикладі Clash of Clans. Об'єктом дослідження є автоматизовані підходи до аналізу візуальних даних (скріншотів), оптимізації пошуку ігрових стратегій та впровадження цих рішень у комунікаційних платформах для спільнот гравців.

Об'єктом дослідження виступають загальні процеси, що включають:

Застосування технологій комп'ютерного зору для аналізу зображень зі стратегічних ігор.

Використання платформ для пошуку контенту (YouTube) з метою надання рекомендацій.

Інтеграцію цих інструментів у популярні комунікаційні сервіси, такі як Discord, для зручного використання гравцями.

Таким чином, об'єкт дослідження окреслює широку сферу, пов’язану з автоматизацією аналітичних завдань у середовищі стратегічних ігор.

Предмет дослідження є методи, алгоритми та інструменти для розробки програмного модуля, що забезпечує:

Автоматичне розпізнавання будівель на скріншотах із гри Clash of Clans за допомогою технологій комп'ютерного зору (Convolutional Neural Networks, OpenCV тощо).

Автоматизований пошук відеоконтенту на платформі YouTube на основі ідентифікованих об'єктів і розташувань.

Інтеграцію модуля у Discord через створення бота, який надає користувачам релевантні відео та рекомендації щодо стратегій атак.

Предмет дослідження є конкретним і охоплює технічні рішення та методи, які використовуються для реалізації автоматизованого аналізу баз у Clash of Clans. До нього належать:

Алгоритми комп'ютерного зору: Технології розпізнавання об'єктів на зображеннях, що дозволяють ідентифікувати будівлі та їх розташування.

YouTube API: Інструменти для програмного пошуку відеоконтенту на основі певних запитів і параметрів.

Discord API: Засоби для створення ботів, які автоматизують взаємодію користувача із системою аналізу та рекомендацій.

Модульний підхід до розробки: Розбиття системи на окремі компоненти, що працюють у зв'язці для досягнення поставленої мети.

Таким чином, предмет дослідження є вузьконаправленим і зосереджується на технологічному вирішенні поставлених завдань.

Взаємозв’язок об'єкта та предмета дослідження

Об'єкт дослідження охоплює загальну проблему автоматизації аналітики в ігровій індустрії, тоді як предмет дослідження конкретизує рішення через розробку модуля, що використовує передові технології комп'ютерного зору, пошуку відеоконтенту та інтеграції в середовищі Discord.

## 2.2 Обґрунтування і вибір математичних методів дослідження поставлених задач

**Методи обробки та аналізу зображень**

Математичний апарат комп’ютерного зору

Для розпізнавання будівель на ігрових скріншотах використовуються методи машинного навчання, зокрема Convolutional Neural Networks (CNN).

Обґрунтування:

CNN є найбільш ефективним методом для задач класифікації та детекції об’єктів на зображеннях завдяки здатності до виділення ознак (features).

Кожен шар у мережі автоматично будує уявлення про зображення: від простих геометричних форм до складних об’єктів (у нашому випадку будівель).

Математична суть:

Операція згортки (convolution) та функція активації для навчання нейронної мережі.

Формула операції згортки для піксельного вікна K:

$$S(i,y) = (I * K)(i,y) = \sum_m \sum_n I(i-m, j-n)K(m,n),$$

де I — вихідне зображення, K — ядро згортки, S — результат згортки.

Алгоритми розпізнавання об'єктів

YOLO (You Only Look Once) або Faster R-CNN: ці алгоритми дозволяють виконати локалізацію та класифікацію будівель на зображеннях одночасно.

Обґрунтування:

YOLO працює швидко і дозволяє обробляти зображення в реальному часі, що є критичним для інтеграції у Discord-бот.

**Математичні методи для роботи з даними**

Алгоритми пошуку схожих об'єктів (Content-Based Search)

Методи порівняння векторів ознак (Feature Matching): Використовуються для зіставлення розпізнаних будівель на скріншоті з базою даних можливих розташувань.

Обґрунтування:

Для зіставлення об'єктів у різних зображеннях застосовуються евклідова відстань або косинусна подібність між векторами ознак.

**Методи інтеграції програмного модуля**

Модульний підхід та алгоритмічна структура. Програмний модуль поділено на окремі блоки:

Розпізнавання зображень (використання CNN);

Інтеграція з Discord.

Обґрунтування:

Модульний підхід забезпечує гнучкість, масштабованість та можливість оптимізації кожного компонента окремо.

## 2.3 Постановка задачі моделювання: обґрунтування припущень та розробка базової моделі

Задача постає таким чином: Розробка базової моделі для автоматизованого розпізнавання будівель на скріншотах із гри Clash of Clans та пошуку релевантних відео-стратегій на платформі YouTube. Система повинна бути інтегрована з платформою Discord для надання користувачам результатів аналізу.

Ціль моделювання:

Створити модель, яка зможе точно розпізнавати будівлі на зображенні за допомогою комп'ютерного зору.

Забезпечити можливість пошуку схожих стратегій у відео на YouTube.

Реалізувати взаємодію через Discord-бот для отримання результатів у зручному для користувача форматі.

Під час моделювання висуваються такі припущення:

Вхідні дані є якісними: Зображення, що надходять від користувачів, мають достатню роздільну здатність і мінімальні шуми для коректної роботи алгоритмів комп'ютерного зору.

Будівлі в грі Clash of Clans є унікальними за структурою: Різні типи будівель мають характерні ознаки (форма, колір, текстура), що дозволяє ефективно їх розпізнавати.

Скриншоти баз мають стандартний формат: Візуальні дані відповідають конкретній ігровій перспективі.

Базова модель складається з трьох основних компонентів:

- Компонент розпізнавання об'єктів

Використання Convolutional Neural Networks (CNN) або попередньо натренованих моделей (наприклад, YOLOv5 чи Faster R-CNN) для детекції та класифікації будівель на скриншоті.

Вхід: Зображення бази.

Вихід: Координати (bounding boxes) будівель та їх типи.

Математична модель детекції об'єктів (YOLO):

Зображення ділиться на сітку

$S \times S$. У кожній клітинці прогнозуються:

Координати рамки: $x, y, w, h$

Ймовірність наявності об'єкта: $p$;

Клас об'єкта: $c_i$.

Загальна формула для розрахунку втрат (Loss Function):$L = L\text{coord} + L\text{conf} + L\text{class}$,

де $L$coord відповідає за точність координат, $L$conf – за ймовірність наявності об'єкта, а $L$class – за точність класифікації.

- Компонент пошуку відео

Здійснення порівняння векторів ознак розпізнаних будівель зі збереженими зразками або на основі попередньо визначених шаблонів. Для релевантного пошуку відео застосовується метод порівняння векторів ознак з використанням евклідової відстані або косинусної подібності.

Вхід: Вектор ознак будівель.

Вихід: Ідентифікатор відео (Video ID) на YouTube.

- Компонент інтеграції з Discord

Використання Discord API для розробки бота, що отримує скриншот, запускає процес розпізнавання і повертає користувачу результати: список будівель, ідентифікованих на зображенні, та посилання на релевантні відео.

Функціонал:

Завантаження зображень від користувача.

Виклик моделі розпізнавання.

Повернення відповіді у вигляді зручного текстового повідомлення з відеопосиланнями.

## 2.4 Розробка основних алгоритмів програмного забезпечення та методик проведення моделювання

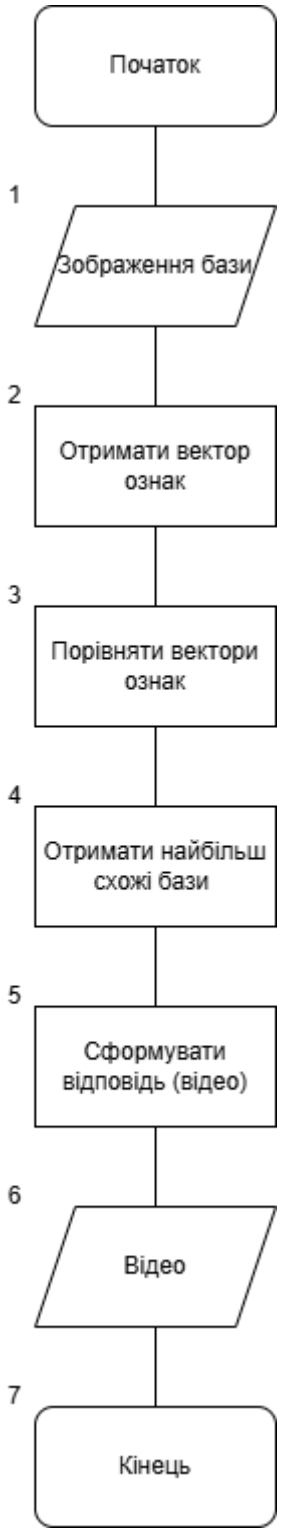


Рисунок 2.1 - Основна блок-схема функціонування бота

Користувач відкриває Discord-сервер «How To Beat», у текстовому каналі «#scan-the-base» вводить команду /scan, тисне Enter, після чого поле

вводу очікуватиме на зображення (скріншот бази), додати зображення можна за допомогою «перетягування» (Drag and drop) або ж натиснувши на величезне поле для зображення (див. рис. 2.2), після чого зображення має з’явитися замість цього поля, тепер уже в формі попереднього перегляду (див. рис.2.3).

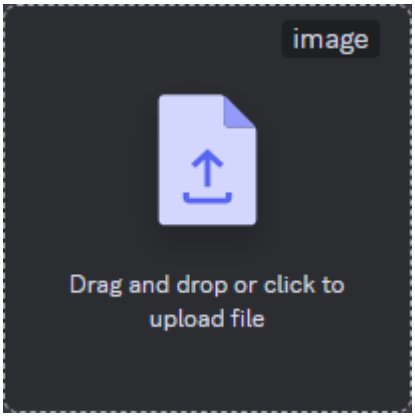


Рисунок 2.2 - Поле для завантаження зображення у Discord


Рисунок 2.3 - Поле для зображення з вже завантаженим зображенням у Discord

Після успішного завантаження зображення користувач ще раз натискає Enter і бот починає оброблювати це зображення і отримує вектор ознак, який він потім порівнює з базою даних баз і отримує найбільш схожу базу (за наявності) і формує відповідь користувачеві, в якій надає посилання на відео, в якому цю базу успішно атакують і посилання на копіювання цієї бази на випадок тренування атаки.

Структурна методика моделювання передбачає ієрархічний поділ системи на підсистеми та модулі, що працюють разом для досягнення єдиної мети. У моїй кваліфікаційній роботі, яка стосується розробки Discord-бота для розпізнавання будівель на скріншотах гри Clash of Clans і пошуку відео-стратегій, структурна методика допоможе створити зрозумілу архітектуру програмного рішення, що є логічно організованою та легко масштабованою.

**Основні принципи структурної методики**

- Декомпозиція: Складна задача розбивається на окремі підзадачі або модулі, кожен із яких виконує конкретну функцію.
- Модульність: Кожен компонент є автономним і має чітко визначений інтерфейс для взаємодії з іншими модулями.
- Ієрархічність: Компоненти системи організуються у вигляді рівнів: від базових модулів до більш складних систем.
- Гнучкість: Можливість легко оновлювати або замінювати модулі без впливу на інші частини системи.

**Побудова структурної моделі системи**

Система розробки Discord-бота для **Clash of Clans** складається з таких основних структурних компонентів:

**Рівень 1: Інтерфейс користувача - модуль взаємодії з Discord**

- Функція: Забезпечення комунікації користувача з ботом через текстові команди та завантаження скріншотів.
- Інтерфейс: Discord API (вхідні дані – скріншот бази).
- Вихід: Передача отриманих даних іншим модулям та повернення результатів у текстовому форматі з посиланнями на відео.

**Рівень 2: Модуль обробки зображень - підсистема розпізнавання будівель (Object Detection System)**

- Функція:
    - Аналіз завантаженого скріншота.
    - Локалізація будівель на зображенні та їх класифікація (наприклад, ратуша, казарма, ресурсні сховища).

- Технології:
    - Згорткові нейронні мережі (YOLOv5 або Faster R-CNN) для детекції об’єктів.
- Архітектура модуля:
    - Вхід: Зображення у форматі .png або .jpg.
    - Вихід: Координати об’єктів (bounding boxes) і клас будівель (тип).

**Рівень 3: Модуль пошуку**

- Функція:
    - Порівняння вектора ознак з базою даних за допомогою метрик подібності (Косинусної подібності, Евклідової відстані).
    - Ранжування результатів – відео із найбільшою косинусною подібнісью, або найменшою евклідовою відстанню вважається найбільш релевантним.
- Вихід:
    - Посилання на відео;
    - Рівень схожості

**Рівень 4: Модуль об'єднання результатів - підсистема інтеграції та виводу результатів**

- Функція: Формування кінцевого результату для користувача на основі отриманих даних від інших модулів.
- Вихід: Текстове повідомлення у Discord з результатами аналізу та посиланнями на відео.

### 2.5 Розробка моделі інтерфейсу програмного забезпечення

За допомогою можливостей Discord було створено блок-привітальне повідомлення з можливістю оглянути основні текстові канали Discord-серверу «How To Beat» (див. рис. 2.4), текстовий канал з блоком правил Discord-спільноти «How To Beat» (див. рис. 2.5), текстовий канал «scan-base», який використовується для застосування основного функціоналу програмного модулю, тобто сканування скріншоту бази, з можливістю прикріпляти

скріншот прямо до команди-повідомлення «/scan» (див. рис. 2.6), і отримання результатів сканування (див. рис. 2.7) [14].


Рисунок 2.4 - Привітальне повідомлення при першому доєднанні на Discord-сервер «How To Beat»

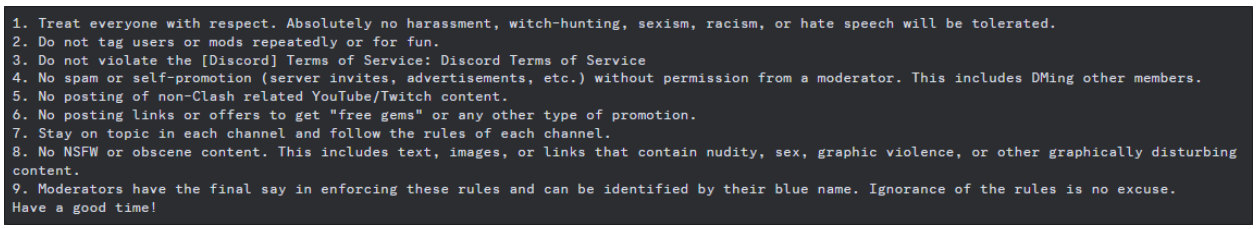


Рисунок 2.5 - Блок правил Discord-спільноти «How To Beat»

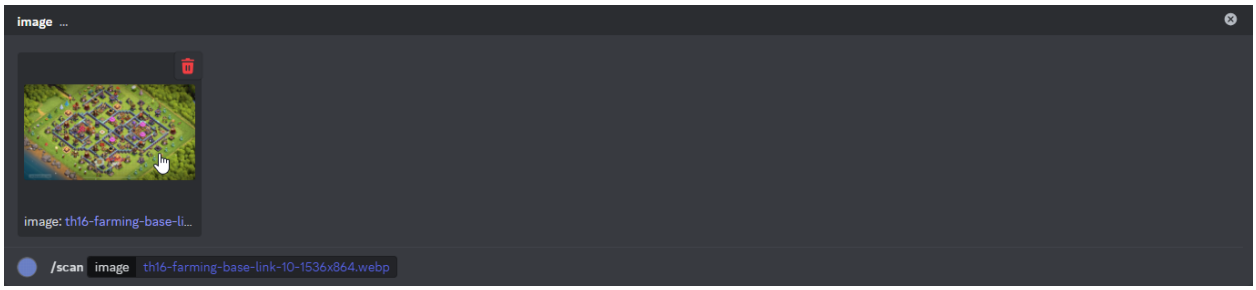


Рисунок 2.6 - Застосування команди /scan у текстовому каналі "scan-base" на Discord-сервері «How To Beat»


Рисунок 2.7 - Результат застосування команди /scan у текстовому каналі "scan-base" на Discord-сервері «How To Beat»

# 3 ПРИКЛАДНА ЧАСТИНА

## 3.1 Обґрунтування і вибір інструментів розробки програмного забезпечення;

Для виконання кваліфікаційної роботи, яка передбачає розробку Discord-бота з використанням комп'ютерного зору та інтеграції з базами даних, Visual Studio Code (VSCode) є оптимальним вибором завдяки його численним перевагам, серед яких:

- Підтримка широкого набору мов програмування - Python є основною мовою у моєму проєкті для розробки нейронної мережі, обробки зображень та інтеграції з Discord API, а VSCode в свою чергу має вбудовану підтримку Python завдяки розширенню Python Extension for Visual Studio Code, яке забезпечує:
    - Синтаксичне підсвічування та автодоповнення коду;
    - Інтерактивний запуск скриптів у терміналі чи Jupyter Notebook;
    - Відлагодження (debugging) із точками зупинки для покрокової перевірки виконання коду.
- Легкість та продуктивність - VSCode є легким середовищем порівняно з важкими IDE, такими як PyCharm або Eclipse, має швидке завантаження та низьке споживання ресурсів, що дозволяє ефективно працювати навіть на середніх за продуктивністю комп'ютерах;
- Гнучкість та можливість розширення - VSCode є модульним і дозволяє налаштувати середовище під конкретні потреби мого проєкту завдяки розширенням (Extensions);
- Інтегрований термінал та середовище відлагодження - У VSCode є вбудований термінал, що дозволяє виконувати Python-скрипти без перемикання між середовищами та встановлювати бібліотеки за допомогою pip;
- Інтеграція з Git та GitHub - VSCode має вбудовану підтримку системи контролю версій (Git), що дозволяє ініціалізувати репозиторії і

відслідковувати зміни у проєкті, завантажувати код на GitHub, що важливо для демонстрації результатів роботи;

- Зручність роботи з даними - оскільки моя робота передбачає обробку великих обсягів даних (вектори ознак), VSCode має функціонал для зручної роботи з ними, вбудовані засоби для аналізу файлів даних у форматах JSON, CSV;
- Кросплатформеність - VSCode працює на всіх основних операційних системах: Windows, macOS, Linux, це дозволяє використовувати одне середовище для розробки незалежно від платформи;
- Багатозадачність та робота з проєктами - VSCode підтримує роботу з кількома файлами та проєктами одночасно, що дозволяє працювати над модулем обробки зображень у Python і паралельно тестувати інтеграцію з Discord API.

Всі ці фактори роблять Visual Studio Code напевно найкращим вибором серед IDE у моєму випадку [15].

Вибір мови програмування був відносно очевидним, бо основним вектором у моєму проекті є робота з нейронними мережами, а Python є найпопулярнішою мовою для розробки систем машинного навчання та штучного інтелекту і, звісно, через об’ємний попередній досвід роботи з цією мовою програмування, але також не можна не згадати і про інші переваги Python, серед яких є:

- Простота синтаксису та швидкість розробки - Python має лаконічний та зрозумілий синтаксис, що дозволяє швидко писати, читати та підтримувати код. Це зменшує час на розробку проєкту та підвищує продуктивність. Python має лаконічний та зрозумілий синтаксис, що дозволяє швидко писати, читати та підтримувати код. Це зменшує час на розробку проєкту та підвищує продуктивність;
- Розвинена екосистема бібліотек та фреймворків - Python пропонує великий набір бібліотек, необхідних для виконання ключових завдань у моєму проєкті:

- OpenCV — для обробки зображень (масштабування, фільтри, перетворення);
- SciPy - для реалізації алгоритмів порівняння векторів ознак (косинусна подібність, евклідова відстань);
- discord.py — бібліотека для створення та управління Discord-ботами;
- inference_sdk - це програмний інтерфейс (SDK), що надає можливість інтеграції вже розгорнутої моделі з RoboFlow у Python для виконання інференсу (процесу отримання передбачень від моделі) на нових зображеннях.

- Підтримка інтеграції з Discord - Бібліотека discord.py є стандартом для створення ботів у Discord на Python. Легкість налаштування та інтеграції API дозволяє створити бот, що:
  - Приймає зображення від користувача;
  - Запускає модель для аналізу зображення;
  - Повертає результати у форматі текстових повідомлень.
- Велика спільнота та документація - Python має велику спільноту розробників, що дозволяє знайти відповіді на запитання або приклади коду на форумах (Stack Overflow, GitHub). Багато документації та навчальних матеріалів полегшують освоєння потрібних інструментів;
- Легке масштабування та інтеграція - Python підтримує розгортання системи на сервері з використанням Docker або AWS Lambda для масштабування, Python підтримує розгортання системи на сервері з використанням Docker або AWS Lambda для масштабування [16].

Для роботи з Discord було обрано бібліотеку discord.py бо це - це асинхронна бібліотека для роботи з Discord API, написана мовою Python, використання якої дає:

- Офіційну підтримка та стабільність - discord.py є стандартом для створення Discord-ботів у Python завдяки офіційній підтримці та активній спільноті. Також бібліотека регулярно оновлюється та

підтримує найновіші функції Discord API, що забезпечує стабільну роботу мого бота;

- Простоту використання - discord.py має зручний і зрозумілий синтаксис, що дозволяє швидко створювати функціонал для бота, навіть без глибокого досвіду в роботі з API, а для обробки завдань, таких як отримання зображення від користувача або надсилання текстової відповіді, потрібен мінімум коду;
- Асинхронність та висока продуктивність - discord.py побудована на асинхронному програмуванні (async/await), що дозволяє обробляти кілька запитів одночасно, асинхронність дозволяє не блокувати роботу бота під час виконання інтенсивних обчислень. це критично важливо для моєї роботи, де бот:
    - Приймає зображення від різних користувачів;
    - Викликає нейронну мережу для обробки скріншотів (що може займати час);
    - Відповідає користувачам з результатами.
- Гнучкість та розширюваність - discord.py підтримує створення команд, обробку подій, реакції на певні дії користувача;
- Інтеграція з іншими бібліотеками Python - discord.py чудово поєднується з іншими бібліотеками Python, які використовуються у моїй роботі як от:
    - OpenCV — для обробки зображень та виявлення будівель;
    - inference_sdk – для інтеграції вже розгорнутої моделі з RoboFlow і процесу отримання передбачень від моделі;
    - SciPy - для реалізації алгоритмів порівняння векторів ознак;

  Це забезпечує гнучку архітектуру, де Discord-бот є інтерфейсом між користувачем і основним модулем обробки.
- Велика спільнота та документація - discord.py має розвинену документацію та велику спільноту розробників, що дозволяє легко знайти приклади коду або відповіді на питання;

- Можливість масштабування - discord.py дозволяє легко масштабувати функціонал:
    - Додавати нові команди (наприклад, перевірка статусу моделі, статистика запитів).
    - Оптимізувати роботу бота для обробки великої кількості одночасних запитів.

  Бот можна запустити на локальному сервері або хмарній платформі (наприклад, AWS, Heroku або Google Cloud).
- Підтримка безпеки та обмежень - Можливість обмеження доступу до певних команд або функцій через рольові системи Discord, а токен для запуску бота є захищеним і конфіденційним, що відповідає вимогам безпеки [16].

RoboFlow є однією з найсучасніших та найзручніших платформ для підготовки даних, тренування моделей комп'ютерного зору та їх інтеграції у робочі проєкти.

- Повний цикл обробки даних - RoboFlow забезпечує єдину платформу для всіх етапів роботи з даними та моделями комп'ютерного зору:
    - Завантаження даних: Можна легко завантажити набір зображень (наприклад, скріншоти Clash of Clans).
    - Анотація даних: У RoboFlow є зручний інтерфейс для ручної анотації (створення bounding boxes навколо будівель та їх класифікація).
    - Обробка даних (Augmentation): Платформа надає вбудовані інструменти для збільшення та покращення набору даних:
    - Масштабування, обертання, яскравість, контрастність, випадкові обрізання.
    - Це дозволяє створити стійку до варіацій модель навіть на невеликому наборі зображень.

- Розподіл даних: Автоматичний поділ даних на тренувальний, валідаційний та тестовий набори, що є стандартом для навчання моделей машинного навчання.

- Готові архітектури нейронних мереж - RoboFlow підтримує найпопулярніші архітектури для обробки зображень та розпізнавання об'єктів:
  - YOLOv5 (You Only Look Once):
  - Використана у твоєму проєкті для швидкого та точного розпізнавання будівель на скріншотах.
  - Забезпечує високу продуктивність з мінімальною затримкою, що є критичним для інтеграції в Discord-бот.
  - Інші архітектури (Faster R-CNN, SSD) доступні за необхідності для різних задач.
  - Перевага: У RoboFlow можна налаштувати параметри навчання (кількість епох, learning rate, розмір зображень) без глибокого знання низькорівневого коду.
- Хмарне навчання моделей:
  - RoboFlow виконує тренування моделей у хмарному середовищі із використанням GPU.
  - Це дозволяє уникнути необхідності налаштовувати складні середовища локально, що економить час і ресурси.
  - Тренування можна запускати у фоновому режимі, а результати доступні у вигляді графіків і метрик.
- Інтеграція з готовими моделями через API
  - Після навчання модель можна розгорнути як API та отримати доступ до неї за допомогою RoboFlow Inference SDK.
  - Це спрощує інтеграцію моделі у твій Discord-бот:
  - Бот надсилає зображення на сервер RoboFlow.

- Модель обробляє зображення та повертає результат у форматі JSON.

- Візуалізація та моніторинг результатів
  - RoboFlow надає зручні графіки метрик (Precision, Recall, Loss, mAP), що дозволяють аналізувати якість навчання моделі.
  - Ти можеш переглянути результати роботи моделі на тестових даних, що допомагає оцінити її ефективність.
- Підтримка експорту моделей:
  - Після навчання модель можна експортувати у різних форматах для подальшого використання:
  - PyTorch — для інтеграції у Python.
  - ONNX — для сумісності з іншими платформами.
  - TensorFlow — для мобільних чи веб-застосунків.
  - RoboFlow API — для хмарного інференсу.
  - Це забезпечує гнучкість та дає можливість адаптувати модель для різних середовищ.
- Швидкість розробки та простота у використанні:
  - Використання RoboFlow дозволяє зменшити час на розробку моделі, оскільки більшість етапів автоматизовано:
  - Анотація даних.
  - Обробка та збільшення набору даних.
  - Навчання у хмарі.
  - Розгортання та інтеграція.
  - Інтерфейс RoboFlow є зрозумілим і доступним, що дозволяє швидко освоїти всі необхідні функції.
- Оптимізація продуктивності:
  - Моделі, створені у RoboFlow (наприклад, YOLOv5), оптимізовані для швидкої обробки зображень, що дозволяє виконувати інференс у реальному часі.

- Це особливо важливо для Discord-бота, де швидкість відповіді є критичною [17].

В якості бібліотеки комп’ютерного зору було обрано OpenCV (Open Source Computer Vision Library) — це потужна бібліотека з відкритим вихідним кодом для комп’ютерного зору, обробки зображень та відео, перевагами якої є:

- Універсальність для обробки зображень - OpenCV надає широкий набір інструментів для попередньої обробки зображень, що є важливим етапом у розпізнаванні об'єктів:
  - Масштабування зображень: зміна розмірів для приведення до єдиного формату.
  - Перетворення кольору: конвертація зображень у відтінки сірого або інші колірні простори (наприклад, BGR → Grayscale, HSV).
  - Фільтрація шумів: згладжування та зменшення артефактів, які можуть заважати роботі моделі.
  - Покращення контрастності: використання гістограмної рівномірності для чіткішого виділення об'єктів.
- Інтеграція з нейронними мережами (YOLO, TensorFlow, PyTorch):
  - OpenCV має вбудовану підтримку для інтеграції з моделями нейронних мереж, такими як YOLOv5, Faster R-CNN, SSD.
  - Це дозволяє комбінувати попередню обробку зображень (масштабування, фільтрація шумів) з інференсом моделей для розпізнавання об'єктів.
  - OpenCV підтримує формати моделей ONNX та DNN, що забезпечує ефективний інференс навіть на слабких обчислювальних системах.
- Швидкість роботи та оптимізація:
  - OpenCV написана на C++ із оптимізацією для швидкодії, що робить її ідеальною для задач реального часу.

  - Завдяки цьому OpenCV швидко виконує операції обробки зображень та інференс навіть на обмежених обчислювальних ресурсах.
- Легка інтеграція з Python:
  - OpenCV має зручний Python API, що дозволяє поєднати обробку зображень із нейронними мережами у рамках однієї програми.
  - Це робить OpenCV ідеальним для інтеграції у мій Discord-бот:
  - Отримання зображень від користувача.
  - Попередня обробка за допомогою OpenCV.
  - Передача обробленого зображення на модель для розпізнавання.
- Підтримка роботи з різними форматами даних - OpenCV підтримує всі основні формати зображень і відео (PNG, JPEG, BMP тощо), що важливо для моєї роботи, оскільки користувачі можуть надсилати зображення у різних форматах.
- Візуалізація результатів - OpenCV надає зручні інструменти для відображення результатів детекції об'єктів:
  - Малювання bounding boxes навколо будівель на зображенні.
  - Відображення текстових міток (наприклад, клас об'єкта, точність розпізнавання).
- Велика спільнота та документація:
  - OpenCV є однією з найпопулярніших бібліотек для обробки зображень, тому вона має велику спільноту розробників.
  - У разі виникнення проблем ти завжди можеш знайти готові рішення або приклади на форумах, GitHub чи Stack Overflow.
- Масштабованість та розширюваність:

- OpenCV дозволяє масштабувати твоє рішення для обробки зображень у реальному часі (наприклад, для роботи з потоками відео чи великими наборами зображень).
- Це дає змогу розширювати функціонал програми у майбутньому. [18].

А для інтеграції моделі комп'ютерного зору, зібраної на платформі RoboFlow було обрано inference_sdk, що забезпечує:

- Швидку інтеграція моделі у проєкт:
  - Навчання моделі комп'ютерного зору відбувалось на платформі RoboFlow, а inference_sdk дозволяє швидко підключити цю модель до моєї програми.
  - Інструмент забезпечує простий доступ до моделі через HTTP API, дозволяючи виконувати інференс без необхідності завантажувати модель локально чи налаштовувати сервер.
- Хмарне виконання інференсу:
  - Завдяки inference_sdk, інференс виконується у хмарі RoboFlow, що дозволяє уникнути високих вимог до локального обладнання.
  - Це особливо важливо для Discord-бота, який повинен бути легким і швидким, працюючи на хостингу чи сервері з обмеженими ресурсами.
  - Хмарне середовище забезпечує стабільну продуктивність навіть за великої кількості запитів.
- Легкість використання:
  - inference_sdk автоматизує всі складні процеси роботи з моделлю:
  - Завантаження та підготовка зображення.
  - Відправка запиту до сервера.
  - Отримання результатів у форматі JSON.

Це дозволяє зосередитися на логіці моєї програми, не витрачаючи час на складну інтеграцію чи розгортання моделі.

- Ефективна взаємодія з Discord - бібліотека легко інтегрується з Discord API, дозволяючи передавати зображення, завантажені користувачами, до RoboFlow для аналізу, а потім повертати результати у форматі, зрозумілому користувачам.
- Масштабованість:
    - Хмарна інфраструктура RoboFlow та inference_sdk дозволяють обробляти багато запитів одночасно, забезпечуючи високу масштабованість.
    - Це означає, що бот може обслуговувати багато користувачів без затримок або зниження продуктивності.
- Структуровані результати:
    - inference_sdk повертає результати у форматі JSON, де вказані:
    - Клас розпізнаного об’єкта (тип будівлі, наприклад, "Town Hall", "Gold Storage").
    - Координати bounding box об’єкта (x, y, ширина, висота).
    - Рівень впевненості (confidence).

      Це дозволяє легко обробляти дані для подальшого використання.
- Підтримка попередньої обробки зображень:
    - inference_sdk дозволяє автоматично виконувати попередню обробку зображень, включаючи:
    - Масштабування до потрібного розміру.
    - Нормалізацію кольорів.
    - Оптимізацію вхідних даних для моделі.

      Це усуває необхідність виконувати ці операції вручну, що спрощує розробку.

- Безперервне оновлення моделі:
    - Якщо модель у RoboFlow оновлюється (наприклад, із додаванням нових класів об'єктів або кращим навчанням), inference_sdk автоматично підключається до нової версії.
    - Це забезпечує постійну актуальність та точність передбачень без необхідності повторного розгортання.
- Стабільність та безпека:
    - inference_sdk працює через захищені API-запити, що забезпечує безпеку даних користувачів.
    - Підтримка ключів API дозволяє контролювати доступ до моделі та запобігає несанкціонованому використанню.
- Гнучкість у налаштуванні:
    - inference_sdk дозволяє задавати додаткові параметри для інференсу, наприклад:
    - Поріг впевненості (confidence threshold): повертаються тільки результати, які перевищують певний рівень довіри.
    - Максимальне перекриття (overlap): мінімізація дублікатів розпізнаних об'єктів [19].

## 3.2 Навчання моделі комп’ютерного зору

Для навчання моделі було використано платформу RoboFlow, також відібрано велику кількість скріншотів баз Clash of Clans, на якій було розмічено кожну захисну будівлю на базі (див. рис. 3.1). Для цього було завантажено на сайт скріншоти баз, потім за допомогою функціоналу сайту було розмічено класи, після чого було сформовано датасет з зображень з анотаціями, модель було навчено за пропорцією 70%/20%/10%, де 70% екземплярів пішло на навчання, 20% на валідацію, а 10% на тестування моделі.

Рисунок 3.1 - Приклад розміченої бази у RoboFlow

### 3.3 Дослідження отриманих результатів роботи нейронної мережі

Модель була оцінена за наступними метриками (див. рис. 3.2):

- mAP (Mean Average Precision): 79.1% - mAP є ключовою метрикою для оцінки якості об'єктів, що детектуються. Вона враховує як точність, так і повноту на різних порогах IoU (Intersection over Union). 79.1% є досить високим значенням, яке свідчить, що модель добре справляється з більшістю об'єктів у датасеті.
- Precision (Точність): 90.5% - З усіх об'єктів, що були передбачені як належні до певного класу, 90.5% справді належать до цього класу. Це свідчить, що модель рідко видає помилкові спрацьовування (False Positives).

- Recall (Повнота): 69% - З усіх реальних об'єктів, присутніх у зображенні, модель правильно виявляє 69%. Значення вказує, що модель не завжди знаходить всі об'єкти (False Negatives) [20].


Рисунок 3.2 - Результат навчання моделі на плафтормі RoboFlow

# 4 АНАЛІЗ ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ПРОЕКТУ

## 4.1 Інноваційна ефективність програмного модуля керування ігровими стратегіями

Програмний модуль керування ігровими стратегіями, розроблений у рамках моєї роботи, має потенціал бути економічно ефективним та інноваційним рішенням завдяки оптимізації ігрових процесів, автоматизації аналізу та інтеграції з сучасними інструментами.

Основними критеріями, що впливають на це наступні:

1. Скорочення витрат на пошук стратегій
    - Оптимізація пошуку інформації - У традиційному підході гравці витрачають значний час на пошук відео з релевантними стратегіями атак. Модуль автоматизує цей процес: на основі скриншота бази знаходяться найкращі відео з готовими стратегіями атак.
    - Ефектом є: скорочення часу на пошук інформації до кількох секунд; більше часу для практичної реалізації стратегії та гри; швидка підготовка до атаки

      Завдяки швидкому аналізу бази та вибору стратегій модуль знижує час підготовки до атак, що особливо важливо під час клановых війн або турнірів.
2. Підвищення ефективності використання ресурсів
    - Зменшення % невдалих атак - вибір оптимальної стратегії дозволяє зменшити кількість невдалих атак, що зазвичай призводять до гірших результатів на клановых війнах та лігах клановых війн, а як наслідок - недоотримання ігрових ресурсів.
    - Результат - економія ігрових ресурсів, більше ефективних атак на бази, що збільшує прогрес гравця.

- Оптимізація витрат часу в грі - енше часу витрачається на експериментальні атаки та навчання завдяки наданню перевірених стратегій.

3. Потенціал монетизації
    - Програмний модуль має кілька каналів монетизації, які забезпечують як прямий, так і непрямий економічний ефект:
    - Підписка на преміум-функції - Користувачі можуть отримати доступ до розширеного функціоналу, наприклад: персоналізовані рекомендації (коучінг), аналітика атак, визначення слабких місць бази.
    - Модель підписки - місячна або річна підписка, разовий платіж за аналітику.
    - Реклама - інтеграція реклами у відповіді бота або на платформах, що пов'язані з ним.
    - Платний доступ до унікальних стратегій - продаж ексклюзивного контенту, наприклад, ефективних стратегій, які не доступні у відкритому доступі.
    - Партнерство з авторами контенту - інтеграція відео популярних авторів або каналів YouTube у результати пошуку, отримання партнерської комісії за перегляди або підписки на канали.
    - Залучення нових гравців до гри - Інструмент може стати частиною екосистеми Clash of Clans, стимулюючи гравців активніше грати або повертатися до гри після перерви.
4. Соціальний та ринковий ефект
    - Підтримка новачків - модуль дозволяє новим гравцям швидко вчитися, уникаючи складнощів із пошуком стратегій, це сприяє зростанню залученості гравців, що позитивно впливає на тривалість їхньої активності у грі.

- Зростання активності спільноти - інструмент стимулює обговорення стратегій серед гравців, сприяючи збільшенню активності у спільнотах (Discord, форуми, соціальні мережі).

## 4.2 Розрахунок собівартості програмного модуля керування ігровими стратегіями

В таблиці 4.1. наведено початкові дані для розрахунку собівартості.

Таблиця 4.1 - Початкові дані для розрахунку собівартості

| Найменування початкових даних | Показник | Джерело отримання |
|---|---|---|
| Трудомісткість розробки програмного модуля, днів | 20 | Фактичні витрати часу на розробку програмного продукту (1 місяць по 20 робочих днів) |
| Місячна ставка розробника програмного модуля, грн | 21650,00 | |
| Кількість годин в місяці, год | 160 | Кількість робочих днів — 20 |
| Додаткова зарплата (%) | 10 | |
| Відрахування до соціальних фондів (%) | 15 | |
| Загальновиробничі витрати (%) | 100 | |
| ПДВ (податок на додану вартість) (%) | 20 | |

Розрахуємо витрати на електроенергію за заданими даними за формулою 4.1:

$$В_{електр} = W * T * C, \qquad (4.1)$$

де $W$ — потужність обладнання (кВт), яке використовується;

$T$ — час роботи (години);

$C$ — вартість 1 кВт·год електроенергії (грн).

$$В_{електр} = 0{,}150 \times 160 \times 4{,}32 = \ 103{,}68 \text{ грн} \qquad (4.2)$$

Тепер розрахуємо годинну тарифну ставку розробника програмного модуля, грн за формулою 4.3:

$$l_{год} = З_{осн} : T_{год}, \qquad (4.3)$$

де $l_{год}$ — годинна тарифна ставка розробника, грн.;

$T_{год}$ — кількість годин у місяці, приймається 160 год. — вхідні дані.

$$l_{год} = 21650{,}00 : 160 = 135{,}3125 \text{ грн} \qquad (4.4)$$

Додаткова заробітна плата визначається за формулою 4.5:

$$З_{дод} = \frac{З_{осн} * Д\%}{100}, \qquad (4.5)$$

де $З_{дод}$ — додаткова заробітна плата, грн.;

Д% — відсоток додаткової заробітної плати, приймається 10% – вхідні дані.

$$З_{дод} = \frac{21650 * 10}{100} = 2165{,}00 \text{ грн.} \qquad (4.6)$$

Відрахування в соціальні фонди знайдемо за формулою 4.7:

$$З_{соц} = \frac{(З_{осн} + З_{доп}) * С\%}{100}, \qquad (4.7)$$

де $З_{соц}$ —відрахування в соціальні фонди, грн.;

$$З_{соц} = \frac{(21650+2165)*15}{100} = 3\ 572{,}25 \text{ грн.} \qquad (4.8)$$

Визначимо загальновиробничі витрати за формулою 4.9:

$$З_{заг} = \frac{З_{осн}*H_1\%}{100}, \qquad (4.9)$$

де $З_{заг}$ — загальновиробничі витрати, грн.;

$H_1\%$ — відсоток загальновиробничих витрат, приймається 100% — вхідні дані.

$$З_{заг} = \frac{21650*100}{100} = 21650{,}00 \text{ грн.} \qquad (4.10)$$

Загальновиробничі витрати дорівнюють $21650{,}00$ грн.

Визначимо виробничу собівартість склавши всі попередні розрахунки за формулою 4.11:

$$S_{пп} = 103{,}68 + 21650.00 + 2165.00 + 3572{,}25 + 21650.00 = 49\ 140{,}93 \text{ грн.} \qquad (4.11)$$

де $S_{пп}$ — виробнича собівартість, грн.

Виробнича собівартість дорівнює $49\ 140{,}93$ грн.

В таблиці 4.2 наведено планову калькуляцію виробничої собівартості виробничої собівартості, ціни підприємства й ціни для замовника на виконання розробки програми.

Таблиця 4.2 - Калькуляція собівартості

| Статті калькуляції | Сума, грн. |
|---|---|
| Стаття 2. Витрати на електроенергію: | 103,68 |
| Стаття 3 Основна заробітна плата | 21650,00 |
| Стаття 4 Додаткова заробітна плата | 2165,00 |

Продовження таблиці 4.2 - Калькуляція собівартості

| | |
|---|---|
| Стаття 5 Відрахування в соціальні фонди | 3572,25 |
| Стаття 6 Загальновиробничі витрати | 21650,00 |
| Виробнича собівартість, 1 місяць | 49 140,93 |
| Собівартість ПЗ | 98 281,86 |

У процесі виконання розрахунків було визначено основні складові собівартості проекту, що дало можливість оцінити реальні витрати на його реалізацію. Розрахунок включав витрати на електроенергію, основну та додаткову заробітну плату, відрахування у соціальні фонди та загальновиробничі витрати.

Отримані результати свідчать про те, що розрахунки проведено з урахуванням усіх необхідних чинників і відповідають економічно обґрунтованим параметрам. Визначена собівартість дозволяє зробити висновок про ефективність планування витрат і раціональне використання ресурсів у межах проекту.

# ВИСНОВКИ

У рамках виконання кваліфікаційної роботи було досліджено та розроблено програмний модуль керування ігровими стратегіями для автоматизації аналізу баз у грі Clash of Clans.

Значна кількість уваги була приділена огляду актуальних рішень у сфері комп’ютерного зору, аналізу аналогічних розробок та виявленню їх недоліків для недопускання цих недоліків у розробці свого програмного модулю, все це було враховано при створенні перелік задач кваліфікаційної роботи.

Також було проведене моделювання сценарію взаємодії користувача з системою, після чого було змодельовано алгоритм роботи системи та складено до нього функціональну блок-схему.

До процесу вибору інструментів розробки програмного забезпечення була дуже ґрунтовна підготовка, щоб обрати максимально ефективні та найпотужніші рішення у своїх нішах, серед яких бібліотеки комп’ютерного зору, інтеграції з Discord, виконання інференсу та розрахунку метрик.

Було виконане моделювання та розробка модулів взаємодії з Discord, обробки зображень, пошуку та об'єднання результатів.

Загалом, виконання магістерської кваліфікаційної роботи надало чітке розуміння того, з якими інструментами найкраще працювати для розробки інтегрованих до платформb обміну миттєвими повідомленнями та цифрового розповсюдження інформації Discord чат-ботів та відкрило неосяжний світ комп’ютерного зору. У майбутньому, за потреби, програмний модуль може нарощувати свій функціонал та масштабуватися у випадку достатнього попиту серед аналогічних рішень.

# ПЕРЕЛІК ПОСИЛАНЬ


1. DALL·E: Creating images from text [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://openai.com/index/dall-e/.
2. SSD: Single Shot MultiBox Detector [Електронний ресурс] / [W. Liu, D. Anguelov, D. Erhan та ін.]. – 2016. – Режим доступу до ресурсу: https://arxiv.org/abs/1512.02325.
3. A Simple Framework for Contrastive Learning of Visual Representations [Електронний ресурс] / T.Chen, S. Kornblith, M. Norouzi, G. Hinton. – 2020. – Режим доступу до ресурсу: https://arxiv.org/pdf/2002.05709.
4. Bootstrap your own latent: A new approach to self-supervised Learning [Електронний ресурс] / [J. Grill, F. Strub, F. Altché та ін.]. – 2020. – Режим доступу до ресурсу: https://arxiv.org/pdf/2006.07733.
5. Welcome to the App Directory! [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: https://support-apps.discord.com/hc/en-us/articles/26501737399575-Welcome-to-the-App-Directory.
6. BurntBase Bot [Електронний ресурс] // Discord App Directory – Режим доступу до ресурсу: https://discord.com/application-directory/661444420733763614.
7. THE TOWN HALL 17 UPDATE IS HERE! [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: https://supercell.com/en/games/clashofclans/blog/game-updates/the-town-hall-17-update-is-here-2/.
8. Офіційний Discord- сервер BurntBase [Електронний ресурс] – Режим доступу до ресурсу: https://discord.gg/epKU44rA.
9. Become a BurntBase Affiliate [Електронний ресурс] – Режим доступу до ресурсу: https://www.burntbase.com/affiliate-program.

10. Find This Base Bot [Електронний ресурс] // Discord App Directory – Режим доступу до ресурсу: https://discord.com/application-directory/903995426179526696.
11. Офіційний Discord- сервер Find This Base [Електронний ресурс] – Режим доступу до ресурсу: https://discord.gg/8EV8eRY.
12. Purchase Find This Base Scantokens for Find This Base [Електронний ресурс] – Режим доступу до ресурсу: https://findthisbase.com/tokens?serverId=ee2e827f-15d7-46e4-b695-b979ad331cab.
13. FAN CONTENT POLICY [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://supercell.com/en/fan-content-policy/en/.
14. Discord Interactions [Електронний ресурс] – Режим доступу до ресурсу: https://discord.com/developers/docs/interactions/receiving-and-responding
15. Visual Studio Code documentation [Електронний ресурс] – Режим доступу до ресурсу: https://code.visualstudio.com/docs.
16. Python 3.13.1 documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/.
17. Build Vision Models with Roboflow [Електронний ресурс] – Режим доступу до ресурсу: https://docs.roboflow.com/.
18. OpenCV modules [Електронний ресурс] – Режим доступу до ресурсу: https://docs.opencv.org/4.x/index.html.
19. Roboflow Inference [Електронний ресурс] – Режим доступу до ресурсу: https://inference.roboflow.com/inference_helpers/inference_sdk/.
20. Clash of Clans Dataset [Електронний ресурс] – Режим доступу до ресурсу: https://universe.roboflow.com/kgft4gi3aghk82a/targetarch/dataset/5.

# Додаток А – Вихідний код програми

*main.py*

```
import json
import discord
from discord.ext import commands
from image_process import get_features
from similar import get_similar
import os

discord_token = <DISCORD_TOKEN>

with open('bases.json', 'r') as j:
    db_bases = json.loads(j.read())

intents = discord.Intents.default()
intents.message_content = True
intents.messages = True


bot = commands.Bot(command_prefix="!", intents=intents)

@bot.event
async def on_ready():
    print(f"Logged in as {bot.user}")
    try:
        synced = await bot.tree.sync()
        print(f"Synced {len(synced)} commands")
    except Exception as e:
        print(e)

@bot.tree.command(name="scan", description="Scan an image")
async def scan(interaction: discord.Interaction, image:
discord.Attachment):
    await interaction.response.send_message("Please wait the
result will be sent in a while.")

    if not os.path.exists('data'):
        os.makedirs('data')

    image_path = os.path.join('data', image.filename)
    await image.save(image_path)

    features = get_features(image_path=image_path)
    similar = get_similar(features, db_bases)

    await
interaction.followup.send(f"{similar['youtube_link']}")


bot.run(discord_token)
```

*similar.py*

```
    from math import sqrt
    from collections import defaultdict
    from scipy.optimize import linear_sum_assignment

    PENALTY_VALUE = 10.0

    def euclidean_dist(b1, b2):
        x_center_1 = (b1['x1'] + b1['x2']) / 2.0
        y_center_1 = (b1['y1'] + b1['y2']) / 2.0
        x_center_2 = (b2['x1'] + b2['x2']) / 2.0
        y_center_2 = (b2['y1'] + b2['y2']) / 2.0

        return sqrt((x_center_1 - x_center_2)**2 + (y_center_1 -
y_center_2)**2)

    def similarity_score(user_base, cand_base):
        user_by_type = defaultdict(list)
        for ub in user_base:
            user_by_type[ub['build_type']].append(ub)

        candidate_by_type = defaultdict(list)
        for cb in cand_base:
            if not isinstance(cb, str):
                candidate_by_type[cb['build_type']].append(cb)

        total_distance = 0.0
        all_types =
set(user_by_type.keys()).union(set(candidate_by_type.keys()))
        for t in all_types:
            user_b = user_by_type[t]
            cand_b = candidate_by_type[t]

            # Якщо для цього типу немає будівель у одного з
наборів
            if not user_b and not cand_b:
                continue
            if len(user_b) == 0:
                # Всі будівлі у кандидата - зайві
                total_distance += len(cand_b) * PENALTY_VALUE
                continue
            if len(cand_b) == 0:
                # У користувача є будівлі, у кандидата немає
                total_distance += len(user_b) * PENALTY_VALUE
                continue

            # Побудова матриці відстаней: рядки - будівлі
користувача, стовпці - будівлі кандидата
            cost_matrix = []
            for u_b in user_b:
                row = []
                for c_b in cand_b:
```

```
                    dist = euclidean_dist(u_b, c_b)
                    row.append(dist)
                cost_matrix.append(row)

            # Використовуємо Hungarian Algorithm для мінімізації
сумарної відстані
            user_indices, cand_indices =
linear_sum_assignment(cost_matrix)

            # Вираховуємо суму мінімальних відстаней
            matched_distance = sum(cost_matrix[u_i][c_i] for
u_i, c_i in zip(user_indices, cand_indices))

            # Перевіряємо різницю у кількості будівель
            diff_count = abs(len(user_b) - len(cand_b))
            penalty = diff_count * PENALTY_VALUE

            total_distance += matched_distance + penalty

        return total_distance

    def get_similar(user_base, db_bases):
        best_index = None
        best_score = float('inf')
        for i, cand in enumerate(db_bases):
            score = similarity_score(user_base, cand)
            if score < best_score:
                best_index = i
                best_score = score

        return db_bases[best_index]
```

*image_process.py*

```
    import cv2
    from inference_sdk import InferenceHTTPClient

    CLIENT = InferenceHTTPClient(
        api_url="https://detect.roboflow.com",
        api_key=<ROBOFLOW_API_KEY>
    )

    names = {'AD': 0, 'AirSweeper': 1, 'BombTower': 2, 'Canon':
3, 'ClanCastle': 4, 'Eagle': 5, 'Inferno': 6, 'KingPad': 7,
'Mortar': 8, 'QueenPad': 9, 'RcPad': 10, 'Scattershot': 11,
'TH10': 12, 'TH11': 13, 'TH12': 14, 'TH13': 15, 'TH14': 16,
'TH15': 17, 'TH16': 18, 'TH4': 19, 'TH5': 20, 'TH6': 21, 'TH7':
22, 'TH8': 23, 'TH9': 24, 'WardenPad': 25, 'WizzTower': 26,
'Xbow': 27, 'builders-hut': 28, 'hidden-tesla': 29, 'monolith':
30, 'multi-archer-tower': 31, 'ricochet-tower': 32, 'spell-
tower': 33, 'archer-tower': 34}

    def get_features(image_path):
```

```
        image = cv2.imread(image_path)

        new_width = 640
        aspect_ratio = new_width / float(image.shape[1])
        new_height = int(image.shape[0] * aspect_ratio)
        resized_image = cv2.resize(image, (new_width,
new_height))
        resized_image_path = "resized.png"
        cv2.imwrite(resized_image_path, resized_image)

        result = CLIENT.infer(resized_image_path,
model_id="targetarch/5")
        features = []
        for prediction in result['predictions']:
            x1, y1, w, h = prediction['x'], prediction['y'],
prediction['width'], prediction['height']
            cl = prediction['class']
            label = names[cl]
            x2 = x1 + w
            y2 = y1 + h
            feature = {}
            feature['build_type'] = label
            feature['x1'] = float(x1) / 640.0
            feature['x2'] = float(x2) / 640.0
            feature['y1'] = float(y1) / 640.0
            feature['y2'] = float(y2) / 640.0
            features.append(feature)

        return features
```