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

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:

Завідувач кафедри

_____ Олександр Коваль

«___»_____2020 р.

Дипломна робота

на здобуття ступеня бакалавра

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

спеціальності 121 «Інженерія програмного забезпечення»

на тему: «Система автоматичного розташування камер відеоспостереження у приміщенні»

Виконала:
студентка IV курсу, групи ТІ-62
Шевчук Анна Олексіївна _____

Керівник:
старший викладач
Мірошніченко Іван Володимирович _____

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

Рецензент:
_____
_____ _____

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

Київ – 2020 року

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

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки: 121 Інженерія програмного забезпечення

Спеціалізація: Програмне забезпечення веб-технологій та мобільних пристроїв

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

ЗАВДАННЯ
на дипломну роботу студенту

_Шевчук Анні Олексіївні_____
(прізвище, ім’я, по батькові)

1. Тема роботи ___Система автоматичного розташування камер відеоспостереження у приміщенні_____
керівник роботи _____ст. викл. Мірошніченко Іван Володимирович_____
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

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

2. Строк подання студентом роботи «10» червня 2020р._____
3. Вихідні дані до роботи __мова програмування Python, фреймворк Django, бібліотека NumPy, бібліотека Matplotlib система керування базами даних MySQL._____
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) проаналізувати предметну область автоматичного розташування камер відеоспостереження у приміщенні, сформувати задачі, які повинна вирішувати система, проаналізувати аналогічні рішення, виділити їх недоліки та переваги, спроектувати систему за допомогою мови моделювання UML, розробити архітектуру системи, навести опис взаємодії складових частин системи, реалізувати алгоритм, створити програмне забезпечення веб-системи для демонстрації роботи алгоритму, проілюструвати користувацький інтерфейс системи._____
5. Перелік ілюстративного матеріалу
Схема структури системи, Діаграма прецедентів, Діаграма послідовностей, Схема алгоритму, Клієнт-серверна архітектура, Архітектура MBT, Етапи тестування при розробці продукту._____

6. Консультанти розділів роботи

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

7. Дата видачі завдання ”07”____жовтня____ 2019 р.

КАЛЕНДАРНИЙ ПЛАН

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

Студент _____ _Шевчук А. О._____
(підпис) (прізвище та ініціали,)

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

АНОТАЦІЯ

Мета роботи: розробити систему, що вирішить проблему проектування системи спостереження з використанням декількох камер за допомогою автоматичного розташування камер відеоспостережння у заданому приміщенні.

Проаналізовано наявні програми які вирішують описану проблему, описано їх переваги та недоліки та сформульовано вимоги й задачі, яким повинна відповідати система.

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

Функціонал системи розбито на підмодулі. Змодельовано та реалізовано архітектуру взаємодії всіх підсистем в рамках цілісної системи, що дозволяє легко розширювати та масштабувати продукт. Описано можливі варіанти взаємодії користувача з програмною системою.

Ключові слова: веб-додаток, клієнт-серверна архітектура, зручний інтерфейс.

ABSTRACT

Purpose: to develop a system that will solve the problem of designing a surveillance system using multiple cameras by automatically placing CCTV cameras in a given room.

The existing programs that solve the described problem are analyzed, their advantages and disadvantages are described and the requirements and tasks that the system must meet are formulated.

Thesis consists of 56 pages, 25 figures, 14 references and 3 appendices.

The functionality of the system is divided into submodules. The architecture of interaction of all subsystems within the framework of integral system is modeled and realized that allows to expand and scale a product easily. Possible options for user interaction with the software system are described.

Keywords: web application, client-server architecture, user-friendly interface.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

MVT – Model View Template

SPA – Single-Page Applications

UML – Unified Modeling Language

UI – User Interface

САПР – Система автоматизованого проектування і розрахунку

PDF – Portable Document Format

HTTP – HyperText Transfer Protocol

ПЗ – Програмне Забезпечення

ВСТУП

Останнім часом набувають все більшого поширення різноманітні алгоритми та системи Smart Home. Кількість бажаючих користуватися програмами для автоматизації щоденних задач та приладів зростає щодня. Останнім часом все більше людей мають намір захистити свої будинки. На допомогу приходять системи відеоспостереження з використанням декількох камер одночасно. Сьогодні є великий вибір камер та програмного забезпечення для їх використання.

Користувачі камер відеоспостереження часто допускають помилки у їх розташуванні, що призводить до неправильної роботи або й зовсім до відсутності результатів роботи камер. Саме тому розробка системи автоматичного розташування камер відеоспостереження у приміщеннях є актуальною в наш час. Автоматизація цього процесу значно полегшить пошук оптимального місця для кріплення камери та допоможе використовувати камери найбільш ефективно [1].

У процесі планування і проектування системи відеоспостереження потрібно визначити скільки і яких відеокамер буде потрібно, де і як розмістити камери, потрібно визначити зони огляду. Тому проектувальнику або монтажника доводиться шукати баланс між можливістю розпізнавання/ідентифікації людей в кадрі, розміром зони огляду і кількістю і типом встановлених камер.

Дана програма спеціально розроблена для установників систем відеоспостереження або кінцевих користувачів камер яким часто не вистачає часу на вирішення розрахункових завдань, виїзд на об'єкт щоб ставити досліди на місці. Одним з плюсів системи є те, що вона здатна працювати віддалено.

Програма має зручний та нескладний користувацький інтерфейс і при цьому надає всі основні функції для планування і проектування відеоспостереження.

На сьогоднішній день проблема оптимального розташування камер має чимало варіантів вирішення. Програми мають широкий спектр функцій. Деякі з них пропонують функції розпізнавання обличчя, номерів машин та інших предметів спостереження.

Проте, по при велику кількість існуючих систем, вони не задовольняють проблеми потенційних користувачів. Більшість існуючих алгоритмів створені для професійного використання, для користувачів, які добре знаються на системах відеоспостереження та спеціальних показниках. Інтерфейс таких систем є незрозумілим та незручним для пересічного користувача. Часто запит великої кількості вхідних даних відштовхує непрофесійних користувачів. А програми, що створені для домашнього використання мають високу ціну або відсутність у відкритому доступі. Деякі системи є додатками до одного бренду камер, та не можуть використовуватися з іншими.

Призначенням програмного забезпечення, яке розробляється є вирішення проблеми неправильної розстановки камер, збільшення ефективності їх використання.

Потенційними користувачами програмного забезпечення є особи без знань у сфері проектування систем відеоспостереження, які бажають використовувати камери у приватних приміщеннях.

1. ПОСТАНОВКА ЗАДАЧІ РОЗРОБКИ СИСТЕМИ АВТОМАТИЧНОГО РОЗТАШУВАННЯ КАМЕР ВІДЕОСПОСТЕРЕЖЕННЯ У ПРИМІЩЕННЯХ

Мета роботи — це вирішення проблеми статичного розміщення камери, і визначення оптимального позиціонування та кількості камер для заданого приміщення.

1.1 Задачі, які розв’язуються програмним забезпеченням; опис вхідної / вихідної інформації

Програмне забезпечення система автоматичного розташування камер відеоспостереження у приміщеннях вирішує всі основні задачі проектування системи відео нагляду. Такі як:

- Визначення скільки і яких відеокамер буде потрібно;
- Де і як розмістити камери;
- Визначення зони огляду;
- Економія коштів при проектуванні системи відеоспостереження;
- Мобільність алгоритму що розробляється.

Система може працювати віддалено, немає потреби бути присутнім у приміщенні, для якого проектується система відеокамер. Реалізація такої автоматизації буде враховувати тип, розміри приміщення та предмети, розташовані у приміщенні, які зменшують зону огляду камери.

Для вирішення задач з якими стикається користувач в процесі планування і проектування системи відеоспостереження та для коректної роботи алгоритму користувач повинен мати вхідну інформацію. Повний функціонал системи вимагає такі вхідні дані: план приміщення, налаштування камер і вид наявних камер.

На вихід користувач отримує всю необхідну інформацію для оптимального розташування камер у приміщенні: зображення плану приміщення з розташованими камерами, на якому буде видно область покриття камерами, їх кількість. Також буде виведено у текстовому форматі вид камер та їх оптимальна кількість.

На вихід користувач повинен отримати зображення плану приміщення з розставленими на ньому камерами.

1.2 Опис підсистем програмного комплексу

Перелік необхідних підсистем:

- Система управління базами даних;
- Система користувачів;
- Система додавання вхідних даних користувачем;
- Система зчитування вхідної інформації;
- Система розташування камер на плані;
- Система виведення результатів алгоритму;
- Система експорту вихідних даних програми.

1.2.1 Система управління базами даних

Система управління базою даних (СУБД) отримує інструкцію від адміністратора бази даних і, відповідно, інструктує систему про необхідність внесення необхідних змін.

У запрограмованій системі автоматичного розташування камер СУБД використовується для завантаження, вилучення або зміни існуючих даних з системи. База даних вирішує задачу зберігання даних проекту у кабінеті користувача та зберігання особистих даних користувача таких як: логін, пароль, ім’я. СУБД завжди забезпечує незалежність даних. Будь-які зміни в механізмі зберігання виконуються без зміни всієї програми.

1.2.2 Система користувачів

Розроблена система користувачів включає в себе реєстрацію, автентифікацію та авторизацію.

Під час реєстрації створюється особистий обліковий запис у системі, з метою отримання доступу до повного функціоналу. У системі розташування камер зареєстрованим користувачам надається можливість зберігання проектів в особистому кабінеті та експорту результатів алгоритму.

Автентифікація це основа безпеки системи, яка полягає в перевірці достовірності даних про користувача сервером. Процес перевіряє правильність вводу користувачем логіну та паролю.

Під час процесу авторизації відбувається перевірка прав користувача і визначається можливість доступу в систему та в особистий кабінет.

1.2.3 Системи додавання вхідних даних користувачем

Система додавання вхідних даних користувачем надає можливість вивантаження вхідної інформації до системи для запуску алгоритму. Внесення плану приміщення, кількості камер та їх виду.

1.2.4 Система зчитування вхідної інформації

Для роботи алгоритму необхідне зчитування вивантаженої користувачем інформації. Система запускає алгоритм автоматичного розташування камер відеоспостереження у приміщеннях та подає на вхід план приміщення.

1.2.5 Система розташування камер на плані

Система аналізує план аналізує план приміщення та виділяє місця, в яких можуть бути розташовані камери [2]. Система генерує всі можливі положення для

кожного типу камери. Алгоритм розташування аналізує набір місць та положень камер, та обирає найоптимальніший набір камер відеоспостереження, що забезпечить максимальне покриття при мінімальних витратах на обладнання.

1.2.6 Система виведення результатів алгоритму

Система конвертує результат алгоритму автоматичного розташування камер відеоспостереження у зображення та надає можливість користувачу взаємодіяти з ним.

1.2.7 Система експорту вихідних даних програми

Система надає можливість експорту вихідних даних програми у форматах jpg та png, а також збереження цих даних у самій системі якщо користувач є попередньо авторизованим. При наступному вході в систему користувач має змогу швидко відтворити результати попередніх запусків алгоритму.

Висновки до розділу

У розділі Постановка задачі розробка системи автоматичного розташування камер відеоспостереження в приміщенні було розглянуто всі вимоги до системи, яка є веб-додатком. Описано задачі які повинно розв’язувати розроблене програмне забезпечення, розглянуто вхідні і вихідні дані системи. Було сформульовано перелік підсистем та чіткі вимоги до їх реалізації.

2. ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

Аналіз проблеми оптимального розташування камер у приміщенні, з максимальним покриттям та мінімальною кількості камер [3].

Автоматичне розміщення камер спостереження у довільних будівлях є складним завданням, а також грає важливу роль у ефективному використанні широкомасштабних систем нагляду. Існуючі підходи до автоматичного розміщення камер не задовольняють усі потреби користувачів [4]. Але майже всі системи мають один сценарій роботи. Користувач повинен спочатку інсталювати програмне забезпечення на свій комп'ютер або використати онлайн версію, якщо можливо. Далі потрібно активувати ліцензію чи здійснити оплату на використання системи на заданий період [5]. Наступний крок – авторизація та налаштування програмного забезпечення [6]. І після низки налаштувань користувач може приступати безпосередньо до проектування. Є можливість завантажити план приміщення або створити його в системі. Далі користувач отримує результат своєї роботи, вид результату залежить від набору функціональностей системи і може завантажити його.

Існуючі системи [7]:

- CAD5D;
- JVSG;
- VivoTek.

2.1 CAD5D

CAD5D це перший онлайн CAD в режимі 2D і 3D для проектувальників і будівельників. Включає в себе каталог товарів, матеріалів і устаткування. Розробники системи пропонують по-справжньому спростити розрахунок кошторисів і зробити дуже зручною роботу над проектом з колегами, замовниками і субпідрядниками.

Функції програмного забезпечення:

- Автоматична розстановка камер;

- Автопрокладання кабелів;
- Друк результатів на принтері або 3D принтері;
- Імпорт підкладок у векторному і растровому форматі;
- Візуалізація зон камер;
- Розрахунок ємності жорсткого диска і пропускної здатності;
- Можливість друку виділеної області;
- Експорт готового креслення в dwg, pdf, ifc;
- Перегляд 3D виду з будь-якої камери;
- Наявність великої бази обладнання beward rvi;
- Тривимірна візуалізація меж зон огляду камери;
- Робота з багаторівневими будівлями і спорудами;
- Проектування інтерфейсу оператора;

Переваги:

1. CAD5D — онлайн програма, тому при проектуванні інформаційної мережі, фахівець по електричних мережах може одночасно працювати з замовником на одному проекті, що істотно скоротить час на проектування;
2. Програма сама поставить PPPoE extender якщо довжина кабелю перевищує 90 метрів;
3. Програма сама встановить і розставить необхідну кількість камер, щоб не було «сліпих» зон;
4. У програмі CAD5D є можливість відразу надати замовнику проект із зазначенням зон спостереження;
5. Виключається людський фактор. Програма проводить всі розрахунки з ідеальною точністю;
6. CAD5D містить всю необхідну інформацію про використане обладнання. Дана інформація є достовірною, так як каталоги і технічний опис обладнання нам надає сам виробник.

Недоліки:

1. У базі виробників відеоспостереження представлені лише декілька виробників: Hikvision, Axis, Sony, Dahua, GeoVision. Якщо користувач має камери іншого бренду, то використання системи CAD5D неможливе;
2. Програмне забезпечення має перевантажений функціонал, та призначена для проектування не лише систем відео нагляду, а і освітлення, електрики, опалення, зв’язку та інших систем;
3. Використання системи з таким інтерфейсом як у CAD5D можливе лише при наявності спеціальних професійних навичок у користувача;
4. Запропоновані функції більше підходять для професійного використання, адже є потреба у багатьох замірах. Користувач, який вже має набір камер та бажання ефективно їх використовувати навряд буде звертатися до CAD5D;
5. При вартості системи 5.5 тисяч гривень на рік вона є найдорожчою на ринку.

2.2 JVSG

IP Video System Design Tool дозволяє швидко знайти оптимальну кількість і розташування камер відеоспостереження, виконати розрахунок системи відеоспостереження, оцінити довжину кабелів і відобразити на плані місцевості або приміщення зони ідентифікації, розпізнавання [8], детектування на основі щільності пікселів, змоделювати перешкоди в 2D і 3D для виявлення мертвих зон і надати замовнику професійно виглядаючий ескізний проект системи відеоспостереження, забезпечений результатами тривимірного моделювання. І все це можна зробити віддалено, не виїжджаючи на об'єкт.

Функції:

- Проектування в 2D;
- Моделювання в 3D;
- Реалістичні 3D-моделі;
- Функція прокладки кабелів;
- Додавання стін що обмежують зону камер;

- Моделювання Fisheye видів з камер;
- Візуальні ефекти, вкладка «Види з камер»;
- Завантаження карт Google Maps;
- Завантаження креслень Автокад у форматах dwg, dxf, dwf;
- Імпорт користувальницьких 3D моделей у форматах obj, dae.

Переваги:

1. Обґрунтування кількості камер за допомогою результатів 3D моделювання;
2. Економія часу і зниження ризику помилок за рахунок швидкого і наочного розрахунку областей видимості, кутів огляду і фокусних відстаней об'єктивів камер відеоспостереження. Краще відразу спроектувати систему охоронного телебачення правильно, ніж потім переробляти;
3. Легко підібрати необхідний дозвіл камери і максимальний кут огляду для цілей детекції, розпізнавання (125 пікселів на метр) та ідентифікації людей (250 пікселів на метр);
4. Розробник системи відеоспостереження може заздалегідь показати замовникам, що вийде в результаті установки системи відеоспостереження та узгодити з ними очікуваний результат;
5. Є можливість завантаження плану приміщень або карти місцевості для швидкого моделювання системи відеоспостереження в форматах JPEG, PDF, PNG. У версії Pro підтримується завантаження файлів у форматі DWG AutoCAD або DXF для інших популярних САПР;
6. Пропонується підвищення рівня захисту об'єкта за рахунок пошуку «сліпих зон» заздалегідь за допомогою 2D і 3D моделювання;
7. Можна швидко оцінити необхідну пропускну здатність мережі з будь-якою кількістю IP камер і відео серверів і підібрати оптимальну швидкість запису і рівень стиснення відео потоків;
8. Миттєвий розрахунок необхідного розміру жорстких дисків для зберігання відео архіву;

9. Функція прокладання кабелів і розрахунку їх довжини з допомогою програми;

10. Програма генерує проект в форматі PDF, а отримані в програмі таблиці, креслення, і результати тривимірного моделювання можуть бути легко перенесені в Word, Excel, OpenOffice, Visio і інші офісні програми.

Недоліки:

1. Відсутність web-версії. Це значно зменшує швидкість проектування, адже замовник та інженер не можуть редагувати проект одночасно віддалено один від одного та працювати онлайн;

2. Програмне забезпечення має застарілий та складний інтерфейс;

3. Система не призначена для використання у домашніх умовах;

4. Система має великий розмір та має високі вимоги до системних показників ПК;

5. Висока вартість системи. Одноразовий платіж сумою від 3.5 до 11.5 тисяч гривень в залежності від функціоналу.

2.3 VivoTek

За допомогою інструменту дизайну системи IP Video користувачі можуть обчислити точну фокусну відстань об'єктива та кути огляду всіх камер за секунди та перевірити поле зору кожної камери. Це дозволяє виявити будь-які мертві зони та підвищити безпеку приміщень за допомогою використання 2D та 3D моделювання. Цей розумний інструмент також може отримати більш точні оцінки пропускної здатності мережі, зберігання даних та виконати багато інших зручних функцій для покращення дизайну системи спостереження.

Функції:

- 2D планування для кожного поля зору камери;
- 3D-перегляди та моделювання;
- Реалістичні 3D-моделі.

Переваги:

1. Обчислення точної фокусної відстані об'єктива камери та кути огляду в секундах;

2. Можливість перевірки поля зору кожної камери та пошуку мертвих зон для підвищення рівня безпеки приміщень за допомогою 2D та 3D моделювання;

3. Оцінка необхідної пропускної здатності мережі для формування мережевих відео систем з будь-якою кількістю IP-камер та відео серверів;

4. Обчислення необхідного вільного простору на жорсткому диску для архіву відео;

5. Завантаження плану у різноманітних форматах зображень JPEG, JPG, BMP або PDF;

6. Можливість роздрукувати або експортувати готовий проект у PDF. Копіювання розрахунків, креслень та 3D макети, для створення відмінної проектної документації.

Недоліки:

1. Система інтегрована та використовується лише з відеокамерами бренду VivoTek;

2. Перевантажений інтерфейс;

3. Необхідність внесення великої кількості показників, які потрібно заміряти за допомогою спеціальних приладів;

4. Неможливість онлайн роботи з системою;

5. Програмне забезпечення розроблено для професійних інженерів та дизайнерів систем відео нагляду та не може бути використано користувачами без спеціальної освіти.

Висновки до розділу

Незважаючи на те, що в розглянутих системах більше переваг ніж недоліків можна зробити висновок, що наразі немає ідеальної системи. Тому кожен інженер, дизайнер системи відеоспостереження у приміщенні або механік-монтажер обирає для себе найбільш зручне програмне забезпечення, яке може вирішити поставлені

задачі. Проте користувачу, який має намір зекономити кошти та використовувати систему в домашніх умовах, обирати немає з чого. Описані вище системи не можуть функціонувати без професіонала з профільною освітою. Тобто пересічний користувач має наймати працівника, яких вміє працювати з такими системами чи може надати вхідні дані для них. Інженер повинен добре знати інтерфейс програми, робити спеціальні заміри, керувати проектом планування та слідкувати за точним виконанням плану.

Отже можна точно сказати що розробка системи автоматичного розташування камер відеоспостереження є доцільною та актуальною у наш час.

3. ЗАСОБИ РОЗРОБКИ

3.1 Програмні засоби

Matplotlib - це бібліотека для створення двовимірних графіків у Python. Хоча вона бере свій початок в емуляції графічних команд MATLAB, вона не залежить від MATLAB і може використовуватися в об'єктно-орієнтованому способі. Хоча Matplotlib написаний в основному чистим Python, він використовує NumPy та інший розширений код, щоб забезпечити хорошу продуктивність навіть для великих масивів.

Однією з найбільших переваг візуалізації є те, що вона дозволяє нам візуально отримати доступ до величезної кількості даних у легкозасвоюваній візуалізації. Matplotlib складається з декількох графіків, таких як лінія, смуга, гістограма тощо.

NumPy - це бібліотека для мови програмування Python, яка підтримує великі багатовимірні масиви та матриці, а також велику колекцію математичних функцій високого рівня для роботи над цими масивами.

NumPy - це основний пакет, необхідний для наукових обчислень з Python. Цей пакет містить:

- Потужний об'єкт N-розмірного масиву;
- Складні (мовні) функції;
- Основні функції лінійної алгебри;
- Основні перетворення Фур'є;
- Складні можливості випадкових чисел;
- Інструменти для інтеграції коду Fortran;
- Інструменти для інтеграції коду C / C ++.

Крім очевидних наукових застосувань, NumPy також може бути використаний як ефективний багатовимірний контейнер із загальними даними. Можна визначити довільні типи даних. Це дозволяє NumPy легко та швидко інтегруватись із найрізноманітнішими базами даних.

NumPy є наступником двох попередніх наукових бібліотек Python: Numeric та Numarray.

Основною функціональністю NumPy є "ndarray", для n-мірного масиву, структури даних. На противагу вбудованій структурі списку даних Python, ці масиви є однорідними: всі елементи одного масиву повинні бути одного типу.

Такі масиви можуть також переглядати буфери пам'яті, що виділяються розширеннями C / C ++, Cython та Fortran в інтерпретатор CPython без необхідності копіювати дані, надаючи ступінь сумісності з існуючими бібліотеками чисел.

У розробленій системі NumPy відповідає за математичні обчислення, перетворення матриць. Розбиває полігон на набір 0 і -1 (перешкоди і стіни) та 1 – камери.

Shapely - це пакет Python для аналізу та маніпуляції з площинними ознаками, використовуючи (через модуль ctypes Python) функції з добре відомої та широко розгорнутої бібліотеки GEOS. GEOS, порт пакету Java Topology Suite (JTS), є двигуном геометрії просторового розширення PostGIS для RDBMS PostgreSQL.. Таким чином, Shapely глибоко укорінений у конвенціях світу географічних інформаційних систем (ГІС), але прагне бути не менш корисним для програмістів, які працюють над нетрадиційними проблемами. У розробленій системі дана бібліотека відповідає за перетворення геометричних фігур на матриці і навпаки, використання полігонів в пайтон коді. Приймає на вхід 4 координати точки та видає матрицю.
Descartes дозволяє візуалізувати shapely в matplotlib.

3.2 Середовище розробки

PyCharm — це інтегроване середовище розробки (IDE), яке використовується в комп'ютерному програмуванні, спеціально для мови Python. Він розроблений чеською компанією JetBrains. Він забезпечує аналіз коду, графічний налагоджувач, інтегрований тестер одиниць, інтеграцію з системами управління версіями (VCS) та підтримує веб-розробки з Django, а також Data Science з Anaconda. PyCharm – кросплатформенний, з версіями Windows, macOS та Linux.

3.3 Мови програмування

Python

Python — інтерпретована, об'єктно-орієнтована мова програмування високого рівня з динамічною семантикою. Високорівневі вбудовані структури даних у поєднанні з динамічним набором тексту та динамічним зв'язуванням роблять її дуже привабливою для швидкого розвитку додатків, а також для використання в якості мови сценаріїв або клею для з'єднання існуючих компонентів разом. Простий, легкий у вивченні синтаксис Python [9] підкреслює читабельність і, таким чином, зменшує витрати на обслуговування програми. Python підтримує модулі та пакети, що заохочує модульність програми та повторне використання коду. Інтерпретатор Python та велика стандартна бібліотека доступні у вихідній чи двійковій формі безкоштовно за всіма основними платформами, і вони можуть вільно поширюватися.

Часто програмісти обирають в Python через підвищену продуктивність, яку він забезпечує. Оскільки кроку компіляції немає, цикл редагування-тестування налагодження надзвичайно швидкий. Налагодження програм Python легке: помилка або неправильний вклад ніколи не спричинить помилку сегментації. Натомість, коли перекладач виявляє помилку, він створює виняток. Коли програма не вловлює виняток, інтерпретатор виводить слід стека. Налагоджувач рівня джерела дозволяє перевіряти локальні та глобальні змінні, оцінювати довільні вирази, встановлювати точки прориву, переходити через код рядок за часом тощо. Налагоджувач написаний самим Python, свідчить про інтроспективну силу Python. З іншого боку, часто найшвидшим способом налагодження програми є додавання декількох тверджень про друк до джерела: швидкий цикл редагування-тестування-налагодження робить цей простий підхід дуже ефективним.

JavaScript

JavaScript (JS) — це легка, інтерпретована, об'єктно-орієнтована мова з першокласними функціями, і найвідоміша як мова сценаріїв для веб-сторінок, але вона також використовується у багатьох середовищах, що не належать до браузера.

Заснована на прототипі, парадигмальна скриптова мова, яка є динамічною і підтримує об'єктно-орієнтований, імперативний та функціональний стилі програмування.

JavaScript працює на клієнтській стороні Інтернету, який може бути використаний для проектування чи програмування поведінки веб-сторінок у разі події. JavaScript — це легка у вивченні, а також потужна мови, широко використовується для контролю поведінки веб-сторінок.

JavaScript може функціонувати як процедурна, і як об'єктно-орієнтована мова. Об'єкти створюються програмно в JavaScript, шляхом приєднання методів та властивостей до інших порожніх об'єктів під час виконання, на відміну від синтаксичних визначень класу, поширених у складених мовах, таких як C ++ та Java. Після побудови об'єкта він може бути використаний як шаблон (або прототип) для створення подібних об'єктів.

Динамічні можливості JavaScript включають побудову об'єктів часу виконання, списки змінних параметрів, змінні функцій, створення динамічного сценарію, інтроспекцію об'єкта та відновлення вихідного коду.

3.4 Мови розмітки та стилів

HTML

HTML розшифровується як мова розмітки гіпертексту. Це дозволяє користувачеві створювати та структурувати розділи, абзаци, заголовки, посилання для веб-сторінок та додатків.

HTML не є мовою програмування, тобто не має можливості створювати динамічну функціональність. Натомість це дозволяє організувати та форматувати документи, подібно до Microsoft Word.

Працюючи з HTML, ми використовуємо прості кодові структури (теги та атрибути) для розмітки сторінки веб-сайту. Наприклад, ми можемо створити абзац, розмістивши доданий текст у початковому тезі та закривши .

Загалом, HTML — це мова розмітки, яку справді просто та легко вивчити навіть для початківців у створенні веб-сайтів.

CSS

Каскадні таблиці стилів, які називають CSS, — це проста мова дизайну, призначена для спрощення процесу прикрашання веб-сторінки.

CSS обробляє зовнішній вигляд веб-сторінки. Використовуючи CSS, можна керувати кольором тексту, стилем шрифтів, інтервалом між абзацами, розмірами та розміщенням стовпців, якими фоновими зображеннями та кольорами, оформленням макетів, варіаціями відображення для різних пристроїв та розмірів екрана а також інші різноманітні ефекти.

CSS легко вивчити та зрозуміти, але він забезпечує потужний контроль над поданням HTML-документа. Найчастіше CSS поєднується з мовами розмітки HTML або XHTML.

CSS створюється та підтримується через групу людей у рамках W3C, що називається Робоча група CSS. Робоча група CSS створює документи, що називаються специфікаціями. Коли специфікація обговорена та офіційно ратифікована членами W3C, вона стає рекомендацією.

Ці ратифіковані специфікації називаються рекомендаціями, оскільки W3C не контролює фактичну реалізацію мови. Незалежні компанії та організації створюють це програмне забезпечення.

Всесвітній веб-консорціум або W3C — це група, яка дає рекомендації щодо того, як працює Інтернет та як він має розвиватися.

3.5 Frameworks

Django — це веб-система Python високого рівня, яка дозволяє швидко розробити безпечні та бездоганні веб-сайти. Побудований досвідченими розробниками, Django піклується про багато клопоту веб-розробки, тому розробник може зосередитись на написанні програми, не потребуючи винаходити колесо. Це безкоштовний та відкритий код, має процвітаючу та активну спільноту, чудову документацію та безліч варіантів безкоштовної та платної підтримки.

Django допомагає писати програмне забезпечення, яке:

Повне. Django дотримується філософії "Усе включено" і забезпечує майже все, що розробники можуть зробити "поза коробкою". Оскільки все, що потрібно, є частиною одного "продукту", це все працює безперебійно, дотримується послідовних принципів дизайну та має велику і сучасну документацію.

Універсальне. Django можна використовувати для створення майже будь-якого типу систем та веб-сайтів — від систем управління контентом та вікі-сайтів, до соціальних мереж та новинних сайтів. Він може працювати з будь-якою базою клієнта і може доставляти вміст майже в будь-якому форматі (включаючи HTML, RSS-канали, JSON, XML тощо). Хоча він надає вибір практично для будь-якого функціоналу, (наприклад, декількох популярних баз даних, двигунів для шаблонів тощо), він також може бути розширений для використання інших компонентів, якщо це необхідно.

Безпечне. Django допомагає розробникам уникати багатьох поширених помилок безпеки, надаючи рамки, розроблені для "правильних дій" для автоматичного захисту веб-сайту. Наприклад, Django забезпечує безпечний спосіб керувати обліковими записами та паролями користувачів, уникаючи поширених помилок, таких як розміщення інформації про сеанс у файлах cookie там, де вона вразлива (натомість файли cookie просто містять ключ, а фактичні дані зберігаються в базі даних) або безпосередньо зберігають паролі а не хеш паролів.

Хеш паролі — це значення фіксованої довжини, створене надсиланням пароля через функцію криптографічного хеша. Django може перевірити правильність введеного пароля, запустивши його через хеш-функцію та порівнявши вихід із збереженим значенням хеша. Однак через "односторонній" характер функції, навіть якщо збережене значення хешу порушено, зловмиснику важко розробити оригінальний пароль. Django забезпечує захист від багатьох уразливостей за замовчуванням, включаючи ін'єкцію SQL, скрипти, підробку запиту на різних веб-сайтах та перехід на клік.

Масштабоване. Django використовує компонентну архітектуру (кожна частина архітектури не залежить від інших, і тому може бути замінена або змінена за потреби). Чітке розділення між різними частинами означає, що він може масштабувати

збільшення трафіку, додаючи апаратне забезпечення на будь-якому рівні: кешування серверів, серверів баз даних або серверів додатків. Деякі із найзайнятіших сайтів успішно розширювали масштаб Django, щоб задовольнити їхні вимоги.

Кросс-платформне. Django написаний на Python, який працює на багатьох платформах. Це означає, що ви не прив'язані до будь-якої конкретної серверної платформи і можете запускати свої програми на багатьох варіантах Linux, Windows та Mac OS X. Крім того, Django добре підтримується багатьма постачальниками веб-хостингів, які часто надають конкретну інфраструктуру та документація для розміщення сайтів Django.

3.6 Системи управління базами даних

MySQL - це система управління реляційними базами даних з відкритим кодом, що базується на структурованій мові запитів (SQL). MySQL доступний у всіх основних операційних системах, включаючи Windows, Linux тощо.

MySQL, як і інші реляційні бази даних, зберігає дані в таблицях, стовпцях та рядках. Кожен запис визначається унікальним ідентифікатором. Її основою завжди були продуктивність та надійність бази даних. MySQL була розроблена та оптимізований для арени веб-розробок; це, мабуть, найпоширеніша база даних, яка використовується при розгортанні веб-серверів. MySQL дуже добре працює з Apache та PHP і часто переходить до бази даних для розгортання стеків LAMP. Сьогодні MySQL діє 9 з 10 веб-сайтів в Інтернеті.

3.7 Програмне забезпечення ПК

Вимоги до програмного забезпечення ПК:

- Операційна система MacOS 10.15.0 та всі наступні версії;
- Програмне забезпечення яке автоматизує розгортання застосунку у середовищах, що підтримують контейнеризацію Docker;
- Не менше восьми гігабайтів оперативної пам'яті;

- Інтерпретатор Python 3.8;
- Сервер додатку Gunicorn.

Docker

Docker — це програмна платформа для створення додатків на основі контейнерів — невеликих та легких середовищ виконання, які спільно використовують ядро операційної системи, але в іншому випадку працюють у відриві один від одного. Незважаючи на те, що контейнери як концепція існують вже деякий час, Докер, проект з відкритим кодом, запущений у 2013 році, допоміг популяризувати технології та допоміг просунути тенденцію до контейнізації та мікросервісів у розробці програмного забезпечення, що стало відомим як хмарний розвиток.

Що таке контейнери? Однією з цілей сучасної розробки програмного забезпечення є тримати додатки на одному хості або кластері ізольовано один від одного, щоб вони не надто перешкоджали роботі або обслуговуванню один одного. Це може бути складно, завдяки пакетам, бібліотекам та іншим програмним компонентам, необхідним для їх роботи. Одним з варіантів вирішення цієї проблеми були віртуальні машини, які підтримують додатки на одному і тому ж апаратному забезпеченні повністю відокремленими і зводять до мінімуму конфлікти між компонентами програмного забезпечення та конкуренцію за апаратні ресурси. Але віртуальні машини є громіздкими — кожна вимагає власної ОС, тому типово розміром гігабайт є складна в обслуговуванні та модернізації.

Контейнери ізолюють середовища виконання програм одне від одного, але поділяють базове ядро ОС. Зазвичай вони вимірюються в мегабайтах, використовують набагато менше ресурсів, ніж VM, і запускаються практично негайно. Вони можуть працювати із значно меншими зусиллями та витратами. Контейнери забезпечують високоефективний механізм для комбінування програмних компонентів у типи степів додатків та сервісів, необхідних на сучасному підприємстві, та для постійного оновлення та обслуговування цих компонентів програмного забезпечення.

Docker — це проект з відкритим кодом, який дозволяє легко створювати контейнери та додатки на основі контейнерів. Спочатку побудований для Linux, тепер Docker також працює на Windows та MacOS. Щоб зрозуміти, як працює Docker, давайте розглянемо деякі компоненти, які ви б використали для створення додатків, що містять контейнери Docker.

Gunicorn

Gunicorn — сервер The Web Server Gateway Interface (WSGI). Gunicorn побудований так, що багато різних веб-серверів можуть взаємодіяти з ним. Також не дуже важливо, що ви використовували для створення свого веб-додатка — доки він може взаємодіяти з використанням інтерфейсу WSGI.

Gunicorn піклується про все, що відбувається між веб-сервером та веб-додатком. Таким чином, кодуючи додаток Django, не потрібно шукати власні рішення для:

- Спілкування з декількома веб-серверами;
- Реагування одразу на велику кількість веб-запитів та розподілення навантаження;
- Ведення кількох процесів роботи веб-програми.

Висновки до розділу

У розділі було розглянуто засоби розробки системи автоматичного розташування камер відеоспостереження у приміщенні. Описано програмні засоби, мови програмування, бібліотеки, мови розмітки та стилів, фреймворки та середовище розробки, вимоги до програмного забезпечення комп’ютера та систему управління базами даних. У розділі підтверджено доцільність використання саме цих засобів для розробки сучасної системи веб-додатку.

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Система автоматичного розташування камер відеоспостереження у приміщенні буде складатися з п’яти модулів, кожен з яких буде розбиватися на функціональні підблоки, загальна кількість яких п’ять. Головним модулем системи буде алгоритм автоматичного розташування камер.

На рисунку 4.1 наведена схема структури системи, на якій розташовані всі програмні модулі.

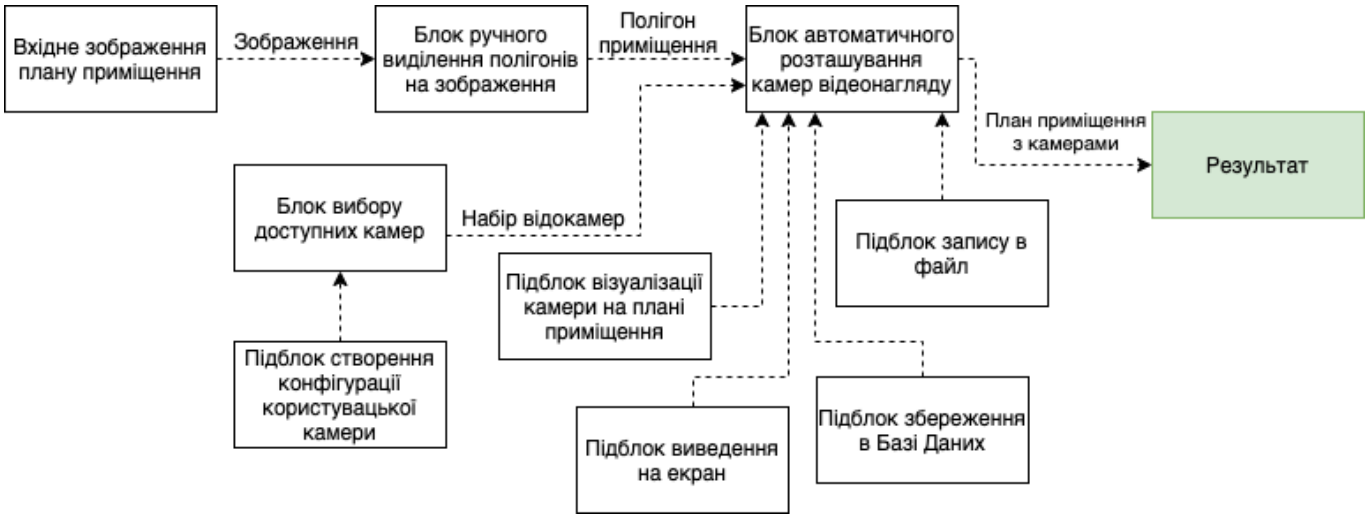

Рисунок 4.1 — Схема структури системи

Для кожного з необхідних модулів та задач системи було проаналізовано наявні варіанти вирішення. Серед них було обрано ті, що будуть задовольняти наступні умови:

- Безперешкодна інтеграція підсистем між собою;
- Швидкодія;
- Можливість оптимізації під конкретну область.

Наприклад для створення модуля додавання вхідних даних користувачем та підсистеми, що відповідає за зчитування вхідної інформації було розроблено алгоритм що зчитує виділені користувачем точки областей плану приміщення та перешкод на початковому фото плану приміщення.

Алгоритм створює цифрове відображення плану приміщення з усіма перешкодами шляхом передачі даних між стандартним компонентом, що обробляє вхідні дані від користувача та модулем створення полігонів з цих даних для подальшого аналізу можливих точок розміщення камер відеоспостереження.

Для вирішення задачі економії коштів користувача було розроблено два режими роботи системи:

- Максимальне покриття при заданому бюджеті або кількості камер;
- Мінімальна вартість для повного покриття.

Виходячи з кінцевих цілей користувач обирає необхідний режим роботи алгоритму.

Мобільність алгоритму була досягнута за допомогою обраного підходу розробки SPA Система є веб-додатком до якої користувачі мають постійний он-лайн доступ.

4.1 Опис функціональності системи

Опис роботи користувача з системою

Програмний додаток для розташування камер у приміщенні містить у собі двох акторів: гість — неавторизований користувач системи, користувач — авторизований.

У Уніфікованій мові моделювання (UML) діаграма прецедентів може узагальнити деталі користувачів вашої системи (також відомих як актори) та їх взаємодії з системою.

Щоб побудувати її потрібно використовувати набір спеціалізованих символів та роз'ємів. Коректно побудована діаграма допоможе представити:

- Сценарії, в яких система чи додаток взаємодіють з людьми, організаціями чи зовнішніми системами;
- Цілі, які система чи додаток допомагає досягти цим організаціям (відомим як актори);
- Обсяг системи.

Діаграма прецедентів не вкладається в багато деталей - наприклад, не змоделює порядок виконання кроків. Натомість діаграма прецедентів належного використання зображує огляд взаємозв'язку між випадками використання, акторами та системами на високому рівні.

UML - це інструментарій моделювання [10], який можна використовувати для побудови діаграм. Корпуси для використання представлені з міткою овальної форми. Фігурні палички представляють дійових осіб у процесі, а участь актора в системі моделюється лінією між актором та випадком використання. Щоб зобразити межу системи, потрібно намалювати поле навколо самого випадку використання.

Загальні компоненти діаграми прецедентів:

Актори: Користувачі, які взаємодіють із системою. Актором може бути людина, організація або зовнішня система, яка взаємодіє з програмою чи системою. Вони повинні бути зовнішніми об'єктами, які виробляють або споживають дані.

Система: конкретна послідовність дій та взаємодій між суб'єктами та системою. Система може також називатися сценарієм.

Цілі: кінцевий результат більшості випадків використання. Успішна діаграма повинна описувати діяльність та варіанти, які використовуються для досягнення мети.

Кейси використання: овали в горизонтальній формі, які представляють різні види використання, які може мати користувач.

Асоціації: лінія між учасниками та випадки використання. У складних діаграмах важливо знати, з якими акторами пов’язані дані випадки використання.

Системні граничні поля: Коробка, яка встановлює область системи для використання справ. Усі випадки використання поза межами поля розглядаються за межами цієї системи.

На рисунку 4.2 зображено діаграму прецедентів для розробленої системи автоматичного розташування камер відеоспостереження у приміщенні.

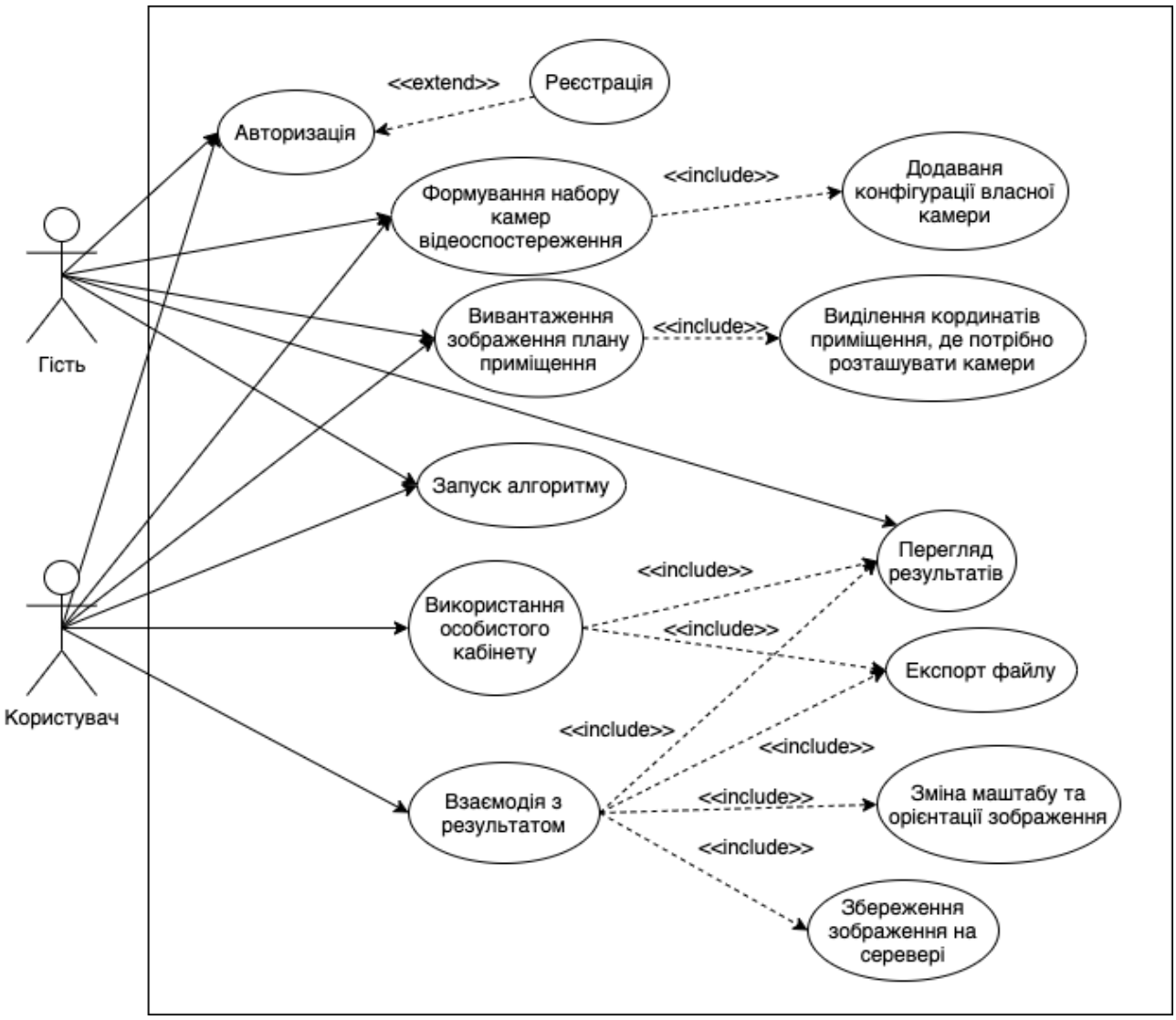

Рисунок 4.2 — Діаграма прецедентів

Діаграми послідовності UML - це діаграми взаємодії, які детально описують спосіб виконання операцій. Вони фіксують взаємодію між об'єктами в контексті співпраці. Діаграми послідовності орієнтовані на час, і вони візуально показують порядок взаємодії, використовуючи вертикальну вісь діаграми для подання часу, які повідомлення надсилаються та коли.

Діаграми послідовності фіксує:

- взаємодії, які відбуваються у співпраці, які реалізують або випадок використання, або операцію;
- взаємодії високого рівня між користувачем системи та системою, між системою та іншими системами або між підсистемами.

На рисунку 4.3 зображено діаграму послідовностей для системи автоматичного розташування камер відеоспостереження.

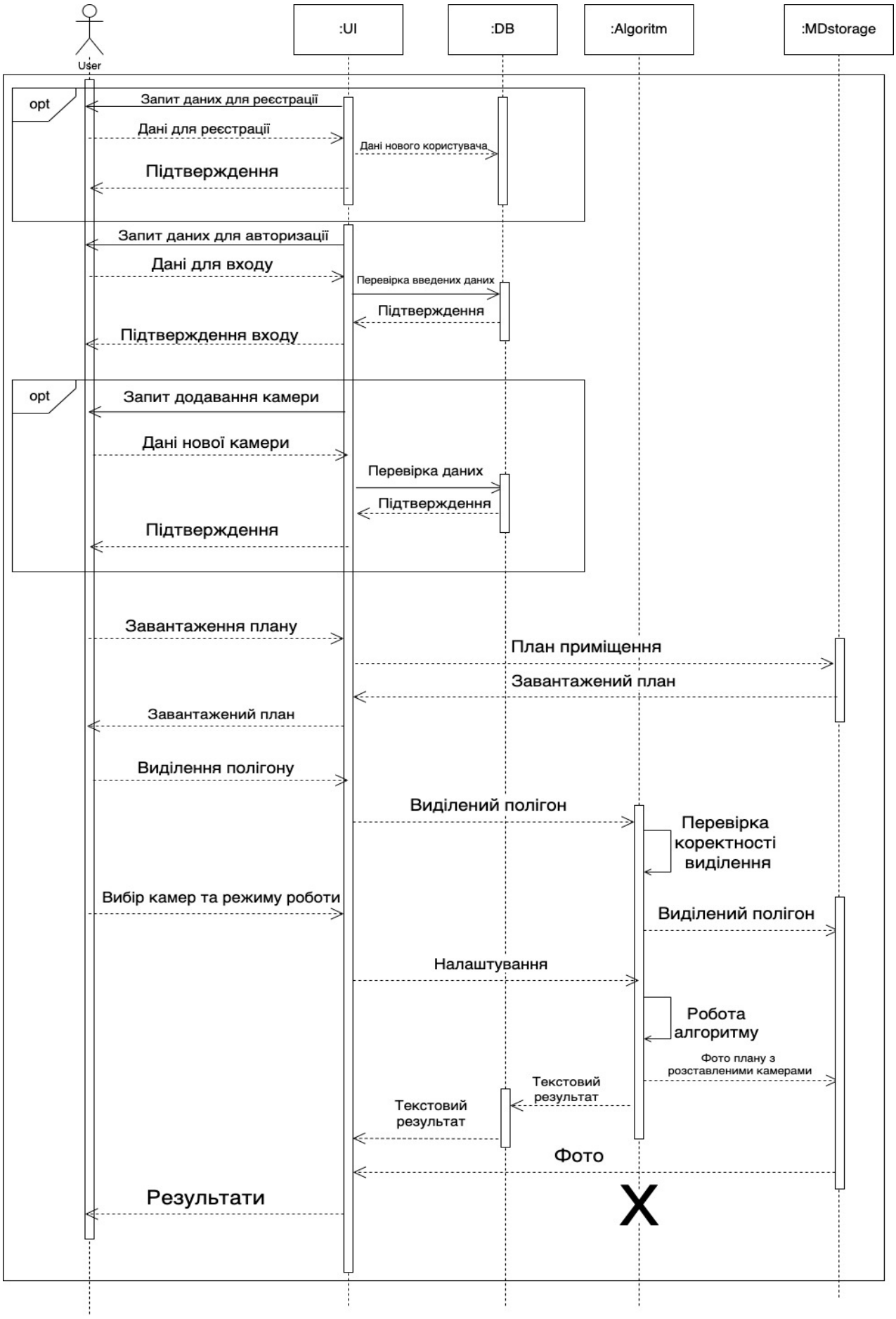

Рисунок 4.3 – Діаграма послідовностей

4.2 Послідовність роботи алгоритму

Першим кроком є завантаження фото плану приміщення. Далі за допомогою скрипта мовою JS користувач виділяє полігони тієї частини приміщення, де є намір розставити камери. Це зроблено для того, щоб зробити план більш чітким, адже завантажене користувачем фото не завжди якісне та часто має поміхи. Алгоритм використовує виділену частину фото. Полігон — описання зображення за допомогою координат. Далі користувач обирає камери з різними характеристиками (кут зору, дальність) з набору і/або може додати свої налаштування для камер, проте вони не зберігаються для всіх користувачів. На наступному кроці алгоритм розділяє план приміщення на grid (сітка блоків) і визначає всі можливі точки для розташування камер [11]. Далі робить набір всіх можливих положень для кожної камери у кожній з цих точок (камера може бути направлена вгору, вниз тощо). Потім алгоритм обирає найкращий набір який забезпечує найбільше покриття [12]. Наступним кроком є виведенням результату системою. Результатом є зображення плану приміщення з розставленими камерами. Далі користувач може експортувати зображення на свій пристрій, або змінити параметри камер чи план та повторно запустити алгоритм розстановки. Після закінчення роботи користувача з системою, результат зберігається в особистий кабінет користувача, якщо він має обліковий запис.

Авторизовані користувачі мають змогу переглянути свої попередні проекти в особистому кабінеті та завантажити їх.

На рисунку 4.4 зображено не деталізовану схему алгоритму яка була описана вище.

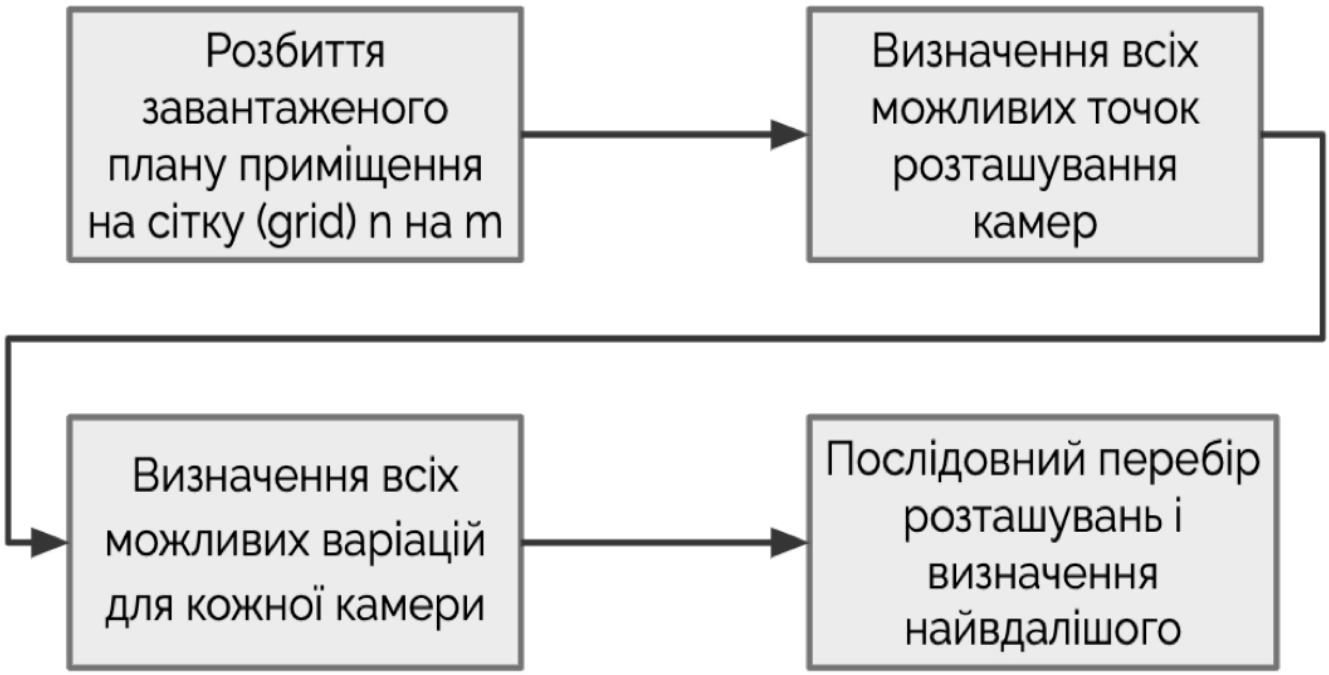

Рисунок 4.4 – Схема алгоритму

4.3 Архітектура додатку

Спілкування користувача з системою відбувається за моделлю клієнт — сервер. Тобто сервер не може надсилати ніякі дані клієнту без попереднього запиту. Сервер не зберігає ніяких проміжних даних про клієнта, при кожному запиті клієнт має передавати повну інформацію про себе [13]. Це дозволяє системі буди незалежною від кількості серверів, придатною до швидкого масштабування та гнучкою.

Сервер відповідає за бізнес-логіку додатку. Він не зберігає даних, для цього система під’єднана до SQL бази даних. Алгоритми обробки даних також зберігаються на сервері, ще одним компонентом сервера є система автентифікації.

Додаток має архітектуру моноліту. Це означає що всі модулі додатку тісно пов’язані між собою [14]. Через це програма містить менше коду, але це унеможливлює розділення її на окремі компоненти. Цей підхід використовують в роботі у невеликих командах.

Рисунок 4.5 — Клієнт-серверна архітектура

MVT (Model View Template) - це модель дизайну програмного забезпечення. Це колекція трьох важливих компонентів Перегляд, Модель та Шаблон.

Модель допомагає обробляти базу даних. Це рівень доступу до даних, який обробляє дані.

Шаблон - це презентаційний шар, який повністю обробляє частину інтерфейсу користувача. Перегляд використовується для виконання бізнес-логіки та взаємодії з моделлю для перенесення даних та візуалізації шаблону.

Немає окремого контролера і повний додаток заснований на перегляді, моделі та шаблоні. Ось чому його називають додатком MVT.

На рисунку деталізовано серверну архітектуру MVT фрейморку Django на основі якого побудовано систему автоматичного розташування камер

відеоспостереження у приміщенні. Рисунок показує MVT на основі керуючого потоку. Тут користувач запитує ресурс на Django, Django працює як контролер і перевіряє наявний ресурс у URL. Якщо URL-карти відображаються, вигляд, який взаємодіє з моделлю та шаблоном, відображає шаблон. Джанго відповідає користувачеві і надсилає шаблон як відповідь.

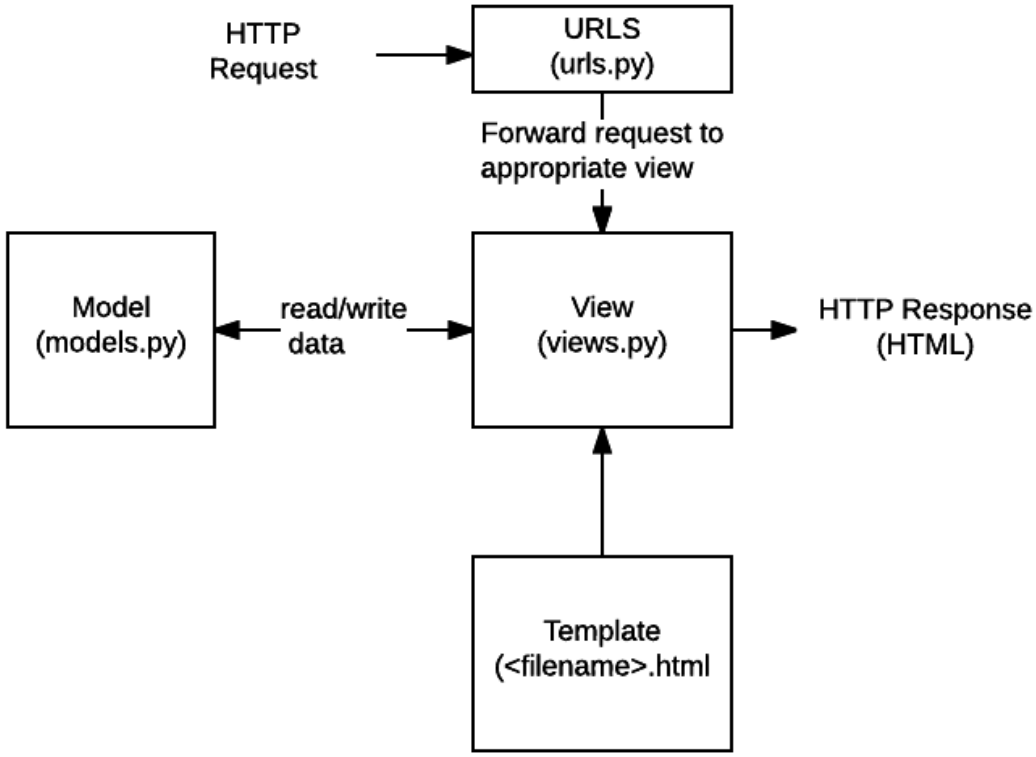

Рисунок 4.6 - Архітектура MBT

Найбільш зрілим підходом в розробці програмного забезпечення є монолітний архітектурний стиль проектування ПО. В рамках даного стилю першочерговим завданням стає розробка фреймворків, або бібліотек, які досить легко адаптуються до всіх потенційних вимог всіх потенційних замовників. Поставлене завдання полягає в розробці web-системи. Систему побудовано на принципах клієнт-серверної архітектури. При цьому додаток підтримує як запити, що оновлюють всю інформацію, що відображається в web-браузері, так і запити, що оновлюють лише частину інформації. Для цього використовується підхід SPA.

Додаток на одній сторінці (SPA) - це додаток, який працює в браузері і не перезавантажує сторінку під час роботи. Як і будь-який інший додаток, він завантажений для того, щоб допомогти користувачеві у вирішити поставлену перед

ним. У SPA можливо використати будь-які серверні технології. Наявна значуща частина веб-додатків переміщається у браузері, вимоги до серверів можна істотно ослабити. На рисунку 4.7 представлено різницю між підходами при реалізації звичайного веб-сайту та додатків на одній сторінці. На схемі видно, що в SPA основна частина роботи з даними переходить із сервером на клієнт.

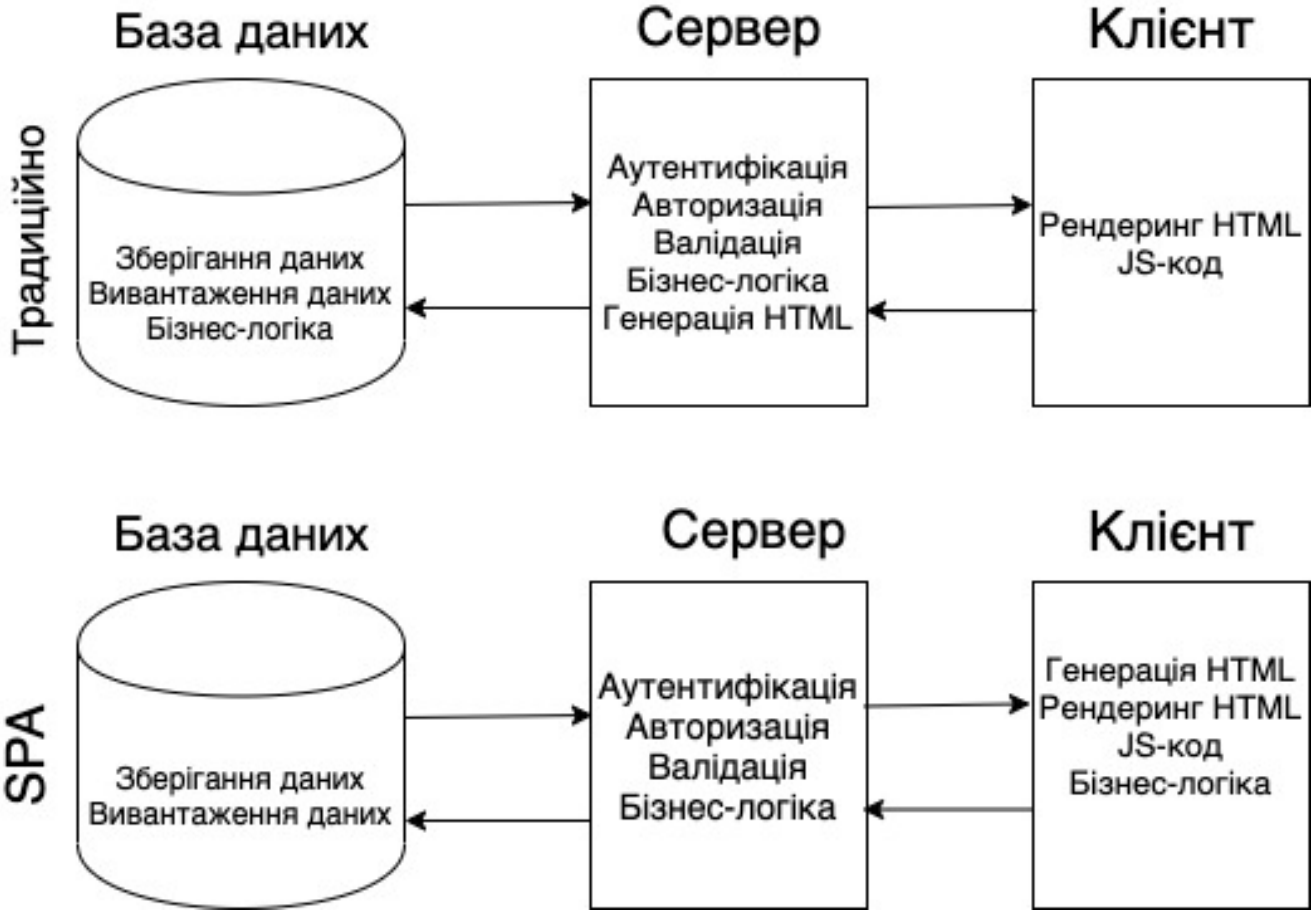

Рисунок 4.7 – Різниця SPA и традиційного підходу

Висновки до розділу

У розділі 4 було розглянуто програмну реалізацію системи автоматичного розташування камер відеоспостереження. Структура системи, приклад розв’язання поставлених задач, архітектура додатку, послідовність роботи алгоритму, діаграма прецедентів та послідовностей.

5. РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Операційна система у користувача має бути MacOS 10.15.0 і усі наступні, Windows 10, Linux Ubuntu 18.04. Будь-який браузер з підтримкою сучасних веб-технологій. Інсталяції додаток не потребує, адже це онлайн система.

5.1 Інструкція по роботі з системою

1. Авторизація користувача. Вхід в систему для зареєстрованих користувачів відбувається за допомогою введення логіну та паролю. На рисунку 5.2.1 представлено вигляд вікна входу. Якщо користувач вперше на сайті, він може створити обліковий запис, для цього необхідно ввести ім’я, електронну пошту та пароль. На рисунку 5.2.2 зображено вікно реєстрації.

Якщо користувач не пам’ятає пароль, він має змогу відновити його за допомогою пошти, рисунок 5.2.3, рисунок 5.2.4.

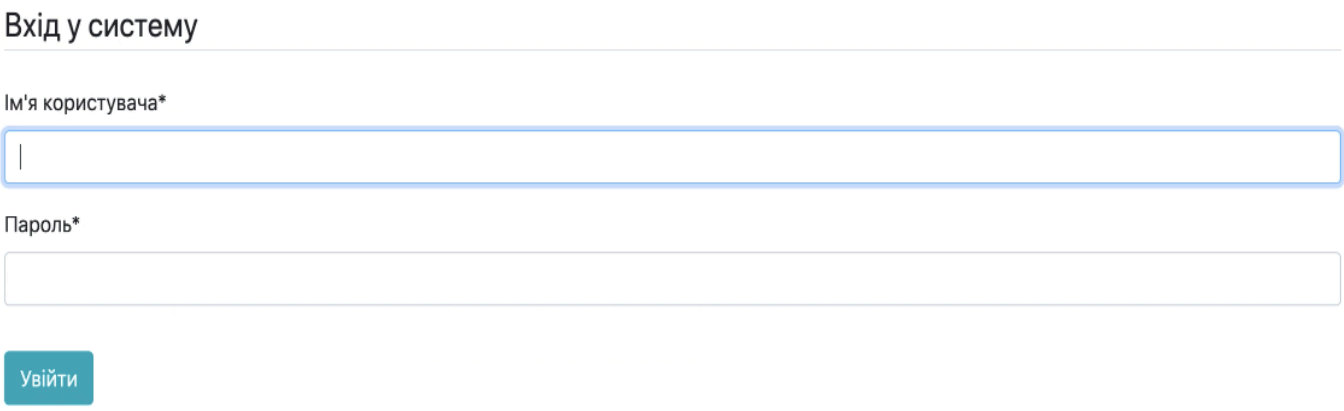

Рисунок 5.2.1 — Вікно входу в систему

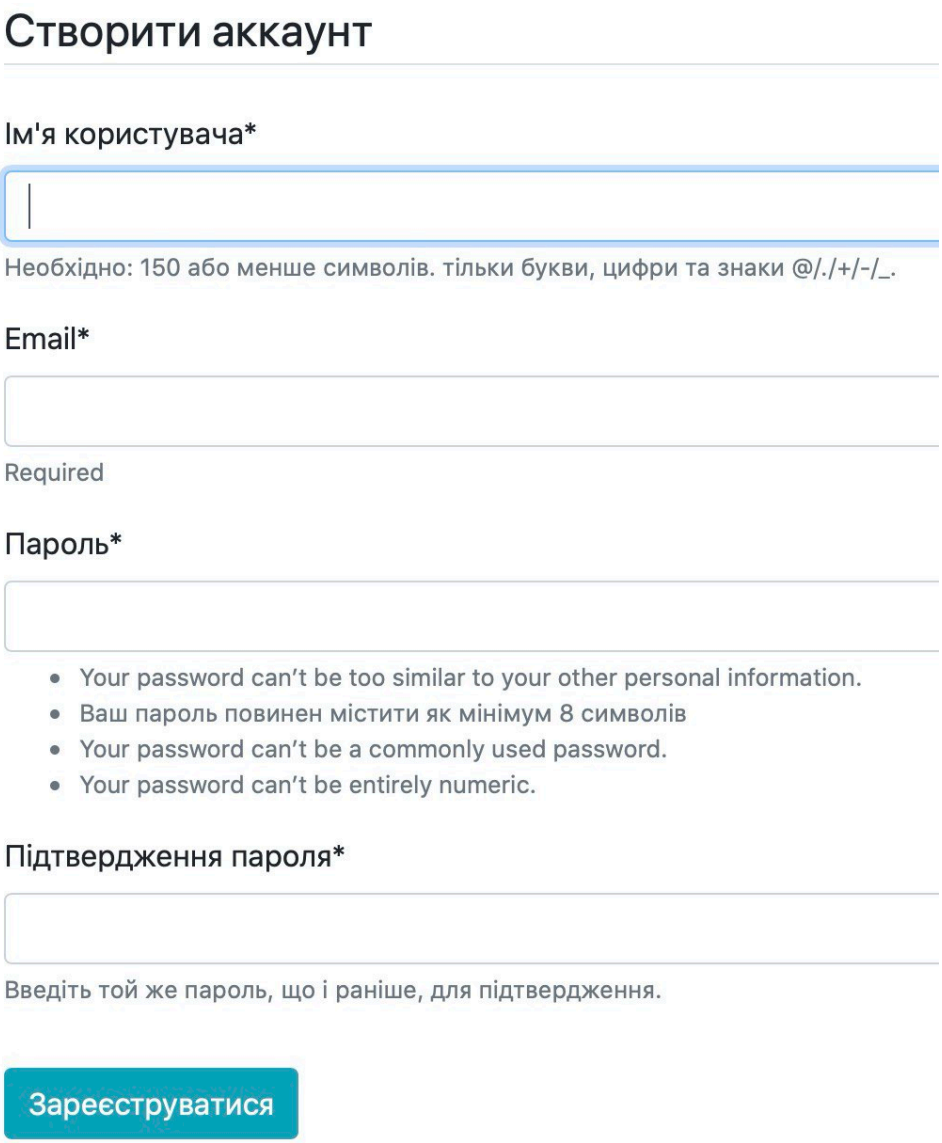

Рисунок 5.2.2 — Вікно реєстрації

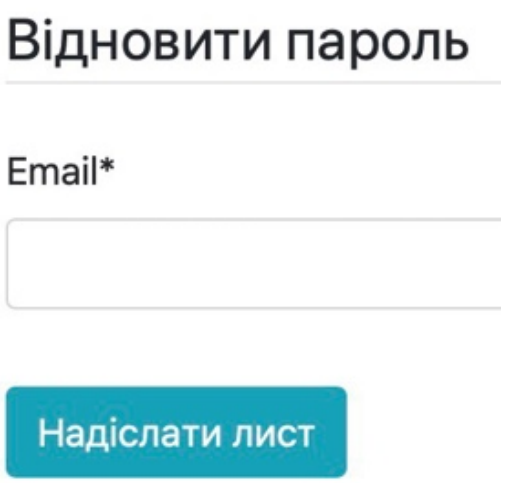

Рисунок 5.2.3 — Вікно відновлення паролю

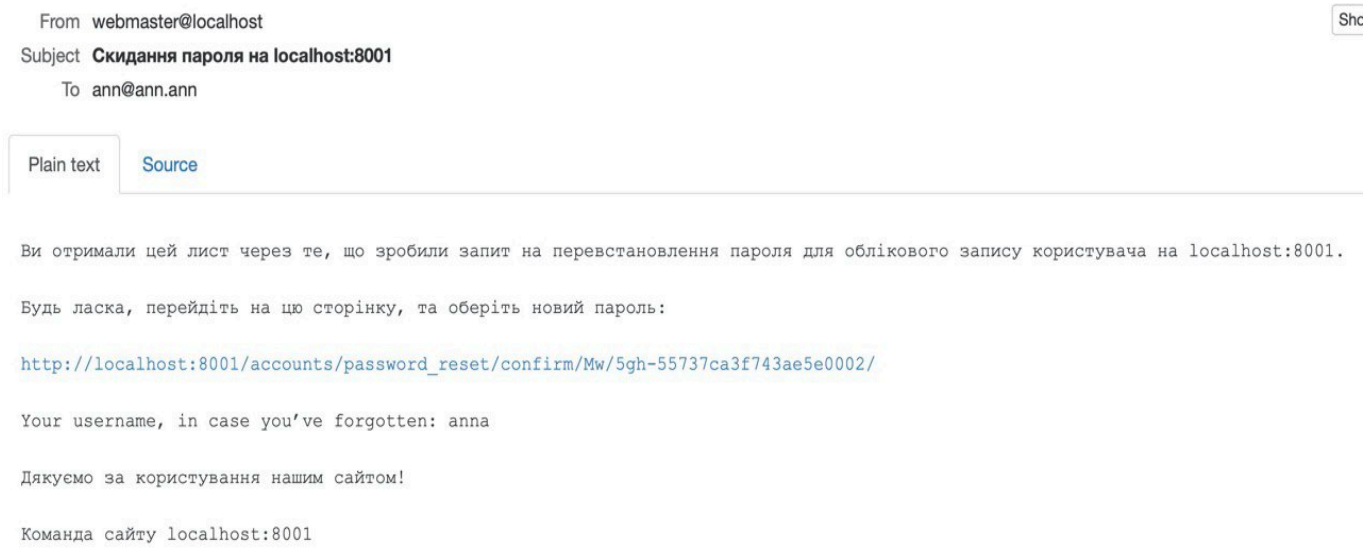

Рисунок 5.2.4 — Лист про відновлення паролю

2. Головний екран системи. Будь-який користувач має змогу переглянути інформацію про систему на головній сторінці, рисунок 5.2.5.

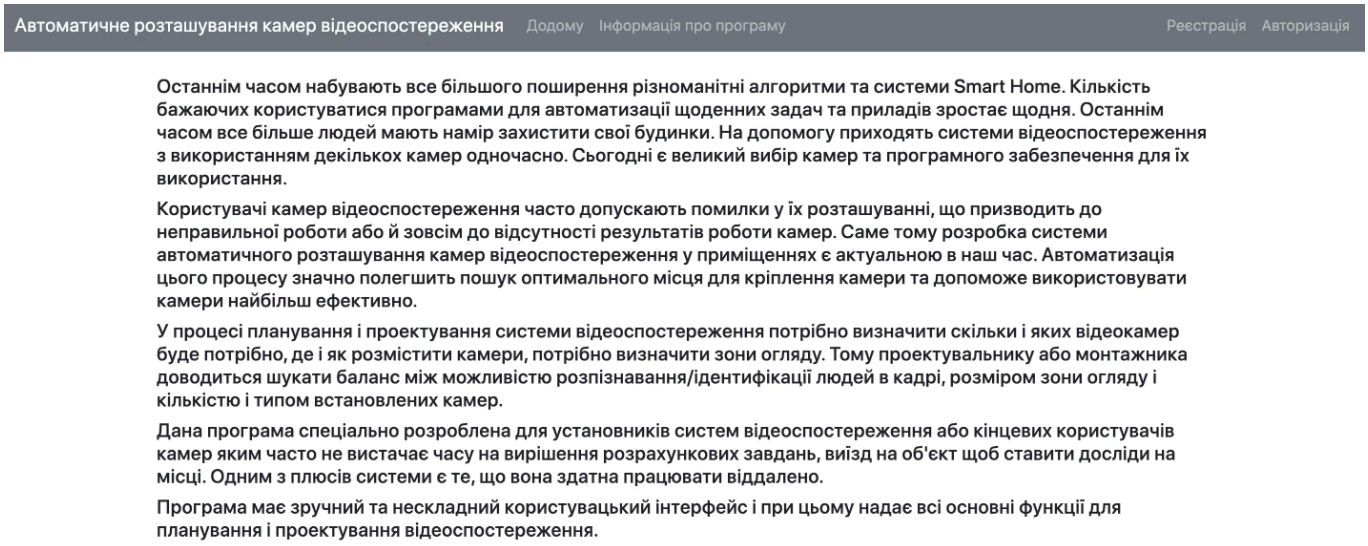

Рисунок 5.2.5 — Інформація про систему

3. Запуск алгоритму. Користувач має змогу переглянути наявні в системі камери, а також їх характеристики, рисунок 5.2.6. Попередньо авторизований

користувач має можливість додати власний тип камер з будь-якими характеристиками, рисунок 5.2.7. Для початку роботи алгоритму користувач повинен завантажити план приміщення в якому необхідно зробити розстановку камер, рисунок 5.2.8. На рисунку 5.2.9 зображено вигляд плану в інтерфейсі системи. Після завантаження користувач повинен виділити ту область плану (полігон), з якою буде працювати алгоритм, рисунок 5.2.10, та перешкоди, рисунок 5.2.10. Також користувачу необхідно вказати довжину приміщення у метрах по осі X, рисунок 5.2.11, обрати камери, які необхідно використовувати для даного приміщення, рисунок 5.2.12, а також обрати мету розстановки камер, рисунок 5.2.13.

Система обробить виділене зображення та виведе його програмне відображення, рисунок 5.2.14. Останній крок – розстановка камер системою, результат роботи програми відображається користувачу у двох форматах:

- Таблиця з описом для обраних типів камер, рисунок 5.2.15;
- Фото приміщення з розставленими камерами, рисунок 5.2.16.

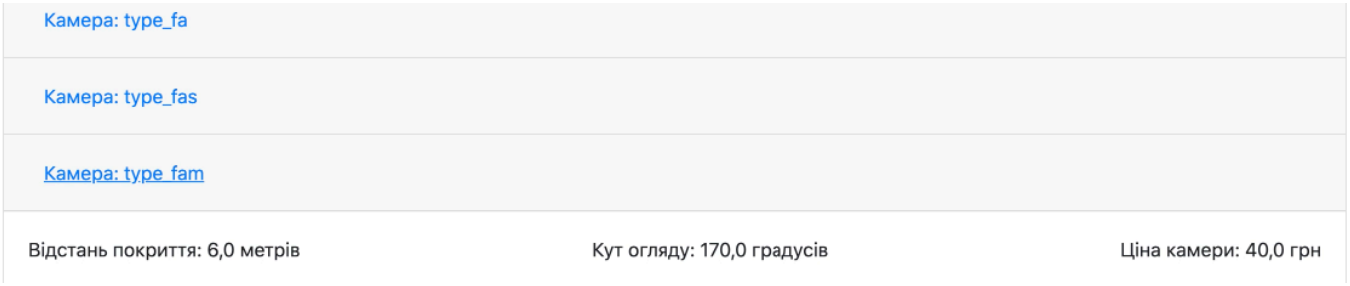

Рисунок 5.2.6 — Інформація про наявні відеокамери

Рисунок 5.2.7 — Додавання камери з користувацькими характеристиками

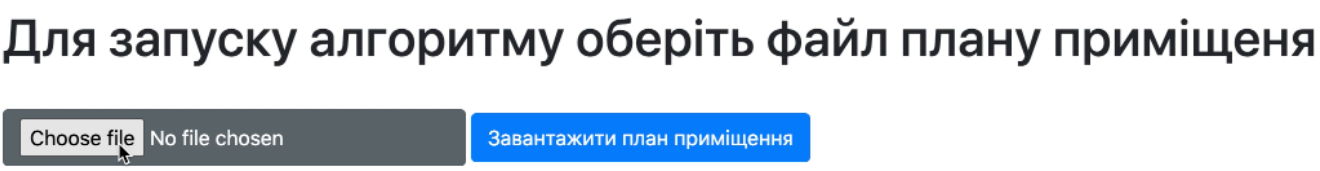

Рисунок 5.2.8 — Додавання плану приміщення в системі

Рисунок 5.2.9 — Вигляд завантаженого плану

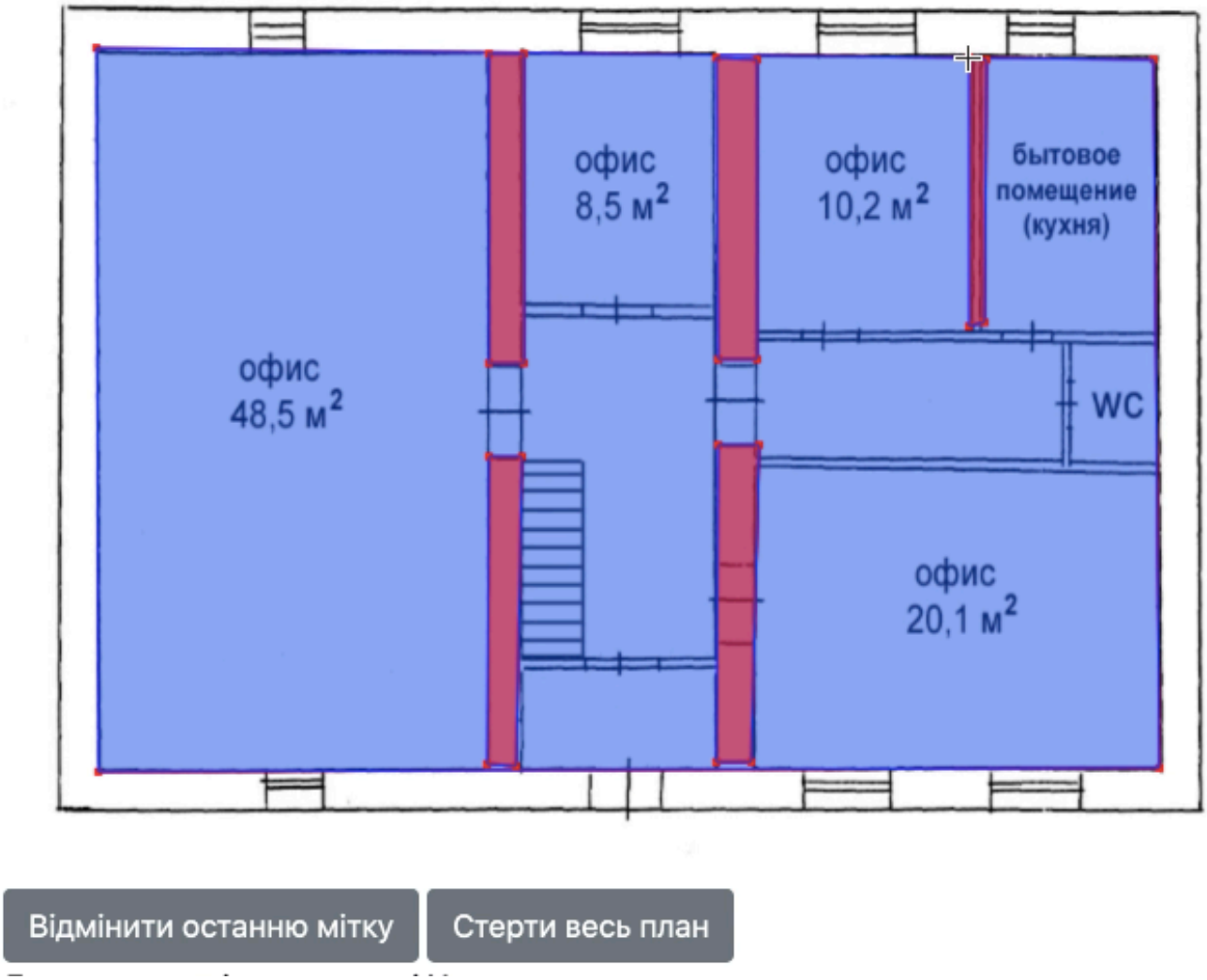

Рисунок 5.2.10 — Виділена область на завантаженому плані

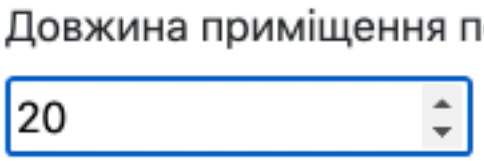

Рисунок 5.2.11 — Форма внесення довжини приміщення у метрах по осі X

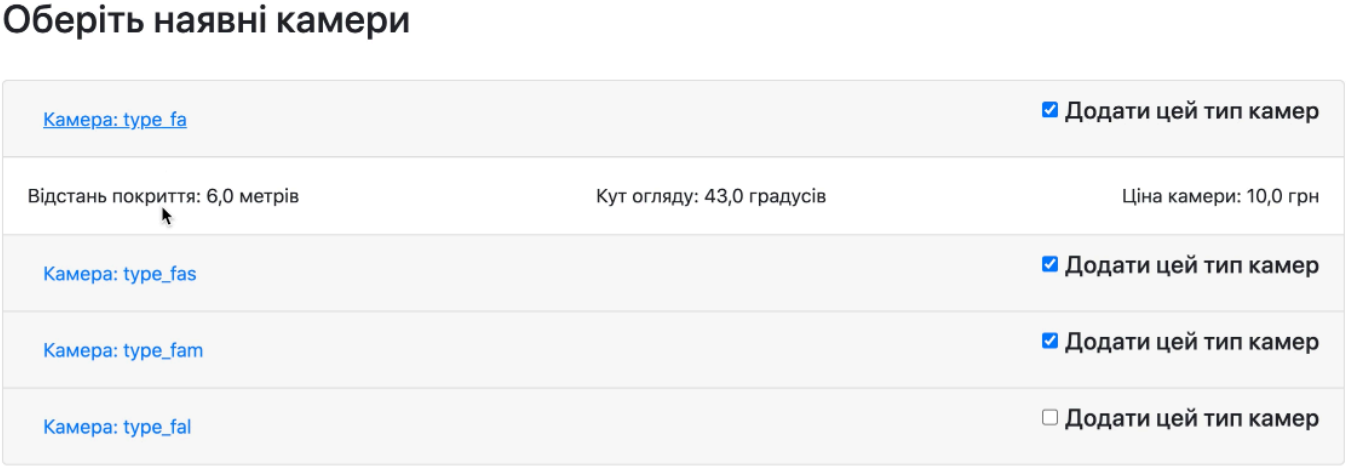

Рисунок 5.2.12 — Вибір камер для даного приміщення

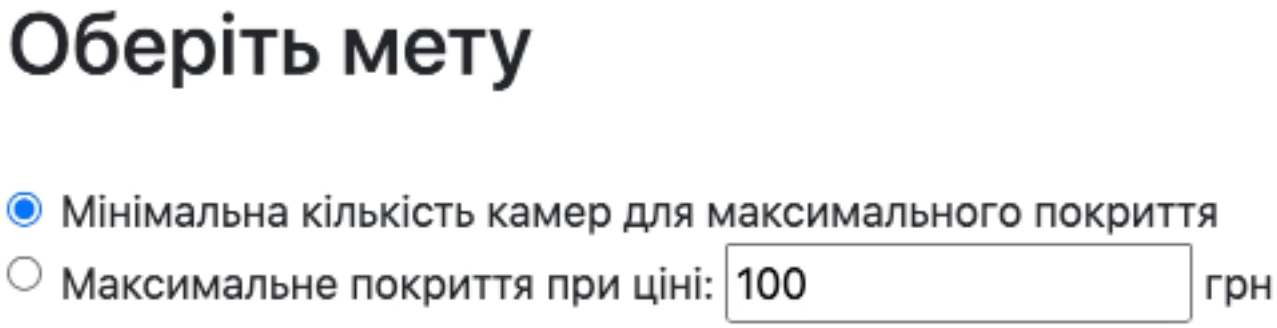

Рисунок 5.2.13 — Вибір мети розстановки камер

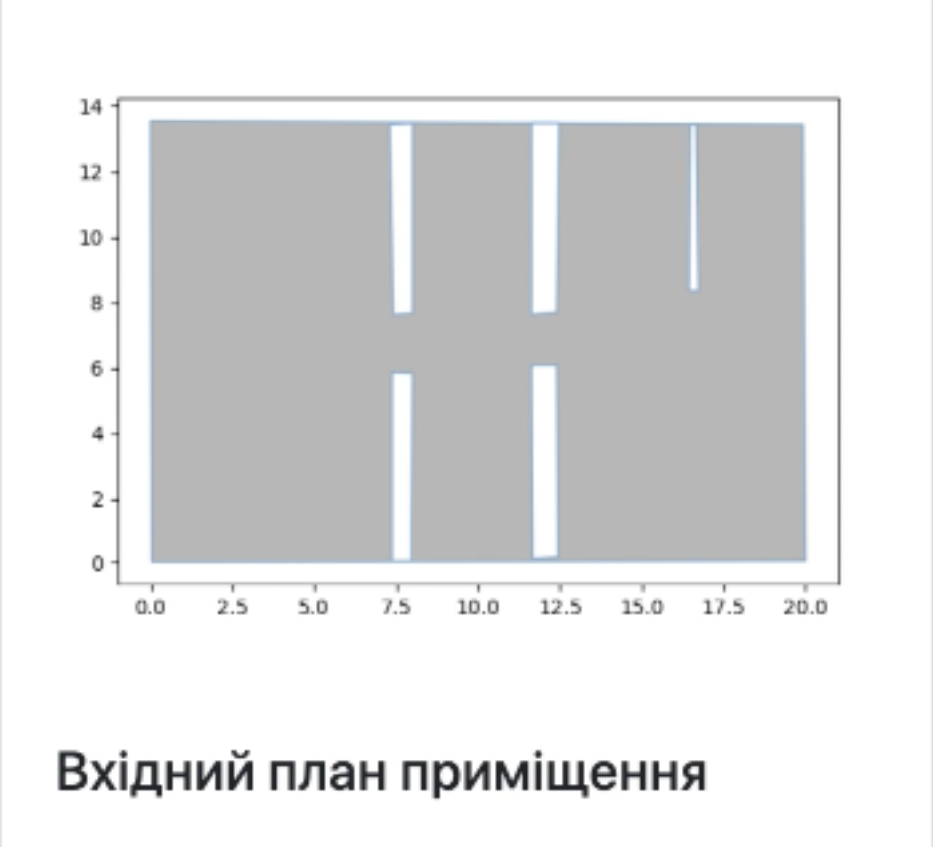

Рисунок 5.2.14 — Програмне відображення вхідних даних

Результати роботи системи автоматичного розміщення обраних камер відеоспостереження у заданому приміщенні

Тип камери	Вартість однієї камери	Кількість камер даного типу	Сума вартості камер даного типу
type_fas	45,0 грн	5	225,0 грн
type_fa	10,0 грн	4	40,0 грн

Рисунок 5.2.15 — Результат роботи програмного алгоритму у вигляді таблиці

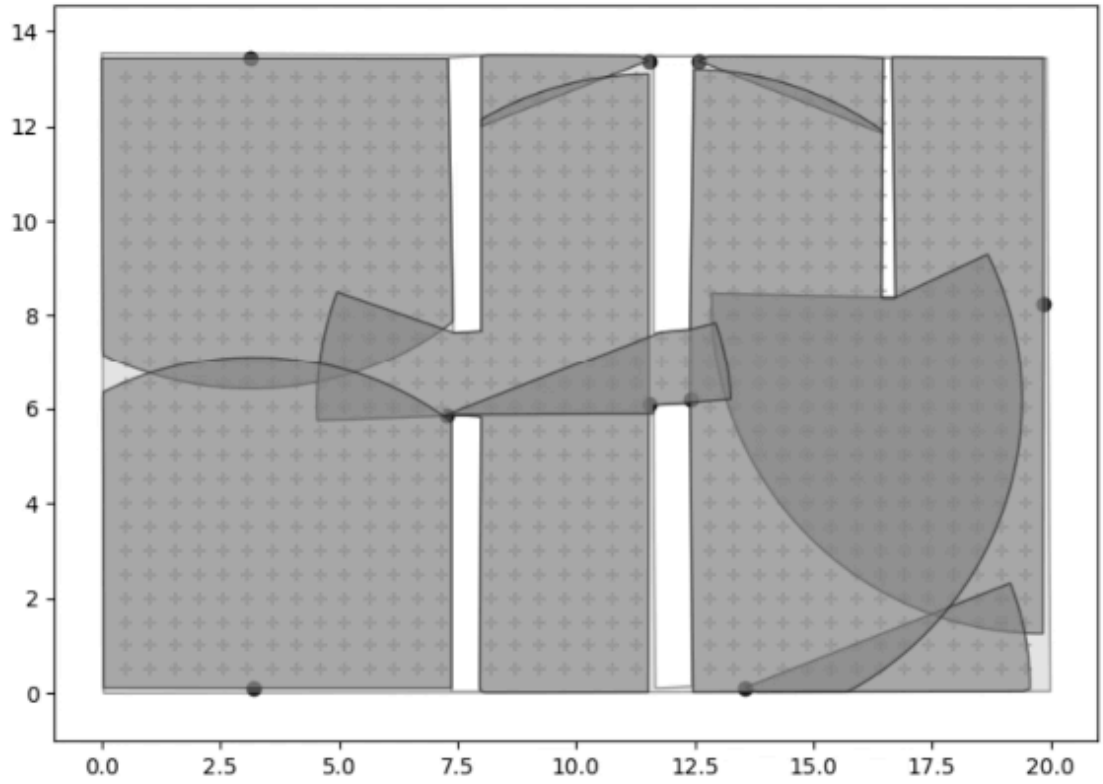

Рисунок 5.2.16 — Кінцевий результат роботи програмного алгоритму

Висновки до розділу

У розділі Робота користувача з системою було розглянуто як функціонує система при діях користувача. Наведено зображення екрану ключових кроків роботи програми. Описані необхідні дії користувача для роботи з системою.

6. ТЕСТУВАННЯ

Забезпечення якості це діяльність по забезпеченню того, щоб розробник програмної системи надавав замовникам найкращий продукт або послугу. Quality Assurance (QA) фокусується на поліпшенні процесів доставки якісних продуктів. Організація повинна забезпечити ефективність і результативність процесів у відповідності зі стандартами якості, встановленими для програмних продуктів.

Для забезпечення стовідсоткової якості продукту необхідно проводити функціональне тестування за участі розробників та мануальне тестування за участі групи потенційних користувачів продукту. На рисунку 6.1 зображено життевий цикл розробки продукту з фазами тестування.

Рисунок 6.1 – Етапи тестування при розробці продукту

Функціональне тестування - це тип тестування програмного забезпечення, при якому система тестується на відповідність функціональним вимогам/специфікаціям.

Функції перевіряються шляхом подачі на них вхідних даних і перевірки вихідних даних. Функціональне тестування гарантує, що вимоги належним чином задоволені додатком. Цей тип тестування пов'язаний не з тим, як відбувається обробка, а з результатами обробки. Він імітує фактичне використання системи, але не робить ніяких припущень про структуру системи.

Під час функціонального тестування використовується метод Box Testing, що включає в себе White Box testing та Black Box testing рисунок 6.2.

White box testing - це тестування внутрішньої структури, дизайну і кодування програмного рішення. У цьому типі тестування код видно тестеру. Основна увага приділяється перевірці потоку вхідних і вихідних даних через додаток, поліпшенню

дизайну та зручності використання, посилення безпеки. White Box testing зазвичай виконується розробниками.

White Box testing включає тестування програмного коду для наступного:

- Внутрішні проблеми в безпеці;
- Зламаний або погано структурованіий код;
- Потік вхідних даних через код;
- Перевірка результату;
- Функціональність умовних циклів;
- Тестування кожного оператора, об'єкта і функції на індивідуальній основі.

Тестування може проводитися на системному, інтеграційному і модульному рівнях розробки програмного забезпечення. Однією з основних цілей тестування whitebox є перевірка робочого процесу для додатка. Він включає в себе тестування ряду зумовлених вхідних даних по відношенню до очікуваних або бажаних вихідних даних, так що коли конкретний ввід не призводить до очікуваного вихідному сигналу, ви зіткнулися з помилкою.

Black Box testing визначається як методика тестування, при якій функціональність тестованої програми тестується без урахування внутрішньої структури коду, деталей реалізації і знання внутрішніх шляхів програмного забезпечення. Цей тип тестування повністю заснований на вимогах і специфікаціях програмного забезпечення. У BlackBox Testing розробник фокусується на входах і виходах програмної системи, не турбуючись про внутрішні процеси цієї програми.

Black-Box може бути будь-якою програмною системою, яку потрібно протестувати. Наприклад, операційна система, така як Windows, веб-сайт, такий як Google, база даних, така як Oracle. У Black Box Testing є можливість протестувати ці додатки, просто зосередившись на входах і виходах, не знаючи їх внутрішньої реалізації коду.

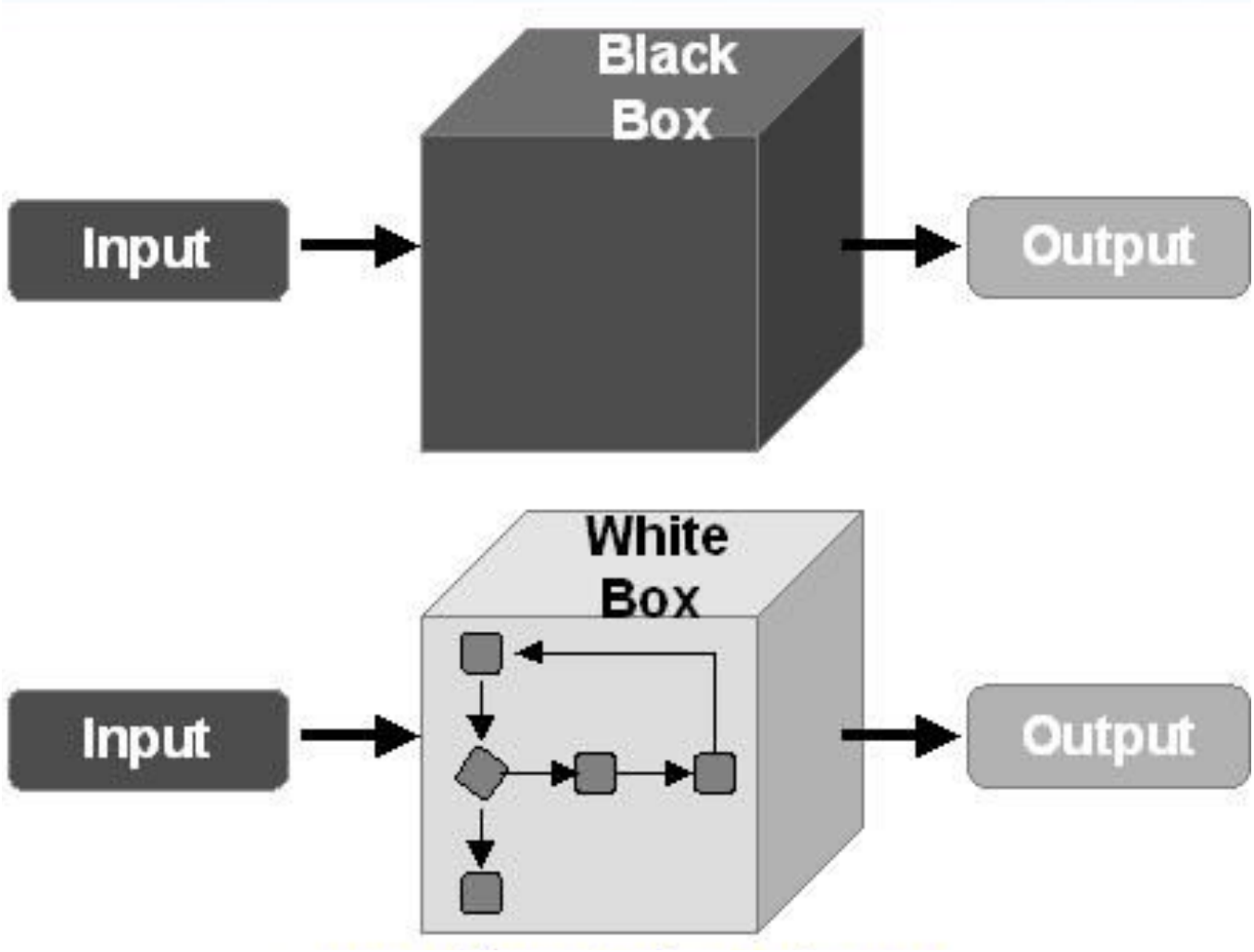

Рисунок 6.2 – Різниця між Black Box Testing та White Box testing

Функціональне тестування зазвичай виконується на рівнях системного тестування та приймального тестування.

Як правило, функціональне тестування включає в себе наступні етапи:

- Визначення функцій, які має виконувати програмне забезпечення;
- Створення вхідних даних на основі характеристик функції;
- Визначення висновків на основі характеристик функції;
- Виконання контрольного тесту;
- Порівняння фактичних та очікуваних результатів.

Функціональне тестування є більш ефективним, коли умови тестування створюються безпосередньо з вимог користувача або бізнесу. Коли умови тестування створюються з системної документації (системні вимоги/проектна документація), дефекти в цій документації не будуть виявлені в ході тестування, і це може стати

причиною гніву кінцевих користувачів, коли вони будуть використовувати програмне забезпечення.

Користувацьке тестування, також відоме як бета-тестування або тестування кінцевим користувачем, визначається як тестування програмного забезпечення користувачем або клієнтом, щоб визначити, чи може воно бути прийнято чи ні. Це остаточне тестування, яке виконує після завершення функціонального тестування. Основна мета цього тестування - перевірка відповідності програмного забезпечення вимогам бізнесу. Ця перевірка виконується кінцевими користувачами, знайомими з вимогами бізнесу.

Оскільки приймальний тест користувача є останнім тестом, який проводиться перед запуском програмного забезпечення, очевидно, що це останній шанс для замовника протестувати програмне забезпечення і виміряти, чи підходить він для цієї мети.

Кількість етапів, що виконуються у виконанні тесту на прийняття користувача, може варіюватися залежно від того, наскільки детально кожна команда хоче визначити кожен етап у процесі. Ці кроки зазвичай включають:

1. Етап планування, де окреслені вимоги бізнесу та стратегія для UAT
2. Ідентифікація та створення тестових сценаріїв. Ці тестові сценарії повинні охоплювати якомога більше функціональних випадків, з якими стикаються кінцеві користувачі.
3. Вибираються групи для тестування. Розробники можуть вирішити, щоб декілька кінцевих користувачів тестували програмне забезпечення або відкривали тест ще багатьом, пропонуючи безкоштовну пробну версію в Інтернеті.
4. Етап тестування та документації, коли кінцеві користувачі починають тестувати програмне забезпечення та реєструються будь-які потенційні помилки та інші проблеми.
5. Фаза виправлення, коли команда розробників вносить корективи в код, вирішує будь-які помилки або вносить запропоновані зміни.
6. Після цього програмне забезпечення має бути готовим до випуску у своє виробниче середовище.

У процесі розробки системи автоматичного розташуванні камер відеоспостереження було проведено тестування.

Деякі випадки тест-кейсів для фукціонального тестування:

- Перевірка коректності завантаження зображення з планом приміщення в систему;
- Коректність функції додавання нової конфігурації камери;
- Перевірка коректності запису даних про нового користувача при реєстрації.

Для проведення користувацького тестування було розроблено тест-кейси та залучено п'ять потенційних користувачів програми для перевірки її якості, та для того, щоб переконатися, що програма задовольняє їх потреби.

Приклади тест-кейсів для користувацького тестування:

- Вибір камери;
- Перевірка роботи функції виділення полігону та перешкод;
- Перевірка функції скидання паролю за допомогою пошти.

Тестування як функціональне так і користувацьке було проведено успішно. Знайдені помилки було налагоджено. Проведено тестування налагоджень, що показало що система працює коректно. Це свідчить про те, що продукт система автоматичного розташування камер відеоспостереження у приміщенні функціонує правильно і готова до використання.

Висновки до розділу

У даному розділі було розглянуто основні підходи до тестування програмного забезпечення. Перевірка якості розробленої системи була проведена по стандартам тестування. Що показало коректність функціонування системи та її готовність до безпосереднього використання користувачами.

ВИСНОВКИ

У ході роботи над дипломним проектом була розроблена система автоматичного розташування камер відеоспостереження. Проаналізовано існуюче програмне забезпечення для вирішення поставлених задач. Аналіз показав, що існуючі системи вирішують проблему дизайну систем не у повному обсязі або подають не коректний результат, який не можна втілити в життя. Деякі з них мають громіздкий інтерфейс та мають високі апаратні вимоги до пристроїв користувачів.

У процесі розробки було реалізовано усі необхідні підсистеми веб-додатку. Обрано оптимальні засоби розробки та підходи до проектування. Система має монолітну клієнт-серверну архітектуру.

Розроблена програмна система дозволяє автоматично проектувати системи відео нагляду у приміщеннях з урахуванням предметів та об’єктів, які можуть перешкоджати видимості камери. Вирішує проблеми користувачів які мають намір розробити дизайн системи спостереження.

Систему було налагоджено, протестовано та випробувано, що показало, коректність роботи програмного забезпечення. Система є ефективною та вирішує всі поставлені задачі по автоматичному розташуванні камер у приміщенні.

Потенційними користувачами системи є особи, які мають бажання захистити свій дім за допомогою камер відео нагляду. І в той же час, прагнуть зробити це найбільш ефективно з найменшими витратами на сторонні додаткові послуги. Ціль потенційних користувачів системи — знайти рішення для оптимального розташування камер і покрити максимальну площу приміщення.

Програмне забезпечення може бути використано на будь-якій операційній системі ПК, має бути встановлений браузер, який підтримує останні веб-стандарти, а також потрібен постійний доступ до інтернету.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Решение практических задач по проектированию видеонаблюдения [Електронний ресурс] — Режим доступу до ресурсу: http://www.jvsg.com/uroki-videnabludeniya/.

2. Jun-Woo Ahn — Two-Phase Algorithm for Optimal Camera Placement [Електронний ресурс] / Jun-Woo Ahn — 2016. — Режим доступу до ресурсу: https://www.hindawi.com/journals/sp/2016/4801784/.

3. Як працюють системи відеоспостереження [Електронний ресурс] — 2018. — Режим доступу до ресурсу: http://www.atlant-holding.com.ua/ua/news/64-yakpracyuyut-sistemi-videosposterezhennya.

4. Системи аналогового відеоспостереження [Електронний ресурс] — 2018. — Режим доступу до ресурсу: http://www.bsi-group.com.ua/ua/systemssecurity/view/Video_analog.

5. Системи відеонагляду. Відмінність цифрової і аналогової системи відеоспостереження [Електронний ресурс] — 2018. — Режим доступу до ресурсу: http://www.ohrana-ua.com/articles/801-sistemi-vdeonaglyadu-vdmnnstcifrovoyi-analogovoyi-sistemi-vdeosposterezhennya.html.

6. Системи безпеки для будинку [Електронний ресурс] — 2018. — Режим доступу до ресурсу: http://kristall-systems.net.ua/ua/resheniya /security_systems_for_home/.

7. Системи IP-відеоспостереження. Огляд [Електронний ресурс] — 2018. — Режим доступу до ресурсу: https://valtek.com.ua/ua/systemintegration/security-control-system/video-surveillance/ip-systems-review.

8. Цифрові системи відеоспостереження [Електронний ресурс] — 2018. — Режим доступу до ресурсу: https://www.vostok.dp.ua/ukr/infa1/sistemy_vidyeonablyudeniya/digital-video/.

9. Лутц Марк — Изучаем Python / Марк Лутц — 2011.

10. Scott W. Ambler — The Object Primer 3rd Ed: Agile Model Driven Development (AMDD) with UML 2 / Scott W. Ambler // Cambridge University Press — 2001. — Режим доступу до ресурсу: http://www.ambysoft.com/books/theObjectPrimer.html

11. Gustavo Olague — Optimal camera placement for accurate reconstruction [Електронний ресурс] / Gustavo Olague. — 2016. — Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/abs/pii/S0031320301000760.

12. Jose-Joel Gonzalez-Barbosa, Teresa Garcia-Ramirez, Joaquin Salas, Juan Bautista Hurtado-Ramos, Jose-de-Jesus Rico-Jimenez Olague — Optimal camera placement for total coverage [Електронний ресурс] / Jose-Joel Gonzalez-Barbosa, Teresa Garcia-Ramirez, Joaquin Salas, Juan Bautista Hurtado-Ramos, Jose-de-Jesus Rico-Jimenez Olague. — 2009. — Режим доступу до ресурсу: https://ieeexplore.ieee.org/document/5152761/authors#authors.

13. Siraj ul Haq — Introduction to Monolithic Architecture and MicroServices Architecture [Електронний ресурс] / Siraj ul Haq — 2018. — Режим доступу до ресурсу: https://medium.com/koderlabs/introduction-to-monolithic-architecture-and-microservices-architecture-b211a5955c63.

14. Shif Ben Avraham — What is REST — A Simple Explanation for Beginners [Електронний ресурс] / Shif Ben Avraham — 2017. — Режим доступу до ресурсу: https://medium.com/extend/what-is-rest-a-simple-explanation-for-beginners-part-1-introduction-b4a072f8740f.

ДОДАТОК А

Система автоматичного розташування камер відеоспостереження у приміщенні

Специфікація

УКР.НТУУ «КПІ». ТІ6295_20Б

Аркушів 2

Київ 2020

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ«КПІ»_ТЕФ_АПЕПС_ТІ6295_20Б	ТІ62_Шевчук_Записка.doc	Текстова частина дипломної роботи
Компоненти		
УКР.НТУУ«КПІ»_ТЕФ_АПЕПС_ТІ6295_20Б	views.py	Логіка роботи веб-застосунку
УКР.НТУУ«КПІ»_ТЕФ_АПЕПС_ТІ6295_20Б	algorithm.py	Алгоритм системи
УКР.НТУУ«КПІ»_ТЕФ_АПЕПС_ТІ6295_20Б	optimization.py	Оптимізація алгоритму
УКР.НТУУ«КПІ»_ТЕФ_АПЕПС_ТІ6295_20Б	plot_results.py	Алгоритм виводу результатів користувачу

ДОДАТОК Б

Система автоматичного розташування камер відеоспостереження у приміщенні

Текст програми

УКР.НТУУ «КПІ». ТІ6295_20Б

Аркушів 18

Київ 2020

views.py

```
from django.shortcuts import render
from django.core.files.storage import FileSystemStorage

from django.http import JsonResponse
from shapely.geometry import Polygon

import matplotlib.pyplot as plt
from descartes import PolygonPatch
from django.contrib.auth.decorators import login_required
import logging
import json

from apps.algorithm_app.algorithm_model import model, clear_tmp_data
from .models import Camera
from .forms import CameraCreationForm


def index_view(request):
    context = {
        "cameras": Camera.objects.all(),
    }
    img_file = request.FILES.get('imgfile')
    if request.method == "POST" and img_file:

        fs = FileSystemStorage()
        filename = fs.save(img_file.name, img_file)

        uploaded_file_url = fs.url(filename)
        context['uploaded_file_url'] = uploaded_file_url

    return render(request, "main_app/index.html", context)


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


def results_view(request):
    with open("media/results.json") as f:
        context = json.load(f)

    cameras_price = dict(context["sensors"])
    cameras_types = [row[0] for row in context["sensors"]]

    from collections import Counter
    counter = Counter(cameras_types)
    cameras_results = [(key, value, counter[key], value * counter[key]) for key, value in
```

```
cameras_price.items()]
    context["cameras_results"] = cameras_results
    context["camera_count"] = len(context["sensors"])

    return render(request, "main_app/results.html", context)


@login_required(login_url='/accounts/login/')
def add_camera_view(request):
    if request.method == "POST":
        form = CameraCreationForm(request.POST)
        if form.is_valid():

            post = form.save(commit=False)
            post.save()
    else:
        form = CameraCreationForm()

    context = {
        "cameras": Camera.objects.all(),
        "form": form,
    }
    return render(request, "main_app/add_camera.html", context)


def save_figures_view(request):
    import time
    time.sleep(3)
    return JsonResponse({})
    main_polygon_perimeter = json.loads(request.GET.get("main_polygon"))

    barricades_perimeters = json.loads(request.GET.get("barricades"))

    enabled_cameras = json.loads(request.GET.get("enabled_cameras"))
    max_size = float(request.GET.get("max_len_x"))
    processing_type = str(request.GET.get("processing_type"))

    max_price = int(request.GET.get("max_price"))
    if not main_polygon_perimeter:
        context = {"err": 'Необхідно виділити полігон'}
        return JsonResponse(context)
    main_polygon = []
    barricades = []
    max_x = 0.0001
    max_y = 0.0001
    # max_size = 10
    for coord in main_polygon_perimeter:
        x = coord["x"]
        y = coord["y"]
        if x > max_x:
            max_x = x
        if y > max_y:
```

```
        max_y = y
if max_x < max_y:
    ratio = max_y / max_x
else:
    ratio = max_x / max_y
# ratio = max_y / max_x
for coord in main_polygon_perimeter:
    x = coord["x"]
    y = coord["y"]
    # x = x * max_size / max_x / ratio
    # y = y * max_size / max_y
    if max_x < max_y:
        x = x * max_size / max_x / ratio
        y = y * max_size / max_y
    else:
        x = x * max_size / max_x
        y = y * max_size / max_y / ratio
    y = max_size - y
    main_polygon.append([x, y])
min_x_t = 100000000

min_y_t = 100000000

max_x_t = 0.0001
max_y_t = 0.0001
for point in main_polygon:

    x = point[0]
    y = point[1]

    if x > max_x_t:
        max_x_t = x
    if y > max_y_t:
        max_y_t = y

    if x < min_x_t:
        min_x_t = x
    if y < min_y_t:
        min_y_t = y
x_ratio = max_size / (max_x_t - min_x_t)
for point in main_polygon:
    x = point[0]
    y = point[1]
    point[0] = (x - min_x_t) * x_ratio
    point[1] = (y - min_y_t) * x_ratio
for barricade_perimeter in barricades_perimeters:
    barricade = []
    for coord in barricade_perimeter:
        x = coord["x"]
        y = coord["y"]
        if max_x < max_y:
            x = x * max_size / max_x / ratio
```

```
            y = y * max_size / max_y
        else:
            x = x * max_size / max_x
            y = y * max_size / max_y / ratio
        y = max_size - y
        x = (x - min_x_t) * x_ratio
        y = (y - min_y_t) * x_ratio
        barricade.append((x, y))
    barricades.append(barricade[::-1])
OBJ_POLYGON = Polygon(main_polygon, barricades)
BLUE = '#6699cc'
GRAY = '#999999'
fig = plt.figure()
ax = fig.gca()
ax.add_patch(PolygonPatch(OBJ_POLYGON, fc=GRAY, ec=BLUE, alpha=0.7, zorder=2))
ax.axis('scaled')
plt.savefig('media/foo.png')

clear_tmp_data()

SENSOR_TYPES = []
for camera in Camera.objects.filter(type__in=enabled_cameras):
    camera_settings = dict(camera.__dict__)
    camera_settings["theta"] = camera.theta
    SENSOR_TYPES.append(camera_settings)

setter = model(
    obj_polygon=OBJ_POLYGON,
    sensor_types=SENSOR_TYPES,
    grid_size=0.5,
    # grid_size=1,
    sensor_buffer=-0.1,
    max_dist=1,
    min_dist=0.5,
    alpha_num=4,
    # alpha_num=8,
    cover_times=1
)
if processing_type == "option-min":
    try:
        num, costs = setter.compute(mode='min', method='ilp')
    except Exception:
        context = {"err": 'Неможливе повне покриття при таких камерах'}
        return JsonResponse(context)
else:
    try:
        num, costs = setter.compute(mode='max', method='dp', value=max_price)
    except Exception:
        context = {"err": 'Помилка, спробуйте ще раз з іншими налаштуваннями'}
        return JsonResponse(context)
# setter.compute(mode='max', method='dp', value=200)
# setter.compute(mode='min', method='ilp')
```

```
    logging.warning(setter.get_coverage)
    logging.warning(setter.sensors.get_sensors)
    fig = plt.figure(figsize=(9, 6))
    ax = fig.add_subplot(111)

    setter.plot(ax)

    ax.set_xlim(OBJ_POLYGON.bounds[0] - 1, OBJ_POLYGON.bounds[2] + 1)
    ax.set_ylim(OBJ_POLYGON.bounds[1] - 1, OBJ_POLYGON.bounds[3] + 1)
    ax.set_aspect('equal')
    plt.savefig('media/foo1.png')

    context = {
        "img_path": 'media/foo.png',
        "processed_img_path": 'media/foo1.png',
        "cost": sum(setter.sensors.get_costs),
        "num": setter.sensors.get_num,
        "sensors": [(sensor.type, sensor.cost) for sensor in setter.sensors.get_sensors],
    }
    with open('media/results.json', 'w') as fp:
        json.dump(context, fp)
    return JsonResponse(context)
```

algorithm.py

```
from shapely.geometry import Point
import numpy as np
import os
from cvxopt import matrix
from cvxopt.glpk import ilp
import logging

from .sensors import Sensor, Sensors
from .visible import visible, clip, discrete
from .plot_results import plot_visible, plot_background, plot_sensors, plot_points

def clear_tmp_data():
    if os.path.isfile('data/A.npy'):
        os.remove('data/A.npy')
    if os.path.isfile('data/b.npy'):
        os.remove('data/b.npy')
    if os.path.isfile('data/c.npy'):
        os.remove('data/c.npy')
    if os.path.isfile('data/configs.csv'):
        os.remove('data/configs.csv')

class model(object):
    def __init__(self, obj_polygon, sensor_types, grid_size=0.5,
                 sensor_buffer=-0.1, max_dist=1.0, min_dist=0.2, alpha_num=8,
                 cover_times=1):
        self.layout = obj_polygon
```

```
        self.grid_size = grid_size
        self.cover_times = cover_times

        sensors = Sensors()
        # if configs exist, load directly.
        if os.path.exists('data/configs.csv'):
            sensors.load_configs()
        # or generate all candidate sensors
        else:
            layout_buffer = obj_polygon.buffer(sensor_buffer, 0)
            candidate_locations = clip(layout_buffer, max_dist, min_dist)
            # step = 0.1
            # while len(candidate_locations) > 100:
            #     max_dist += step
            #     candidate_locations = clip(layout_buffer, max_dist, min_dist)
            # print(len(candidate_locations), max_dist)

            sensors.generate_configs(sensor_types, candidate_locations, alpha_num)
            sensors.save_configs()

        self.sensors = sensors

        self.initialize()

    def initialize(self):
        # discrete layout
        # self.grid_size = 5
        layout_points = discrete(self.layout, self.grid_size)
        # while len(layout_points) > 100:
        #     self.grid_size += 0.5
        #     layout_points = discrete(self.layout, self.grid_size)
        self.layout_points = layout_points
        self.layout_points_mask = np.zeros(len(self.layout_points), np.bool)
        print("==========>>> Discrete layout into {} points ...
<<<==========".format(len(self.layout_points)))
        print("==========>>> Grid size {} ... <<<==========".format(self.grid_size))

        if os.path.exists('data/A.npy') and os.path.exists('data/b.npy') and os.path.exists('data/c.npy'):
            print("==========>>> Load cover A, b, c directly ... <<<==========")
            self.A = np.load('data/A.npy')
            self.b = np.load('data/b.npy')
            self.c = np.load('data/c.npy')
        else:
            print("==========>>> Compute and save cover A, b, c ... <<<==========")
            A = self._cover_A()
            b = self._cover_b(cover_times=self.cover_times)
            c = self._cover_c()
            print("==========>>> Sensor candidates: %d <<<==========" % self.sensors.get_num)
            mask = A.sum(axis=1) > 0
            self.A = A[mask].astype(np.int8)
            self.b = b
            self.c = c[mask]
```

```
            np.save('data/A.npy', self.A)
            np.save('data/b.npy', self.b)
            np.save('data/c.npy', self.c)
            self.sensors.update(mask)
            self.sensors.save_configs()
            print("==========>>> Sensor candidates has been decreased to: %d <<<==========" %
self.sensors.get_num)

    def _cover_A(self):
        cover_A = []
        for sensor in self.sensors.get_sensors:
            visibile_region = visible(sensor, self.layout)
            if visibile_region == None:
                cover_a = np.zeros(len(self.layout_points))
            else:
                cover_a = [visibile_region.contains(Point(point)) for point in self.layout_points]

            cover_A.append(cover_a)
        return np.array(cover_A)

    def _cover_b(self, cover_times):
        num = len(self.layout_points)
        return np.ones(num) * cover_times

    def _cover_c(self):
        return self.sensors.get_costs

    def compute(self, mode='min', method='ilp', value=0):
        if mode == 'min':
            if method == 'ilp':
                x = min_solver_ilp(self.A, self.b, self.c)

        elif mode == 'max':
            if method == 'dp':
                x = max_solver_dp(self.A, self.b, self.c, value)

        mask = x
        self.sensors.update(mask)

        self.layout_points_mask = np.dot(self.A.T, x) >= 1
        return self.sensors.get_num, self.sensors.get_costs

    def plot(self, ax, sensor_list=None):

        plot_background(ax, self.layout)

        xs, ys = self.sensors.get_locations

        if sensor_list == None:
            visible_regions = [visible(sensor, self.layout) for sensor in self.sensors.get_sensors]
        else:
            visible_regions = [visible(self.sensors.get_sensors[i], self.layout) for i in sensor_list]
```

```
            xs = [xs[i] for i in sensor_list]
            ys = [ys[i] for i in sensor_list]

        plot_visible(ax, visible_regions)
        plot_sensors(ax, xs, ys)

        plot_points(ax, self.layout_points, self.layout_points_mask)

    @property
    def get_coverage(self):
        total_n = len(self.layout_points)
        cover_n = np.sum(self.layout_points_mask)
        return cover_n / total_n

    @property
    def get_cost(self):
        return self.sensors.get_costs

# max problem:
================================================================================
================================
# Maximize coverage
# obj. xA - b >= 0
# s.t. xc <= C
def max_solver_dp(A, b, c, C):
    m = c.shape[0]          # x dimension
    n = b.shape[0]          # s.t. dimension
    mat_obj = np.zeros((m, C, n), dtype=np.int)
    mat_x = np.zeros((m, C, m), dtype=np.bool)
    for i in range(m):

        for j in range(C):

            # when the cost of ith is less than total cost
            if c[i] <= j:

                forward_obj = mat_obj[i - 1, int(j - c[i])] + A[i]

                backward_obj = mat_obj[i - 1, j]
                forward_sum = (forward_obj - b >= 0).sum()
                backward_sum = (backward_obj - b >= 0).sum()
                if forward_sum >= backward_sum:
                    mat_obj[i, j] = forward_obj

                    mat_x[i, j] = mat_x[i - 1, int(j - c[i])]
                    mat_x[i, j, i] = True
                else:
                    mat_obj[i, j] = backward_obj

                    mat_x[i, j] = mat_x[i - 1, j]
    return mat_x[-1, -1]
```

```
# min problem:
================================================================================
================================
# Minimize cost
# obj. xc
# s.t. Ax >= b
def min_solver_ilp(A, b, c):
    # ilp form:
    # obj. min c'x
    # s.t. Gx <= h
    G = matrix((-1) * A.T.astype(np.float))
    h = matrix((-1) * b.astype(np.float))
    c = matrix(c.astype(np.float))
    x_num = len(c)
    (status, x) = ilp(c, G, h, B=set(range(x_num)))
    return np.array(x)

# Multi-object problem:
================================================================================
# obj. max xA - b >= 0
```

optimization.py

```
import numpy as np
import matplotlib.pyplot as plt
import time

COLORS = ['#1a1a1a', '#404040', '#808080', '#bfbfbf']


class Individual(object):
    # individual unit
    def __init__(self):
        self.genotype = None
        self.phenotype = None

        # non-dominated sorting
        self.rank = None

        self.domination_count = None
        self.dominated_solutions = set()

        # crowding distance sorting
        self.crowding_distance = None

    def set_genotype(self, new_genotype):
        self.genotype = new_genotype

    def set_phenotype(self, new_phenotype):
        self.phenotype = new_phenotype
```

```
class Population(object):
    # population unit
    def __init__(self):
        # a list to store individuals
        self.population = []
        # a multilevel list to store individuals
        self.fronts = []

    def __len__(self):
        return len(self.population)

    def __iter__(self):
        return self.population.__iter__()

    def append(self, new_individual):
        self.population.append(new_individual)

    def extend(self, new_individuals):
        self.population.extend(new_individuals)

    def contains(self, new_individual):

        if_contain = False
        for individual in self.population:
            if np.array_equal(new_individual.genotype, individual.genotype):

                if_contain = True
                break
        return if_contain

    def get_phenotypes(self):
        phenotypes = []
        for individual in self.population:

            phenotypes.append(individual.phenotype)
        return np.array(phenotypes)

    @property
    def max_phenotype(self):
        maximum = np.max(self.get_phenotypes(), axis=0)
        return maximum

    @property
    def min_phenotype(self):
        minimum = np.min(self.get_phenotypes(), axis=0)
        return minimum

class Problem(object):
    def __init__(self, W, b, c):
        """
        obj_1:          min sum(Wx - b) < 0
        obj_2:          min cx
```

```
    :param W:
    :param b:
    :param c:

    """
    self.W = W

    self.b = b
    self.c = c
    self.genotype_num = len(c)
    self.phenotype_num = 2
    assert W.shape == (b.shape[0], c.shape[0]), "Please input compatible W, b and c"

def create_individual(self):
    individual = Individual()

    # generate a binary genotype

    genotype = np.random.randint(100, size=self.genotype_num) // 99

    individual.set_genotype(new_genotype=genotype)
    phenotype = self.function(individual)
    individual.set_phenotype(new_phenotype=phenotype)
    return individual

def function(self, inidividual):
    x = inidividual.genotype
    non_cover = np.mean(np.dot(self.W, x) - self.b < 0)
    cost = np.dot(self.c, x)
    return np.array([non_cover, cost])

def create_population(self, population_size):
    population = Population()
    while len(population) < population_size:

        individual = self.create_individual()
        # if new individual is different from individuals in population, then add

        if not population.contains(new_individual=individual):

            population.append(individual)
    return population

def epsilon_dominates(self, individual_1, individual_2, epsilon=(0, 0)):
    phenotype_1 = individual_1.phenotype
    phenotype_2 = individual_2.phenotype

    epsilons = np.array(epsilon)

    # whether individual 1 dominates individual 2
    dominate_num = np.sum(phenotype_1 - (phenotype_2 + epsilons) < 0)
    if dominate_num > 0:
```

```
            return True
        else:
            return False

class NSGA_ii(object):
    # main algorithm
    def __init__(self, problem, population_size, select_size, mutate_size):
        self.problem = problem
        self.population_size = population_size
        self.select_size = select_size
        self.mutate_size = mutate_size

        print("==========>>> Initialize algorithm ... <<<==========")

        # initialize the first parent population and offspring population - 2n

        self.population = self.problem.create_population(population_size=self.population_size)
        # epsilon non-dominated_sorting
        self.epsilon_non_dominated_sorting(self.population)
        # corwding distance sorting
        for front in self.population.fronts:
            self.crowding_distance_sorting(front)
        # generate offspring
        self.offspring = self.generate_offspring(self.population)

    def evolve(self, num_of_generations, ax):
        durations = []
        for i in range(num_of_generations):

            start = time.time()
            # main loop

            # combine

            self.population.extend(self.offspring)
            # epsilon non-dominated_sorting
            self.epsilon_non_dominated_sorting(self.population)

            new_population = Population()
            front_num = 0

            # corwding distance sorting
            while len(new_population) + len(self.population.fronts[front_num]) < self.population_size:
                self.crowding_distance_sorting(self.population.fronts[front_num])

                new_population.extend(self.population.fronts[front_num])
                front_num += 1

            # fill up
            self.crowding_distance_sorting(self.population.fronts[front_num])
            new_population.extend(self.population.fronts[front_num][:self.population_size -
len(new_population)])
```

```
            self.population = new_population

            # generate new offspring
            self.offspring = self.generate_offspring(self.population)

            duration = time.time() - start
            durations.append(duration)
            print("==========>>> Evolution: [%d / %d] Time: %.3f sec <<<==========" % (i,
num_of_generations, duration))

            if ax:
                if i % 10 == 0:
                    self.plot(ax=ax, color=str(1 - i/num_of_generations))

        durations = np.array(durations)

        return self.population, durations

    def epsilon_non_dominated_sorting(self, population):
        population.fronts = []
        population.fronts.append([])

        for individual in population:
            individual.domination_count = 0
            individual.dominated_solutions = set()

            for other_individual in population:

                if self.problem.epsilon_dominates(individual_1=individual, individual_2=other_individual):
                    individual.dominated_solutions.add(other_individual)

                elif self.problem.epsilon_dominates(individual_1=other_individual, individual_2=individual):
                    individual.domination_count += 1

            if individual.domination_count == 0:
                population.fronts[0].append(individual)
                individual.rank = 0

        i = 0
        while len(population.fronts[i]) > 0:
            temp = []
            for individual in population.fronts[i]:

                for other_individual in individual.dominated_solutions:

                    other_individual.domination_count -= 1
                    if other_individual.domination_count == 0:

                        other_individual.rank = i + 1
                        temp.append(other_individual)
```

```
            i = i + 1
            population.fronts.append(temp)

    def crowding_distance_sorting(self, front):
        if len(front) == 1:
            front[0].crowding_distance = self.problem.phenotype_num
        elif len(front) >= 2:
            for individual in front:
                individual.crowding_distance = 0
            for i in range(self.problem.phenotype_num):
                if i == 0:
                    # ascending sorted from small to big

                    front = sorted(front, key=lambda p: p.phenotype[i])

                    front[0].crowding_distance += self.problem.phenotype_num

                    front[-1].crowding_distance += self.problem.phenotype_num

                    for j in range(1, len(front) - 1):
                        front[i].crowding_distance += (front[j - 1].crowding_distance - front[j +
1].crowding_distance) / \
                                                      (self.population.max_phenotype[i] - self.population.min_phenotype[i])
        # sort front according to crowding distance

        front = sorted(front, key=lambda p: - p.crowding_distance)

    def generate_offspring(self, population):

        offspring = []
        while len(offspring) < self.population_size:

            # select
            parent_1 = self._select(population)
            parent_2 = self._select(population)

            while np.array_equal(parent_1.genotype, parent_2.genotype):
                parent_2 = self._select(population)

            # crossover
            child_1, child_2 = self._crossover(parent_1, parent_2)

            # mutate
            self._mutate(child_1)
            self._mutate(child_2)

            if not population.contains(child_1):
                phenotype_1 = self.problem.function(child_1)
                child_1.set_phenotype(phenotype_1)

                offspring.append(child_1)
```

```
        if len(offspring) == self.population_size:
            break

        if not population.contains(child_2):
            phenotype_2 = self.problem.function(child_2)
            child_2.set_phenotype(phenotype_2)
            offspring.append(child_2)
    return offspring

def _crowding_operator(self, individual_1, individual_2):

    if individual_1.rank < individual_2.rank or (individual_1.rank == individual_2.rank
                                    and individual_1.crowding_distance > individual_2.crowding_distance):
        return True
    else:

        return False

def _select(self, population):
    participant_indices = np.random.choice(np.arange(self.population_size), size=self.select_size)

    participants = [population.population[i] for i in participant_indices]

    best_one = None
    for participant in participants:

        if best_one == None or self._crowding_operator(participant, best_one):
            best_one = participant
    return best_one

def _crossover(self, individual_1, individual_2):
    child_1 = self.problem.create_individual()
    child_2 = self.problem.create_individual()
    num = self.problem.genotype_num

    crossover_indices = np.random.choice(np.arange(num), size=num // 2)

    for i in range(num):
        if i in crossover_indices:

            child_1.genotype[i] = individual_2.genotype[i]
            child_2.genotype[i] = individual_1.genotype[i]

        else:
            child_1.genotype[i] = individual_1.genotype[i]
            child_2.genotype[i] = individual_2.genotype[i]
    return child_1, child_2

def _mutate(self, individual):
    num = self.problem.genotype_num
    mutate_indices = np.random.choice(np.arange(num), size=self.mutate_size)
```

```
        for i in mutate_indices:
            individual.genotype[i] = 1 - individual.genotype[i]

    def plot(self, ax, color=None):
        self.epsilon_non_dominated_sorting(self.population)
        for i, front in enumerate(self.population.fronts):
            for individual in front:
                if color == None:
                    ax.scatter(1 - individual.phenotype[0], individual.phenotype[1], s=5, color=COLORS[i // 4])
                else:
                    ax.scatter(1 - individual.phenotype[0], individual.phenotype[1], s=5, color=color)

if __name__ == '__main__':
    W = np.load('data/A.npy').T
    b = np.load('data/b.npy')
    c = np.load('data/c.npy')

    fig = plt.figure(figsize=(8, 6))
    ax = fig.add_subplot(111)

    problem = Problem(W=W, b=b, c=c)
    nsga = NSGA_ii(problem=problem, population_size=200, select_size=20, mutate_size=20)
    population, times = nsga.evolve(num_of_generations=1000, ax=ax)

    plt.show()
```

plot_results.py

```
from descartes.patch import PolygonPatch
import numpy as np

# color identification
BLUE = '#58D2E8'
GREEN = '#61FF69'
RED = '#F2B6B6'
YELLOW = '#E8ED51'
BLACK = '#1A1A1A'

DARK_90 = '#1a1a1a'

DARK_75 = '#404040'

DARK_50 = '#808080'

DARK_25 = '#bfbfbf'

def plot_visible(ax, visible_regions):
    for visibile_region in visible_regions:
        visible_patch =  PolygonPatch(visibile_region,
                                      facecolor=DARK_50, edgecolor=DARK_90, alpha=0.6, zorder=2)
```

```
        ax.add_patch(visible_patch)

def plot_background(ax, obj_polygon):
    background_patch = PolygonPatch(obj_polygon,
                                    facecolor=DARK_25, edgecolor=DARK_90, alpha=0.4, zorder=2)
    ax.add_patch(background_patch)

def plot_sensors(ax, x, y):
    ax.scatter(x, y, color=DARK_90, marker='o')

def plot_points(ax, points, cover_mask):
    xs, ys = points.T
    ax.scatter(np.ma.masked_array(xs, cover_mask),
               np.ma.masked_array(ys, cover_mask), color=DARK_75, marker='+')

    ax.scatter(np.ma.masked_array(xs, ~cover_mask),
               np.ma.masked_array(ys, ~cover_mask), color=DARK_25, marker='+')
```

ДОДАТОК В

Система автоматичного розташування камер відеоспостереження у приміщенні

Опис програми

УКР.НТУУ «КПІ». ТІ6295_20Б

Аркушів 10

Київ 2020

АНОТАЦІЯ

Додаток містить опис основних компонентів системи автоматичного розташування камер відеоспостереження у приміщенні:

- Графічний інтерфейс користувача для завантаження та взаємодії з планом приміщення;
- Створення полігонів із виділеної області на плані приміщення;
- Алгоритм розстановки відеокамер на заданому плані приміщення;
- Логіка виведення результатів роботи алгоритму в графічний інтерфейс користувача.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ

У цьому додатку міститься опис основних компонентів системи автоматичного розташування камер відеоспостереження у приміщенні. У додатку Б міститься програмний код відповідних компонентів.

Веб-систему розроблено використовуючи мову програмування Python та фреймворк Django. Графічний інтерфейс користувача створено за допомогою мови програмування JavaScript.

Для створення алгоритму розстановки відеокамер використано бібліотеки NumPy та Shapely.

Для зберігання даних використовується система управління базами даних MySQL.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Система автоматичного розташування камер відеоспостереження у приміщенні надає наступний функціонал:

- Завантаження плану приміщення;
- Графічний інтерфейс вибору полігону та перешкод на завантаженому плані, а також конфігурації приміщення;
- Додавання власних конфігурацій камер;
- Перегляд результатів розташування камер відеоспостереження на обраному плані приміщення;
- Перегляд інформації про систему.

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Програмний комплекс складається з двох модулів згідно з наведеною архітектурою:

- Алгоритм автоматичного розташування камер відеоспостереження у приміщенні;
- Веб-додаток для використання алгоритму.

Веб-система складається з користувацького інтерфейсу та логіки взаємодії користувацького інтерфейсу з алгоритмом автоматичного розташування камер відеоспостереження у приміщенні.

ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Для використання системи користувач повинен мати підключення до мережі інтернет та персональний комп’ютер зі встановленим браузером.

Для запуску власної копії серверу веб-систему повинен мати персональний комп’ютер зі встановленим ПЗ Docker.

ВИКЛИК І ЗАВАНТАЖЕННЯ

Система не потребує інсталяції завдяки тому, що вона реалізована у вигляді веб-додатку, доступ до якого можна отримати за наступним посиланням http://127.0.0.1:8001.

ВХІДНІ ДАНІ

Вхідна інформація системи автоматичного розташування камер відеоспостереження у приміщенні:

- Фото файл плану приміщення певного типу, що буде завантажено в систему;
- Виділена область плану приміщення на завантаженому плані;
- Ширина приміщення по осі X;
- Виділені області перешкод на завантаженому плані;
- Набір конфігурацій камер для розташування на виділеному плані приміщення;
- Мета роботи системи.

ВИХІДНІ ДАНІ

Вихідна інформація системи автоматичного розташування камер відеоспостереження у приміщенні:

- Інформація про розроблену систему;
- Кабінет користувача;
- Інформація про виділений користувачем план приміщення;
- Інформація про обрані користувачем конфігурації камер;
- Результати розташування камер відеоспостереження на виділеному плані приміщення;
- Збереження розташування камер в особистому кабінеті користувача.