Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Розробка програмного продукту на основі хмарної архітектури для машинного аналізу зображень (комплексна тема)

Виконали: студенти VI курсу, групи СНнм-61
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

_____ Зелінський А. О.
(підпис) (прізвище та ініціали)

_____ Лісовський В. В.
(підпис) (прізвище та ініціали)

Керівник _____ Никитюк В. В.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Мацюк О.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Осухівська Г.М.
(підпис) (прізвище та ініціали)

Тернопіль
2023

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«____» _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Зелінському Андрію Олеговичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка програмного продукту на основі хмарної архітектури для машинного аналізу зображень (комплексна тема)

Керівник роботи Никитюк В'ячеслав В'ячеславович, к. т. н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 22 » листопада 2022 року № 4/7-950

2. Термін подання студентом завершеної роботи

3. Вихідні дані до роботи Наукові публікації про проектування хмарних рішень та про задачі машинного аналізу зображень і методи їх вирішень

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ.1 Аналіз предметної області з хмарної архітектури та комп'ютерного бачення. 1.1 Аналіз потокової передачі зображень. 1.2 Джерела безперервного потоку зображень. 1.3 Інженерія даних. . 1.4 Машинний аналіз. 1.5 Висновок. 2 Аналіз систем обробки потоку даних та моделей машинного аналізу зображень. 2.1 Опис продукту та вимог. 2.2 Аналіз хмарних платформ. 2.3 Системи передачі та опрацювання. потокових даних. 2.4 Методи та моделі машинного аналізу зображень. 2.5 Висновок. 3 Проектування та реалізація продукту. 3.1 Пошук актантів та варіантів. 3.2 Проектування архітектури продукту. 3.3 Опис програмних рішень. 3.4 Опис інтерфейсу веб-клієнта. 3.5 Розгортання системи та допоміжних сервісів на хмарній платформі. 3.6 Висновок. 4 Охорона праці та безпека в надзвичайних ситуація. 4.1 Характеристика НС воєнного характеру. 4.2 Міжнародні норми соціальної відповідальності. Стандарт SA 8000 «Соціальна відповідальність». 4.3 Підвищення стійкості роботи підприємств приладобудівної галузі в воєнний час. 4.4 Оцінка стійкості роботи економіки до впливу поражаючих факторів ядерної зброї. 4.5 Висновок. Висновки. Перелік джерел. Додатки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Мацюк О.В., доцент		
Безпека в надзвичайних ситуаціях	Клепчик В.М., ст. викладач		

7. Дата видачі завдання 14 листопада 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	14.11.2022-15.11.2022	*Виконано*
2.	Підбір наукових джерел про хмарні рішення,	16.11.2022-27.11.2022	*Виконано*
	архітектуру та методи машинного аналізу зображень		
3.	Опрацювання наукових публікацій та збір даних по	28.11.2022-25.12.2022	*Виконано*
	темі роботи		
4.	Виконання дослідження згідно мети кваліфікаційної	09.01.2023-12.03.2023	*Виконано*
	роботи		
5.	Оформлення розділу «Аналіз предметної області з	13.03.2023-19.03.2023	*Виконано*
	хмарної архітектури та комп’ютерного бачення»		
6.	Оформлення розділу «Аналіз систем обробки потоку	20.03.2023-26.03.2023	*Виконано*
	даних та моделей машинного аналізу зображень»		
7.	Оформлення розділу «Проектування та реалізація	27.03.2023-02.04.2023	*Виконано*
	продукту»		
8.	Виконання завдання до підрозділу «Охорона праці»	10.04.2023-16.04.2023	*Виконано*
9.	Виконання завдання до підрозділу «Безпека в	17.04.2023-23.04.2023	*Виконано*
	надзвичайних ситуаціях»		
10.	Оформлення кваліфікаційної роботи	24.04.2023-30.04.2023	*Виконано*
11.	Нормоконтроль	01.05.2023-07.05.2023	*Виконано*
12.	Перевірка на плагіат	08.05.2023	*Виконано*
13.	Попередній захист кваліфікаційної роботи	15.05.2023	*Виконано*
14.	Захист кваліфікаційної роботи	23.05.2023	

Студент _____ Зелінський А. О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Никитюк В. В.
(підпис) (прізвище та ініціали)

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«____» _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Лісовському Владиславу Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка програмного продукту на основі хмарної архітектури для машинного аналізу зображень (комплексна тема)

Керівник роботи Никитюк В'ячеслав В'ячеславович, к. т. н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 22 » листопада 2022 року № 4/7-950

2. Термін подання студентом завершеної роботи

3. Вихідні дані до роботи Наукові публікації про проектування хмарних рішень та про задачі машинного аналізу зображень і методи їх вирішень

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ.1 Аналіз предметної області з хмарної архітектури та комп'ютерного бачення. 1.1 Аналіз потокової передачі зображень. 1.2 Джерела безперервного потоку зображень. 1.3 Інженерія даних. . 1.4 Машинний аналіз. 1.5 Висновок. 2 Аналіз систем обробки потоку даних та моделей машинного аналізу зображень. 2.1 Опис продукту та вимог. 2.2 Аналіз хмарних платформ. 2.3 Системи передачі та опрацювання. потокових даних. 2.4 Методи та моделі машинного аналізу зображень. 2.5 Висновок. 3 Проектування та реалізація продукту. 3.1 Пошук актантів та варіантів. 3.2 Проектування архітектури продукту. 3.3 Опис програмних рішень. 3.4 Опис інтерфейсу веб-клієнта. 3.5 Розгортання системи та допоміжних сервісів на хмарній платформі. 3.6 Висновок. 4 Охорона праці та безпека в надзвичайних ситуація. 4.1 Характеристика НС воєнного характеру. 4.2 Міжнародні норми соціальної відповідальності. Стандарт SA 8000 «Соціальна відповідальність». 4.3 Підвищення стійкості роботи підприємств приладобудівної галузі в воєнний час. 4.4 Оцінка стійкості роботи економіки до впливу поражаючих факторів ядерної зброї. 4.5 Висновок. Висновки. Перелік джерел. Додатки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Мацюк О.В., доцент		
Безпека в надзвичайних ситуаціях	Клепчик В.М., ст. викладач		

7. Дата видачі завдання 14 листопада 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	14.11.2022-15.11.2022	*Виконано*
2.	Підбір наукових джерел про хмарні рішення,	16.11.2022-27.11.2022	*Виконано*
	архітектуру та методи машинного аналізу зображень		
3.	Опрацювання наукових публікацій та збір даних по	28.11.2022-25.12.2022	*Виконано*
	темі роботи		
4.	Виконання дослідження згідно мети кваліфікаційної	09.01.2023-12.03.2023	*Виконано*
	роботи		
5.	Оформлення розділу «Аналіз предметної області з	13.03.2023-19.03.2023	*Виконано*
	хмарної архітектури та комп’ютерного бачення»		
6.	Оформлення розділу «Аналіз систем обробки потоку	20.03.2023-26.03.2023	*Виконано*
	даних та моделей машинного аналізу зображень»		
7.	Оформлення розділу «Проектування та реалізація	27.03.2023-02.04.2023	*Виконано*
	продукту»		
8.	Виконання завдання до підрозділу «Охорона праці»	10.04.2023-16.04.2023	*Виконано*
9.	Виконання завдання до підрозділу «Безпека в	17.04.2023-23.04.2023	*Виконано*
	надзвичайних ситуаціях»		
10.	Оформлення кваліфікаційної роботи	24.04.2023-30.04.2023	*Виконано*
11.	Нормоконтроль	01.05.2023-07.05.2023	*Виконано*
12.	Перевірка на плагіат	08.05.2023	*Виконано*
13.	Попередній захист кваліфікаційної роботи	15.05.2023	*Виконано*
14.	Захист кваліфікаційної роботи	23.05.2023	

Студент _____ Лісовський В. В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Никитюк В. В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка програмного продукту на основі хмарної архітектури для машинного аналізу зображень (комплексна тема) // Кваліфікаційна робота освітнього рівня «Магістр» // Зелінський Андрій Олегович, Лісовський Владислав Володимирович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СНнм-61 // Тернопіль, 2023 // С. 114, рис. – 43, табл. – 3, кресл. – 0, додат. – 2, бібліогр. – 58.

Ключові слова: хмара, зображення, машинне навчання, kinesis, розпізнавання об'єктів, сервер, комп'ютерне бачення.

Кваліфікаційна робота присв'ячена розробці прототип продукту на основі хмарної архітектури, який буде виконувати машинний аналіз зображень з використанням нейронних мереж. У даній кваліфікаційній роботі було досліджено велику кількість різних підходів до роботи з потоковими даними, а саме потоковим відеорядом. Розроблено програмне рішення стійке до навантажень та оптимізоване для роботи з великою кількістю користувачів одночасно. Основа програмного рішення складається з передових технологій машинного навчання та комп'ютерного бачення. Проведено інтеграцію сервісу із хмарним рішенням Amazon Web Services.

В першому розділі кваліфікаційної роботи освітнього рівня «Магістр» проаналізовано використання хмарної архітектури для обробки зображень, ознайомлено з доступними методами опрацювання потоку даних, висвітлено можливі джерела безперервного потоку зображень. Також досліджено підхід інженерії даних для зберігання і опрацювання великих об'ємів потокових даних і поверхнево розглянуто глибоке навчання та комп'ютерне бачення для обробки зображень.

В другому розділі кваліфікаційної роботи проведено опис продукту та його вимог, проаналізовано хмарні рішення Amazon Web Services та Google Cloud Platform, проведено їхнє порівняння, та було обрано AWS як основне хмарне рішення для розгортання продукту. Досліджено популярні системи передачі та опрацювання потокових даних, обрано AWS Kinesis головним сервісом для опрацювання відеоряду, описано методи машинного аналізу зображень, визначено моделі для розпізнавання об’єктів, розпізнавання лиць, оцінки пози рук.

В третьому розділі кваліфікаційної роботи проведено пошук актантів та варіантів використання продукту, спроектовано архітектуру продукту, сформовано опис програмних рішень сервера та веб-клієнта, описано інтерфейс веб-клієнта та розгорнуто опрацювання безперервного потоку даних та встановлено розроблену систему у хмарному рішенні AWS.

ANNOTATION

Software Development Based On Cloud Architecture for Analysis of Machine Images // Qualification work of the educational level "Master" // Zelinskyi Andrii, Lisovskyi Vladyslav // Ternopil Ivan Pulyuy National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science, SNnm-61 group // Ternopil, 2023 // P. 114, fig. - 43, tables - 3, chair. - 0, annexes - 2, references. - 58.

Key words: cloud, images, machine learning, kinesis, object detection, server, computer vision.

Thesis is devoted to the development of an a cloud architecture-based product prototype that will perform machine image analysis using neural networks. A large number of different approaches to working with streaming data, namely video streaming, have been explored in this thesis. The software solution is load-resistant and optimized for working with a large number of users at the same time. The basis of the software solution consists of advanced machine learning and computer learning technologies. The service has been integrated with the Amazon Web Services cloud solution.

In the first section of the thesis, the use of cloud architecture for image processing is analyzed, the available methods of data flow processing are introduced, and the possibility of a continuous image flow source is highlighted. A data engineering approach for storing and processing large volumes of streaming data and surface-based deep learning and computer vision for image processing are also explored.

The second part of the qualification work described the product and its requirements, analyzed cloud solutions Amazon Web Services and Google Cloud Platform, compared them, and selected AWS as the main cloud solution for product

deployment. Popular streaming data transmission and processing systems were studied, AWS Kinesis was chosen as the main service for processing video sequences, methods of machine image analysis were described, models for object recognition, face recognition, and hand position estimation were defined.

In the third section of the qualification work, a search for actants and options for using the product was carried out, the product was designed, a description of the server and web client software solutions was formed, the web client interface was described, and continuous data flow processing was deployed, and the developed system was installed in the cloud architectural extension of AWS.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ACID (англ. Atomicity, Consistency, Isolation, Durability) – набір властивостей, що гарантують надійну роботу транзакцій бази даних: атомарність, узгодженість, ізольованість, довговічність.

API (англ. Application Programming Interface) – прикладний програмний інтерфейс.

AWS (англ. Amazon Web Services) – дочірньою компанією Amazon.com, що надає платформу хмарних обчислень в оренду.

CNN (англ. Convolutional Neural Network) – згорткова нейронна мережа, клас глибинних штучних нейронних мереж.

EC2 (англ. Amazon Elastic Compute Cloud) – вебсервіс, котрий надає обчислювальні потужності в хмарі.

GCP (англ. Google Cloud Platform) – запропонований компанією Google набір хмарних служб, які виконуються на тій же самій інфраструктурі, яку Google використовує для своїх продуктів.

IoT (англ. Internet of Things) – Інтернет речей.

IoU (англ. Intersection over Union) – пересічення по об’єднанні, також відомий як коефіцієнт Жаккара.

YOLO (англ. You Only Look Once) – сімейство архітектури моделей нейронних мереж для задач розпізнавання об’єктів, що використовують одноетапний прохід у розпізнаванні.

ШІ – Штучний інтелект.

ШНМ – Штучні нейронні мережі.

ЗМІСТ

ВСТУП

Актуальність теми. У міру вдосконалення технологій з’являється все більше нових інформаційних технологій обробки машинної обробки зображень, надаючи велику кількість нових можливостей у розробці. Є велика кількість можливих сфер для їх практичного застосування, проте багато з них стикаються з проблемою використання ресурсів для повної реалізації потенціалу таких систем. Тому розроблення системи на основі хмарної архітектури для машинного аналізу зображень із використанням сучасних інформаційних технологій аналітичного опрацювання є актуальним напрямком сучасних наукових досліджень.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи освітнього рівня «Магістр» є підвищення рівня повноти подання інформації щодо поєднання методів машинного аналізу зображень та сервісів на хмарній архітектурі. Для досягнення поставленої мети потрібно виконати ряд завдань, зокрема:

– Проаналізувати стан досліджень в області хмарних платформ, методів опрацювання та обробки потоку зображень.

– Дослідити існуючі на даний час методи машинного розпізнавання зображень та сутностей на ньому, а також хмарних платформ.

– Проаналізувати та порівняти методи машинного розпізнавання зображень та сутностей на ньому, таких як об’єктів, лиць та рук, і сервісів розгортання систем на хмарі.

– Розробити прототип продукту на основі хмарної архітектури, який буде виконувати машинний аналіз зображень з використанням нейронних мереж**.**

Об’єкт дослідження. Процеси аналізу зображень машиною та розгортання системи на хмарі.

Предмет дослідження. Методи аналізу та обробки зображень штучним інтелектом і хмарні платформи.

Наукова новизна одержаних результатів кваліфікаційної роботи полягає у тому, що отримав подальший розвиток комбінація різноманітних методів аналізу зображень разом з їх розгортанням на хмарній архітектурі.

Практичне значення одержаних результатів. Виконано прототипування програмного продукту на основі хмарної архітектури для машинного аналізу зображень.

Апробація результатів магістерської роботи. Основні результати проведених досліджень обговорювались на XI Міжнародної науково-практичної конференції молодих учених та студентів «Актуальні задачі сучасних технологій» Тернопільського національного технічного університету імені Івана Пулюя (м. Тернопіль, 2022 р.).

Публікації. Основні результати кваліфікаційної роботи опубліковано у двох працях конференції (Див. додатки А).

Структура й обсяг кваліфікаційної роботи. Кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків, списку літератури з 58 найменувань та 2 додатків. Загальний обсяг кваліфікаційної роботи складає 114 сторінок, з них 84 сторінки основного тексту, який містить 43 рисунки та 3 таблиць.

Авторські внески. Дослідження проводилось двома розробниками, кожен з яких виконав:

1. Зелінський А. О. – досліджував методи та системи потокової передачі зображень та виконав розгортання системи і допоміжних сервісів на хмарній платформі. Перелік параграфів авторського внеску – 1.1 Аналіз потокової передачі зображень, 1.2 Джерела безперервного потоку зображень, 1.3 Інженерія даних, 2.1 Опис продукту та вимог, 2.2 Аналіз хмарних платформ, 2.3 Системи передачі та опрацювання потокових даних, 3.1 Пошук актантів та варіантів використання, 3.2 Проектування архітектури продукту, 3.5

Розгортання системи та допоміжних сервісів на хмарній платформі, 4.1 Характеристика НС воєнного стану, 4.2 Оцінка стійкості роботи економіки до впливу поражаючих факторів ядерної зброї.

2. Лісовський В. В. – проводив аналіз методів та моделей машинного аналізу зображень та виконав розробку сервера обробки зображень. Перелік параграфів авторського внеску – 1.4 Машинний аналіз та обробка зображень, 2.1 Опис продукту та вимог, 2.4 Методи та моделі машинного аналізу зображень, 3.1 Пошук актантів та варіантів використання, 3.2 Проектування архітектури продукту, 3.3 Опис програмних рішень, 3.4 Опис інтерфейсу веб-клієнта, 4.3 Міжнародні норми соціальнох відповідальності. Стандарт SA 8000 «Соціальна відповідальність», 4.4 Підвищення стійкості роботи підприємств приладобудівної галузі в воєнний час.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ З ХМАРНОЇ АРХІТЕКТУРИ ТА КОМП'ЮТЕРНОГО БАЧЕННЯ

1.1 Аналіз потокової передачі зображень

Потокова передача зображень – це техніка, яка використовується для доставки зображень через мережу в режимі реального часу. Вона передбачає безперервну передачу зображень у вигляді відеопотоку, що дозволяє користувачам переглядати зображення та взаємодіяти з ними під час їх передачі. Зображення зазвичай стискаються, щоб зменшити вимоги до пропускної здатності, а використовуваний формат стиснення залежить від конкретного випадку використання [10].

Потокова передача зображень необхідна для різних програм, включаючи відеоспостереження, віддалений моніторинг і відеоконференції. У системах відеоспостереження потокова передача зображень дозволяє персоналу служби безпеки контролювати кілька камер у режимі реального часу, покращуючи обізнаність про ситуацію та час реагування. Програми віддаленого моніторингу використовують потокове передавання зображень для передачі живого відео з віддалених місць, таких як станції моніторингу погоди, нафтові вишки або будівельні майданчики. Програми для відеоконференцій використовують потокове зображення для передачі відео та аудіо потоків у реальному часі, що дозволяє учасникам спілкуватися та співпрацювати з різних місць.

Розробники, які працюють із потоковою передачею зображень, можуть зіткнутися з кількома проблемами. Однією з найважливіших проблем є затримка, яка може спричинити затримки в доставці зображень і вплинути на якість потоку. Щоб зменшити затримку, розробникам може знадобитися оптимізувати мережеву інфраструктуру, наприклад використовувати швидші з'єднання, або реалізовувати мережеву інфраструктуру використовуючи

хмарні рішення які надають пряме та швидке з’єднання різних куточків світу, або реалізувати механізми кешування, тощо [10, 11].

Інша проблема полягає в забезпеченні безпеки потоку зображень, особливо при передачі конфіденційної інформації.

Розробникам може знадобитися застосувати шифрування, контроль доступу та інші заходи безпеки, щоб захистити потік від несанкціонованого доступу або перехоплення. Крім того, розробникам необхідно враховувати масштабованість системи, особливо при підтримці великої кількості користувачів або пристроїв. Для цього може знадобитися балансування навантаження, кластеризація або інші методи, щоб система могла обробляти великі обсяги трафіку.

1.1.1 Використання хмарної архітектури

Хмарні платформи стали популярним вибором для доставки додатків потокового передавання зображень і керування ними. Із зростанням високошвидкісного підключення до Інтернету та зростаючим попитом на спілкування на основі зображень у реальному часі хмарні платформи пропонують надійне, масштабоване та економічно ефективне рішення для доставки зображень через мережу.

Хмарні платформи можуть надати розробникам доступ до різноманітних інструментів і сервісів, які можуть допомогти їм оптимізувати їхні додатки потокового передавання зображень, наприклад, віртуальні машини, контейнери, безсерверні архітектури, розширену аналітику та машинне навчання. У цьому контексті важливо розуміти, як хмарні платформи можуть допомогти з потоковою передачею зображень, переваги, які вони пропонують, і проблеми, з якими розробники можуть зіткнутися під час роботи з ними. Далі будуть розглянуті способи, якими хмарні платформи можуть допомогти з потоковим передаванням зображень, і як розробники можуть використовувати

хмарні сервіси для створення високопродуктивних масштабованих програм для потокового передавання зображень.

Хмарні обчислення – це модель доступу до великого об’єднаного пулу обчислювальних ресурсів, яка забезпечує постійний і комплексний доступ через мережу. Хмарні рішення самі відповідають за доступ до власних сервісів. Ресурси можуть включати мережі зв’язку, сервери, засоби зберігання даних, додатки та служби, і можуть бути надані або звільнені автоматично з мінімальним часом, керуванням і прямим контактом з постачальником [5].

Міжнародні стандарти NIST включають такі ключові особливості хмарних обчислень [6, 7]:

1. Загальний пул багатозамовницьких ресурсів – постачальник послуг об'єднує ресурси для обслуговування великої кількості споживачів в єдиний пул для перерозподілу енергії між споживачами в умовах постійної зміни попиту на енергію; у цьому випадку споживачі контролюють лише основні параметри послуги (наприклад, обсяг даних, швидкість доступу), але фактичний розподіл ресурсів, що надаються споживачеві, здійснюється провайдером (у деяких випадках споживачі все ще можуть контролювати деякі фізичні параметри перепризначення, наприклад, вказувати потрібний центр обробки даних через географічну близькість).

2. Самообслуговування на вимогу – споживач може самостійно визначати та змінювати обчислювальні потреби, такі як час сервера, доступ до даних і швидкість обробки, а також обсяг збережених даних, без взаємодії з представником постачальника послуг.

3. Миттєва еластичність – послуги можна надавати, розширювати, скорочувати в будь-який момент без додаткових витрат на взаємодію з провайдером, як правило, в автоматичному режимі.

4. Універсальний доступ до мережі – послуги доступні споживачам через мережу передачі даних, незалежно від кінцевого пристрою.

5. Облік споживання послуг – постачальник послуг автоматично розраховує спожиті ресурси на певному рівні абстракції (наприклад, обсяг збережених даних, пропускна здатність, кількість користувачів, кількість транзакцій) і на основі цих даних оцінює обсяг послуг, що надаються споживачам.

Модель хмарного розгортання – це конкретна конфігурація налаштувань середовища, яка включає доступність, право власності на інфраструктуру розгортання та ємність зберігання. Різні типи хмарних розгортань відрізняються тим, хто контролює інфраструктуру та де ця інфраструктура знаходиться [5, 8].

Існує 4 типи моделей розгортання:

1. Публічна хмара – це хмарна інфраструктура, призначена для вільного використання широким загалом. Загальнодоступною хмарою можуть володіти та керувати комерційні, академічні (освітні та дослідницькі) або державні організації (або будь-яка їх комбінація). Публічна хмара знаходиться під юрисдикцією постачальника хмарних послуг.

2. Приватна хмара – хмарна інфраструктура, призначена для використання виключно організацією, яка включає кількох користувачів (наприклад, підрозділи). Приватна хмара може належати, управлятися та експлуатуватися самою організацією або третьою стороною (або їх поєднанням). Така хмара може фізично перебувати як у межах юрисдикції власника, так і за його межами.

3. Гібридна хмара – це хмарна інфраструктура, що складається з двох або більше різних хмарних інфраструктур, які залишаються окремими об’єктами, але пов’язані між собою стандартизованими або приватними технологіями, які забезпечують переносимість даних і додатків (наприклад, використовуючи загальнодоступні хмарні ресурси для балансування навантаження між хмарами).

4. Хмара спільноти – надає ексклюзивне право на використання хмарних обчислень відповідним спільнотам, які мають спільні завдання.

Компанії, що надають хмарні послуги, надають свої послуги відповідно до трьох моделей обслуговування, також відомих як три рівні абстракції (див. рис. 1.1).

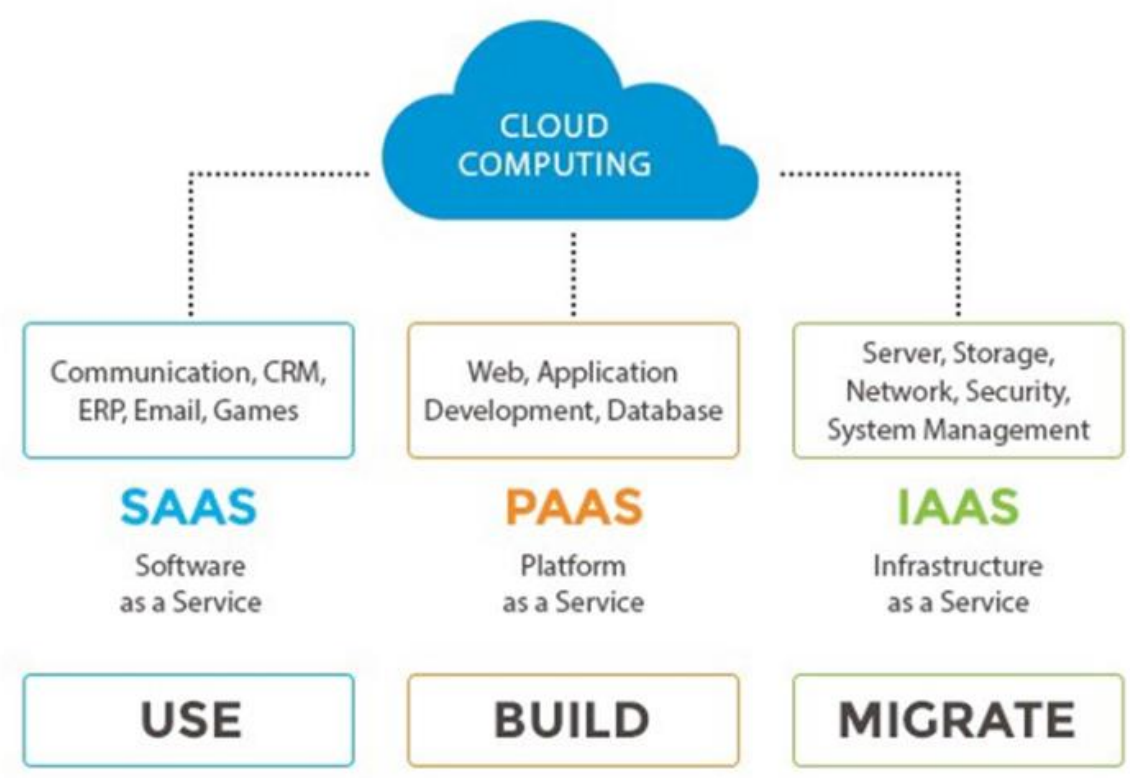


Рисунок 1.1 – Три типи моделей обслуговування

Інфраструктура як послуга (англ. Infrastructure as a Service, IaaS) дозволяє компаніям отримувати доступ до критично важливих веб-ресурсів, таких як простір для зберігання, сервери та підключення, без необхідності самостійно купувати цю інтернет-інфраструктуру та керувати нею. В рамках цієї послуги споживачам надаються віртуальні сервери з певною обчислювальною потужністю та пам’яттю, а всі додаткові «апаратні» ресурси (апаратні засоби) несе провайдер. Під час свого візиту постачальник послуг встановлює на ці ресурси програмне забезпечення, необхідне для створення віртуальних машин, але не використовується для встановлення та підтримки програмного

забезпечення користувача. При цьому провайдер контролює лише фізичну та віртуальну інфраструктуру, а контроль над розміщенням програмного забезпечення та налаштуваннями покладається на споживача.

Платформа як послуга (англ. Platform as a Service, PaaS) – це служба, яка дозволяє створювати та розгортати програми в хмарних інфраструктурах IaaS. Експерти створюють платформу, яка пропонує масштабованість і передбачувані витрати. За допомогою PaaS можна запустити свою програму без жодних зусиль, окрім базової розробки цієї міграції. Крім того, PaaS дозволяє масштабувати проект, після чого він базується на хмарних обчисленнях. Головні переваги PaaS – масштабованість і передбачувана вартість.

У цьому випадку хмарний провайдер надає доступ до операційних систем, засобів розробки та тестування, систем управління базами даних. Провайдер не тільки контролює сервери, системи зберігання даних і обчислювальну потужність, але також пропонує користувачам вибір певних платформ і інструментів для управління ними.

SaaS (Software as a Service, укр. програмне забезпечення як послуга) – це бізнес-модель розгортання та впровадження програмного забезпечення, де постачальник (постачальник) розробляє програму, ліцензує її, керує нею та надає споживачам (бізнес-клієнтам) доступ до програмного забезпечення в Інтернеті. SaaS – це програмне забезпечення плюс послуга, програмне забезпечення як послуга та на вимогу.

Програмне забезпечення як постачальник послуг забезпечує клієнту реалізацію бізнес-функцій, функціональність бізнес-додатків, вирішує питання інтеграції свого сервісу в IT-систему споживача, бере на себе всі функції з розробки та підтримки рішень та забезпечення їх масштабування [8, 9].

1.1.2 Методи та системи зберігання та обробки потоку даних

Існує кілька методів і систем, які використовуються для збору й обробки потокових даних у реальному часі. Ось кілька прикладів:

1. Системи черги повідомлень: системи черги повідомлень, такі як RabbitMQ, Apache ActiveMQ і Apache Pulsar, можна використовувати для збору та буферизації потокових даних у реальному часі [23]. Ці системи забезпечують надійну, масштабовану та розподілену інфраструктуру обміну повідомленнями, яка може допомогти роз’єднати виробників і споживачів даних.

2. Фреймворки потокової обробки: інфраструктури потокової обробки, такі як Apache Kafka, Apache Flink і Apache Spark Streaming, забезпечують розподілену, масштабовану та відмовостійку платформу для обробки великої кількості потокових даних у реальному часі [24]. Ці структури дозволяють розробникам визначати конвеєри обробки даних у реальному часі, які можуть фільтрувати, перетворювати та агрегувати дані в режимі реального часу.

3. Хмарні платформи. Хмарні платформи, такі як Amazon Kinesis, Azure Stream Analytics і Google Cloud Pub/Sub, надають керовані послуги для збору й обробки потокових даних у реальному часі. Ці платформи забезпечують масштабоване, надійне та економічно ефективне рішення для обробки даних у реальному часі, усуваючи потребу в управлінні інфраструктурою [25].

4. Платформи IoT: платформи IoT, такі як AWS IoT, Azure IoT і Google Cloud IoT, забезпечують уніфіковану платформу для збору та обробки потокових даних у реальному часі з пристроїв IoT. Ці платформи надають такі функції, як керування пристроями, безпека та аналітика, які можуть допомогти у створенні додатків Інтернету речей у реальному часі.

Загалом вибір методу та системи залежить від різних факторів, таких як тип і обсяг даних, бажана затримка та пропускна здатність, а також доступність інфраструктури та ресурсів. Вибравши правильний метод і систему, організації можуть створювати масштабовані, надійні та продуктивні програми обробки

даних у режимі реального часу, які можуть надавати цінність і статистику в реальному часі.

1.2 Джерела безперервного потоку зображень

Дані потокового відео генеруються різними джерелами, такими як платформи потокового передавання, камери безпеки та спостереження, дрони та роботи, а також пристрої IoT. Ці джерела надають відеодані в реальному часі, які можна використовувати для різних програм, таких як моніторинг, аналіз і прийняття рішень. Використовуючи ці джерела даних, організації можуть створювати програми потокового відео в реальному часі, які можуть надавати цінність і статистику в реальному часі. Розуміння різних джерел даних потокового відео є важливим для організацій, які хочуть використовувати відеодані в реальному часі, щоб отримати конкурентну перевагу та покращити свою діяльність.

Інтернет речей (IoT) є одним з таких джерел генерування безперевного потоку даних, а саме зображень. Він описує фізичні об’єкти (або групи таких об’єктів) із датчиками, можливостями обробки, програмним забезпеченням та іншими технологіями, які підключаються й обмінюються даними з іншими пристроями та системами через Інтернет чи інші комунікаційні мережі. Термін «Інтернет речей» вважався неправильним, оскільки пристрої не потребують підключення до загальнодоступного Інтернету, їх потрібно лише підключити до мережі та індивідуальної адреси [26].

Галузь розвивалася завдяки конвергенції багатьох технологій, включаючи повсюдне обчислення, товарні датчики, потужніші вбудовані системи та машинне навчання. Традиційні галузі вбудованих систем, бездротових сенсорних мереж, систем управління та автоматизації (включаючи автоматизацію дому та будівель) самостійно та спільно створюють Інтернет речей. На споживчому ринку технологія IoT є синонімом продуктів, пов’язаних

з концепцією «розумного дому», включаючи пристрої та прилади (такі як освітлювальні прилади, термостати, системи домашньої безпеки, камери та інші побутові прилади), які підтримують одну або кілька загальних екосистем. І ними можна керувати за допомогою пристроїв, підключених до цієї екосистеми, таких як смартфони та розумні колонки. IoT також використовується в системах охорони здоров'я [27].

Існує низка занепокоєнь щодо ризиків, пов'язаних із розвитком технологій і продуктів IoT, особливо в сферах конфіденційності та безпеки, тому промисловість і уряд почали вживати заходів для вирішення цих проблем, включаючи міжнародні та місцеві події. Стандарти, рекомендації та нормативна база. Основні галузі інтернету речей зображені на рисунку 1.2.

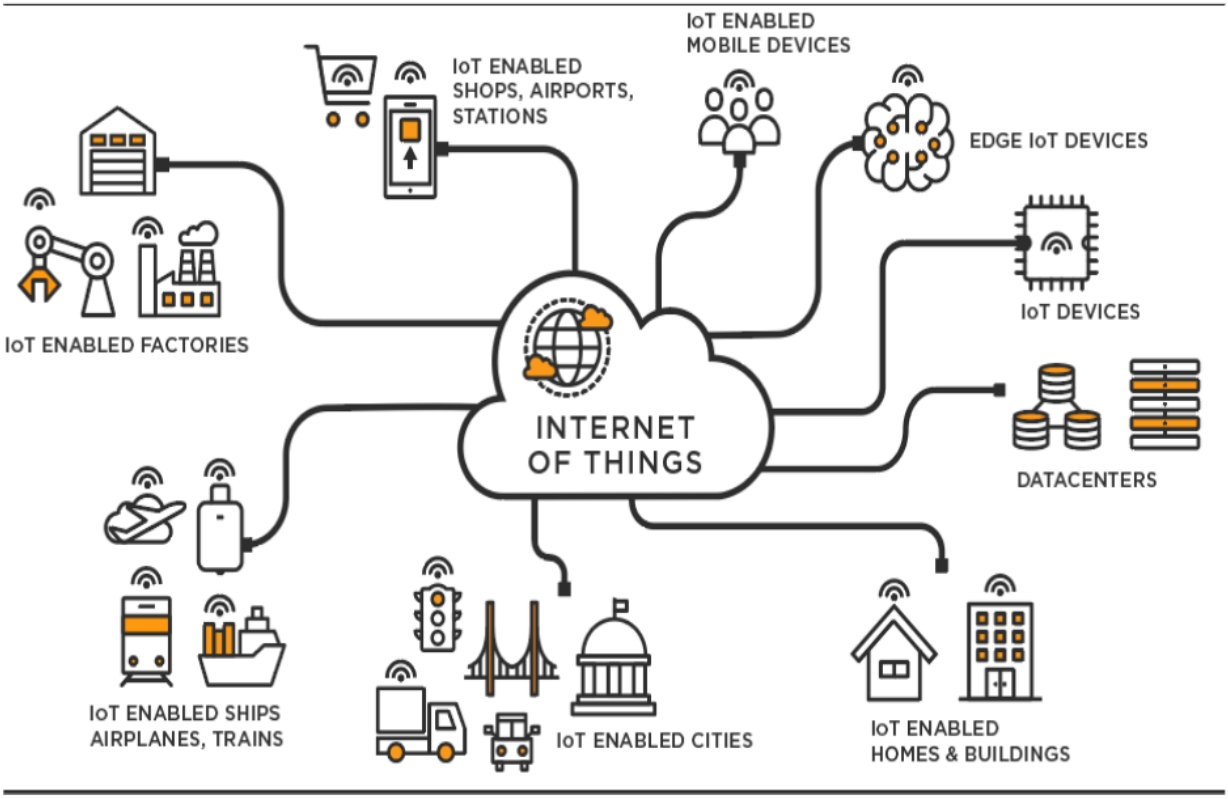


Рисунок 1.2 – Галузі, в яких використовується інтернет речей

Основна концепція мережі інтелектуальних пристроїв була описана як «*[переміщення] невеликих пакетів даних до великого набору вузлів, щоб*

інтегрувати та автоматизувати все, від побутової техніки до цілих фабрик» [13]. А сам термін вперше був сформульований як «*Інтернет речей, або IoT, – це інтеграція людей, процесів і технологій із підключеними пристроями та датчиками для забезпечення віддаленого моніторингу, стану, маніпуляції та оцінки тенденцій таких пристроїв*» [12].

Основна тема Інтернету речей полягає в тому, щоб вбудувати мобільні отримувачі малого радіусу дії в різні гаджети та предмети повсякденної потреби, щоб уможливити нові форми спілкування між людьми та речами, а також між самими речами.

1.2.1 Домашня автоматизація

Домашня автоматизація або домотика – це автоматизація будівництва будинку, яку називають розумним будинком. Система домашньої автоматизації відстежуватиме та/або керуватиме такими атрибутами будинку, як освітлення, клімат, розважальні системи та побутова техніка. Це також може включати системи безпеки будинку, такі як контроль доступу та системи сигналізації. Підключені до Інтернету домашні пристрої є важливою складовою Інтернету речей [14].

Система домашньої автоматизації зазвичай підключає керовані пристрої до центрального концентратора розумного будинку (іноді його називають шлюзом). Інтерфейс користувача для керування системою використовує або настінні термінали, планшетні або настільні комп’ютери, програму для мобільного телефону або веб-інтерфейс, який також може бути доступний за межами сайту через Інтернет.

Хоча існує багато конкуруючих постачальників, все більше зусиль докладають системи з відкритим кодом. Однак існують проблеми з поточним станом домашньої автоматизації, включаючи відсутність стандартизованих

заходів безпеки та припинення використання старих пристроїв без зворотної сумісності.

Домашня автоматизація має високий потенціал для обміну даними між членами родини або довіреними особами для особистої безпеки та може призвести до заходів з енергозбереження з позитивним впливом на навколишнє середовище в майбутньому.

Домашня автоматизація поширена в різних сферах, зокрема:

– Опалення, вентиляція та кондиціонування повітря: можна дистанційно керувати всіма домашніми енерго моніторами через Інтернет, використовуючи простий і зручний інтерфейс користувача.

– Система керування освітленням: розумна мережа, яка включає зв’язок між різними входами та виходами системи освітлення за допомогою одного чи кількох центральних обчислювальних пристроїв.

– Контроль приладів та інтеграція з інтелектуальною мережею та інтелектуальним лічильником, використовуючи переваги, наприклад, високої потужності сонячних панелей у середині дня для роботи пральних машин.

– Домашні роботи та безпека: система домашньої безпеки, інтегрована з системою домашньої автоматизації, може надавати додаткові послуги, такі як дистанційне спостереження за камерами безпеки через Інтернет або контроль доступу та центральне замикання всіх периметральних дверей і вікон.

– Виявлення витоків, детектори диму та CO.

– Системи внутрішнього позиціонування.

– Домашня автоматизація для людей похилого віку та інвалідів.

– Догляд за домашніми тваринами та немовлятами, наприклад, відстеження переміщень домашніх тварин і немовлят і контроль прав доступу домашніх тварин.

– Контроль якості повітря (всередині та на вулиці). Наприклад, Air Quality Egg використовується людьми вдома для моніторингу якості повітря та рівня забруднення в місті та створення карти забруднення.

– Розумна кухня, з інвентарем для холодильника, готовими програмами приготування їжі, спостереженням за приготуванням їжі тощо.

– Пристрої голосового керування, як-от Amazon Alexa або Google Home, які використовуються для керування побутовою технікою чи системами.

У 2022 році ринок домашньої автоматизації становив 64 мільярди доларів США, а в 2028 році, за прогнозами, зросте до понад 163 мільярдів доларів США.

1.3 Інженерія даних

Інженерія даних відіграє вирішальну роль у програмах потокової передачі даних у реальному часі. Потокове передавання даних у реальному часі генерує великі обсяги даних, які необхідно обробляти, аналізувати та виконувати дії в реальному часі. Інженерія даних – це дисципліна, яка проектує та реалізує конвеєри даних, архітектури та процеси, необхідні для обробки цих складних вимог до даних [30].

У потоковій передачі даних у реальному часі інженерія даних відповідає за інтеграцію даних із різних джерел, обробку великих обсягів даних у режимі реального часу, забезпечення якості даних і розробку масштабованих архітектур, які можуть впоратися зі змінністю обсягу та швидкості даних. Без ефективної обробки даних потокові програми в реальному часі можуть стати надто важкими, неефективними та ненадійними.

Роль розробки даних у потоковій передачі даних у реальному часі полягає в тому, щоб розробити та реалізувати конвеєри даних, які можуть впоратися зі складнощами потокової передачі даних у реальному часі, забезпечити точність і надійність даних і надати корисну інформацію в режимі реального часу [31].

Інженерія даних стосується побудови систем, які дозволяють збирати та використовувати дані. Ці дані зазвичай використовуються для подальшого аналізу та дослідження даних; яка часто передбачає машинне навчання.

Роблення даних придатними для використання зазвичай вимагає значних обчислень і зберігання, а також обробки та очищення даних. Інженер даних – це тип інженера програмного забезпечення, який створює ETL-конвеєри великих даних для керування потоком даних через організацію. Інженери з обробки даних зазвичай мають досвід розробки програмного забезпечення та добре володіють такими мовами програмування, як Java, Python, Scala та Rust. Вони будуть більше знайомі з базами даних, архітектурою, хмарними обчисленнями та гнучкою розробкою програмного забезпечення.

Інженерія даних відіграє вирішальну роль у програмах потокової передачі даних у реальному часі. Ось кілька причин, чому і як слід використовувати інженерні дані для потокової передачі даних у реальному часі [29]:

1. Інтеграція даних. Програми потокової передачі даних у реальному часі часто вимагають інтеграції з різними джерелами даних і системами. Інженерія даних може бути використана для розробки та реалізації конвеєрів даних, необхідних для прийому та інтеграції даних у реальному часі з цих джерел.

2. Обробка даних. Програми потокової передачі даних у реальному часі генерують великі обсяги даних, які необхідно обробляти та аналізувати в режимі реального часу. Інжиніринг даних можна використовувати для проектування та реалізації конвеєрів обробки даних, які можуть обробляти цей обсяг даних і швидкість.

3. Якість даних. Програми для потокової передачі даних у реальному часі вимагають високоякісних даних для надання точної та надійної інформації. Інженерія даних може бути використана для розробки та впровадження процесів якості даних, які гарантують точність і надійність даних, що надходять і обробляються.

4. Масштабованість: Програми для потокової передачі даних у реальному часі повинні мати можливість обробляти різні обсяги та швидкості даних. Інженерія даних може бути використана для проектування та

впровадження масштабованих архітектур даних, які можуть обробляти ці зміни в режимі реального часу.

Усі сервіси які активно використовуються в інженерії даних зображені на рисунку 1.3.

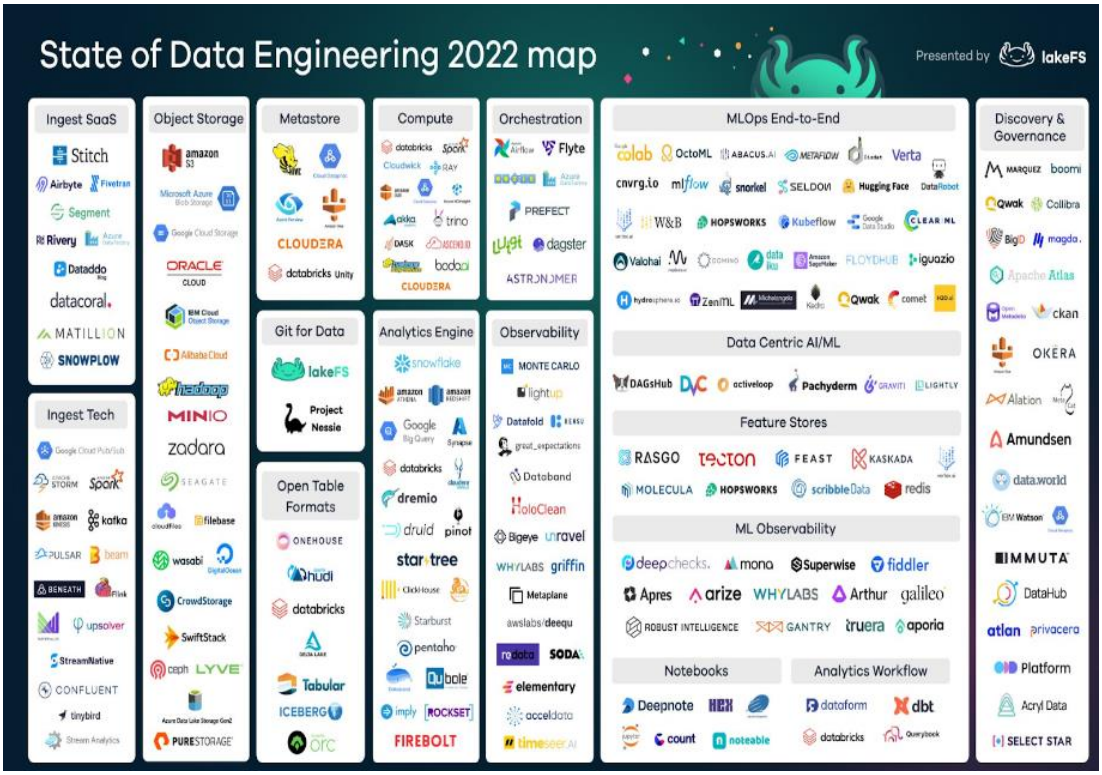


Рисунок 1.3 – Сервіси та технології які використовують інженери даних

Основні напрямки інженерії даних включають:

– Обчислення. Високопродуктивне обчислення має вирішальне значення для обробки та аналізу даних. Одним із особливо поширених підходів до обчислень для розробки даних є програмування потоку даних, у якому обчислення представлено у вигляді спрямованого графа (граф потоку даних); вузли – це операції, а ребра – потік даних. Серед популярних реалізацій – Apache Spark і TensorFlow для глибокого навчання. Більш пізні реалізації, такі як Differential/Timely Dataflow, використовували інкрементні обчислення для значно ефективнішої обробки даних.

– Зберігання. Дані зберігаються різними способами, одним із ключових вирішальних факторів є спосіб їх використання.

– Бази даних. Якщо дані структуровані та потрібна певна форма онлайн-обробки транзакцій, то зазвичай використовуються бази даних. Спочатку використовувалися в основному реляційні бази даних із сильними гарантіями правильності транзакцій ACID; більшість реляційних баз даних використовують SQL для своїх запитів. Однак із зростанням обсягу даних у 2010-х роках бази даних NoSQL також стали популярними, оскільки вони горизонтально масштабувалися легше, ніж реляційні бази даних, відмовившись від гарантій транзакцій ACID, а також зменшивши невідповідність об’єктно-реляційного імпедансу. Зовсім недавно бази даних NewSQL, які намагаються дозволити горизонтальне масштабування, зберігаючи гарантії ACID, стали популярними.

– Сховища даних. Якщо дані структуровані та потрібна онлайн-аналітична обробка (але не онлайн-обробка транзакцій), то основним вибором є сховища даних. Вони дозволяють аналізувати дані, видобуток і штучний інтелект у набагато більших масштабах, ніж це можуть дозволити бази даних, і справді дані часто надходять із баз даних у сховища даних. Бізнес-аналітики, інженери з обробки даних і вчені з обробки даних можуть отримати доступ до сховищ даних за допомогою таких інструментів, як SQL або програмне забезпечення бізнес-аналітики.

– Файли. Якщо дані менш структуровані, то часто вони просто зберігаються у вигляді файлів. Є кілька варіантів: файлові системи представляють дані ієрархічно у вкладених папках; блокове сховище розбиває дані на частини регулярного розміру; це часто збігається з (віртуальними) жорсткими дисками або твердотільними накопичувачами; об’єктне сховище керує даними за допомогою метаданих; часто кожному файлу призначається ключ, такий як UUID.

– Управління. Кількість різних процесів обробки даних і місць зберігання може швидко стати надзвичайною. Це спонукає до використання системи управління робочим процесом (наприклад, Airflow), яка дозволяє вказувати, створювати та контролювати завдання даних. Завдання часто специфікуються як спрямований ациклічний граф [28].

Загалом, інженерія даних має важливе значення для створення додатків потокового передавання даних у реальному часі, які можуть надавати цінність і статистику в реальному часі. Використовуючи потужність інженерії даних, організації можуть створювати конвеєри даних у реальному часі, які можуть впоратися зі складнощами потокової передачі даних у реальному часі та надавати практичну інформацію, яка сприяє успіху бізнесу.

1.4 Машинний аналіз та обробка зображень

Наука про дані та глибоке навчання відіграють вирішальну роль в обробці й аналізі зображень машинами. Обробка та аналіз зображень є складним завданням, яке вимагає виділення значущих характеристик із великих обсягів даних зображення. Наука про дані та методи глибокого навчання дозволяють машинам навчатися на даних зображень, розпізнавати шаблони та робити прогнози [32].

Методи науки про дані, такі як інтелектуальний аналіз даних, статистичний аналіз і алгоритми машинного навчання, можна використовувати для попередньої обробки даних зображення, вилучення значущих ознак і перетворення даних зображення у формат, який можна використовувати для аналізу. Глибоке навчання, підмножина машинного навчання, використовує нейронні мережі, щоб вивчати дані зображень, розпізнавати шаблони та робити прогнози.

Роль науки про дані та глибокого навчання в обробці та аналізі зображень машинами полягає в тому, щоб дозволити машинам розпізнавати об’єкти,

класифікувати зображення та робити прогнози на основі даних зображень. Ці методи мають широке застосування в таких сферах, як охорона здоров’я, виробництво та безпека, де дані зображень використовуються для прийняття важливих рішень.

1.4.1 Глибинне навчання

Глибоке навчання є частиною ширшого сімейства методів машинного навчання, яке базується на штучних нейронних мережах із навчанням представленням. Навчання представленням – це клас підходів до машинного навчання, які дозволяють системі виявляти представлення, необхідні для виявлення ознак або класифікації з вихідних даних. Вимоги до ручного проектування функцій зменшуються, дозволяючи машині вивчати функції та застосовувати їх до певної діяльності.

Навчання може бути контрольованим, напівконтрольованим або неконтрольованим.

Архітектури глибокого навчання, такі як глибокі нейронні мережі, мережі глибоких переконань, глибоке навчання з підкріпленням, рекурентні нейронні мережі, згорткові нейронні мережі та трансформатори, застосовувалися в таких областях, як комп’ютерне бачення, розпізнавання мови, обробка природної мови, машинний переклад, біоінформатика, дизайн ліків , аналіз медичних зображень, кліматологія, інспекція матеріалів і програми настільних ігор, де вони дали результати, які можна порівняти з результатами експертів, а в деяких випадках навіть перевершити їх [33].

Штучні нейронні мережі (ШНМ) мають різні відмінності від біологічного мозку. Зокрема, штучні нейронні мережі мають тенденцію бути статичними та символічними, тоді як біологічний мозок більшості живих організмів є динамічним (пластичним) та аналоговим.

Прикметник «глибокий» у глибокому навчанні відноситься до використання кількох рівнів у мережі. Рання робота показала, що лінійний перцептрон не може бути універсальним класифікатором, але мережа з неполіноміальною функцією активації з одним прихованим шаром необмеженої ширини може. Глибоке навчання — це сучасна варіація, яка пов’язана з необмеженою кількістю шарів обмеженого розміру, що дозволяє практичне застосування та оптимізоване впровадження, зберігаючи теоретичну універсальність у м’яких умовах. У глибокому навчанні також дозволено, щоб рівні були неоднорідними та суттєво відхилялися від біологічно обґрунтованих коннекціоністських моделей заради ефективності, можливості навчання та розуміння [34].

1.4.2 Комп’ютерне бачення

Завдання комп’ютерного бачення включають методи отримання, обробки, аналізу та розуміння цифрових зображень, а також вилучення даних великої розмірності з реального світу для отримання числової або символьної інформації, наприклад у формах рішень. Розуміння в цьому контексті означає перетворення візуальних образів (введення сітківки ока) в описи світу, які мають сенс для мисленнєвих процесів і можуть викликати відповідні дії. Таке розуміння зображення можна розглядати як відокремлення символічної інформації від даних зображення за допомогою моделей, побудованих за допомогою геометрії, фізики, статистики та теорії навчання.

Наукова дисципліна комп’ютерного зору займається теорією створення штучних систем, які витягують інформацію із зображень. Дані зображення можуть приймати різні форми, наприклад відеоряди, зображення з кількох камер, багатовимірні дані з 3D-сканера або медичних скануючих пристроїв. Технологічна дисципліна комп’ютерного бачення прагне застосувати свої теорії та моделі до побудови систем комп’ютерного бачення.

Підобласті комп’ютерного бачення включають реконструкцію сцени, виявлення об’єктів, виявлення подій, відстеження відео, розпізнавання об’єктів, оцінку 3D пози, навчання, індексування, оцінку руху, візуальне обслуговування, моделювання 3D сцени та відновлення зображення. Розгляд задач та методів їх вирішення у комп’ютерному баченні подано в Додатку А.

Запровадження технології комп’ютерного бачення може бути кропіткою справою для організацій, оскільки для неї не існує єдиного рішення. Дуже небагато компаній пропонують уніфіковану та розподілену платформу чи операційну систему, де програми комп’ютерного зору можна легко розгортати та керувати ними.

1.5 Висновок до першого розділу

В першому розділі кваліфікаційної роботи освітнього рівня «Магістр» проведено аналіз предметної області по потоковій передачі зображень, її використанню, викликів, з якими може зіткнутися розробник під час роботи з нею, а також задач, в яких вона використовується. Також розглянуто використання хмарної архітектури для вирішення проблем в обробці зображень, а саме використання ресурсів і обробка навантаження для вирішення затримки та забезпечення якості. Були проаналізовані джерела генерації потоку та їх можливості у передачі даних. Було розглянута інженерія даних, яка допоможе якісно та ефективно обробляти великі обсяги даних у режимі реального часу, і машинний аналіз, зокрема глибинне навчання і комп’ютерне бачення, які виконують вирішальну роль в обробці зображень машинами.

2 АНАЛІЗ СИСТЕМ ОБРОБКИ ПОТОКУ ДАНИХ ТА МОДЕЛЕЙ МАШИННОГО АНАЛІЗУ ЗОБРАЖЕНЬ

2.1 Опис продукту та вимог

Мета розробки продукту полягає у створенні рішення, яке наддасть можливість генерувати команди та виконувати їх за допомогою жестів рук з використанням будь-якої доступної камери.

Важливою вимогою є те, що камери можуть бути в різних місцях одночасно і також користувачів (людей на камері) може бути декілька, і для кожного з них буде виконуватися персоналізована команда.

Для децентралізації і масштабування сервісу потрібне хмарне рішення, яке наддасть змогу обробляти відеопотік з різних куточків світу.

Для перетворення відеоряду в набір корисної інформації потрібно використати моделі машинного навчання, які надають змогу знаходити людей, розпізнавати лиця та жести рук, тощо.

Для інтерактивної взаємодії користувача з сервісом необхідно розробити веб-клієнт, який дозволить користувачу переглянути свої команди, додати команди або навіть прив’язати елементи розумного будинку та використовувати розумного асистента для виконання команд.

2.2 Аналіз хмарних платформ

Хмарна платформа виступає як найкраще рішення для швидкої, ефективної та дешевої інтеграції власного сервісу, додатку або навіть комплексної системи взаємопов’язаних між собою сервісів. Хмара надає можливість розгорнути обробку потоку даних та одномоментно опрацьовувати дані з великої кількості різних джерел, динамічно масштабуючись горизонтально в залежності від навантаження. Це забезпечить стабільну роботу

сервісу, а також зекономить велику кількість коштів на відміну від звичайних серверів, де усі ресурси, мережу, безпеку, мікросервісну взаємодію (при наявності) потрібно налаштовувати власноруч і оплачувати те, що орендовано, а не за те, що використовується у даний момент часу [35].

2.2.1 Amazon Web Services

Одним із найпопулярніших хмарних рішень є Amazon Web Services.

Amazon Web Services є дочірньою компанією Amazon.com, яка надає платформу хмарних обчислень в оренду окремим особам, компаніям і урядам на основі передплати. Технологія дозволяє абонентам мати в своєму розпорядженні повноцінний віртуальний кластер комп'ютерів, який завжди доступний через Інтернет. Віртуальні машини AWS мають більшість атрибутів справжнього комп’ютера, включаючи апаратну периферію; операційна система; мережа; і попередньо встановлені прикладні програми, такі як веб-сервер, база даних, CRM тощо. Кожна система AWS також віртуалізує консольний ввід-вивід (клавіатура, монітор і миша), що дозволяє користувачам AWS підключатися до системи AWS за допомогою браузера. Браузер діє як вікно на віртуальній машині, дозволяючи користувачеві підключатися, налаштовувати та використовувати свої віртуальні системи, як справжній фізичний комп’ютер. Це дозволяє їм налаштувати систему для надання Інтернет-орієнтованих послуг і послуг своїм клієнтам [36].

Технологія AWS заснована на кластерах серверів (ферм), розташованих по всьому світу. Плата за користування базується на комбінації функцій апаратного забезпечення / операційної системи / програмного забезпечення / мережі, вибраних користувачем, а також вимог щодо доступності, резервування, безпеки та додаткових опцій. Залежно від того, що потрібно і за що платить користувач, він може зарезервувати віртуальну машину, кластер віртуальних машин, фізичний (реальний) комп’ютер (сервер)

для свого виключного використання або навіть кластер фізичних комп'ютери (кластер серверів). Amazon прагне підтримувати й оновлювати своє програмне й апаратне забезпечення відповідно до необхідних стандартів безпеки. AWS працює в багатьох географічних регіонах, включаючи Канаду, Німеччину, Ірландію, Сінгапур, Токіо, Сідней, Пекін, Лондон та інші.

AWS надає понад 70 послуг, що охоплюють широкий спектр, включаючи обчислення та зберігання даних, передачу даних, аналітику, мобільні програми, інструменти розробника тощо. Найпопулярнішими з них є Amazon Elastic Compute Cloud (EC2) і Amazon Simple Storage (S3). Більшість послуг не надаються безпосередньо кінцевим користувачам, а натомість пропонують функціональні можливості через API, які розробники можуть використовувати у своїх програмах. Пропозиції Amazon Web Services доступні через HTTP з використанням архітектурного стилю REST і протоколу SOAP.

Amazon рекламує AWS як спосіб отримати обчислювальну потужність, яка масштабується швидше та дешевше, ніж створення власного кластера фізичних серверів. Усі послуги виставляються на основі використання, але кожна послуга вимірює використання по-своєму.

2.2.2 Google Cloud Platform

Google Cloud Platform (GCP), запропонована Google, – це набір хмарних обчислювальних служб, які працюють на тій самій інфраструктурі, яку Google використовує внутрішньо для своїх продуктів для кінцевих користувачів, таких як Пошук Google, Gmail, Диск Google і YouTube. Окрім набору інструментів керування, він надає низку модульних хмарних сервісів, включаючи обчислення, зберігання, аналітику даних і машинне навчання. Для реєстрації необхідні реквізити кредитної картки або банку.

Google Cloud Platform надає інфраструктуру як послугу, платформу як послугу та безсерверні обчислювальні середовища.

У квітні 2008 року Google анонсувала App Engine, платформу для розробки та розміщення веб-додатків у керованих Google центрах обробки даних, яка була першою службою хмарних обчислень компанії. Сервіс став загальнодоступним у листопаді 2011 року. Після анонсу App Engine Google додав кілька хмарних сервісів на платформу [37].

Google Cloud Platform є частиною Google Cloud, яка включає загальнодоступну хмарну інфраструктуру Google Cloud Platform, а також Google Workspace (G Suite), корпоративні версії Android і ChromeOS та інтерфейси прикладного програмування (API) для машинного навчання. і картографічні послуги компанії

2.2.3 Порівняння та вибір платформи

Зараз найбільшими претендентами на хмарні обчислення є AWS, GCP, IBM, Alibaba та MS Azure. Amazon Web Services (AWS), був в цій грі з самого початку, а Google Cloud Platform (GCP), порівняно новий гравець, почав дуже швидко зростати.

Як правило у всіх хмарних рішеннях велика кількість сервісів подібна по призначенню, приклад наведено у таблиці 2.1.

Таблиця 2.1 – Відповідність сервісів між різними хмарними рішеннями

Google Cloud Platform	**Amazon Web Services**	**Microsoft Azure**	**Oracle Cloud**
1	**2**	**3**	**4**
Compute Engine	EC2	Virtual Machines	Cloud Infra OCI
App Engine	Elastic Beanstalk	App Services	Application Container
Kubernetes Engine	Elastic Kubernetes Service	Kubernetes Service	Kubernetes Service

Продовження таблиці 2.1

1	2	3	4
Cloud Bigrable	DynamoDB	Cosmos DB	NoSQL Database
BigQuery	Redshift	Synapse Analytics	Autonomous Data Warehouse
Cloud Functions	AWS Lambda	Azure Functions	Cloud Fn
Cloud Datastore	DynamoDB	Cosmos DB	NoSQL Database
Cloud Storage	S3	Blob Storage	Storage OCI

Що ж, обидва хмарні провайдери мають свої плюси та мінуси, але якщо підсумувати, то можна сказати, що AWS був на ринку надто довго та чудово працює з максимальною часткою ринку. Що стосується послуг, то AWS є безперечним переможцем, оскільки кількість послуг, які пропонує AWS, значно перевищує кількість послуг, які пропонує GCP. Послуги, доступні на AWS, надзвичайно широкі. Ці різноманітні служби дійсно добре інтегровані, і вони надають дуже комплексну хмарну службу.

Розглядаючи наявні сервіси для обробки потокових даних, AWS надає досить великий вибір сервісів які протягом часу чудово зарекомендували себе на ринку, а також AWS надає доступ до сервісу Amazon Kinesis Video Streams та має велику кількість інтегрованих між собою сервісів таких як: Amazon Rekognition Video, SageMaker, MxNet/Tensorflow, Custom Media Processing.

Підсумовуючи результати, Google Cloud є чудовим конкурентом, який дуже швидко розвивається, проте саме для задачі аналізу потокових відеоданих більше підходить AWS [38].

2.3 Системи передачі та опрацювання потокових даних

Основними конкурентами в обробці потокових даних є рішення з відкритим кодом Apache Kafka і сервісна група AWS Kinesis Family.

Apache Kafka – це розподілена платформа для зберігання подій і потокової передачі даних. Це система з відкритим кодом, розроблена Apache Software Foundation, написана на Java та Scala. Проект має на меті забезпечити уніфіковану, високопродуктивну платформу з низькою затримкою для обробки потоків даних у реальному часі. Kafka може підключатися до зовнішніх систем (для імпорту/експорту даних) через Kafka Connect і надає бібліотеки Kafka Streams для програм обробки потоків. Kafka використовує бінарний протокол на основі TCP, оптимізований для підвищення ефективності, і спирається на абстракцію «набору повідомлень», яка природним чином групує повідомлення разом, щоб зменшити накладні витрати на передачу по мережі. Це «призводить до більших мережевих пакетів, більших послідовних дискових операцій, безперервних блоків пам’яті, що дозволяє Kafka перетворювати пакет записів випадкових повідомлень у лінійні записи».

Apache Kafka базується на журналі фіксації, що дозволяє користувачам підписуватися на нього та публікувати дані в будь-якій кількості систем або програм у режимі реального часу. Приклади додатків включають керування набором пасажирів і водіїв в Uber, забезпечення аналітики в реальному часі та прогнозне технічне обслуговування розумного будинку British Gas, а також виконання багатьох зад у режимі реального часу на LinkedIn [39, 40].

Основне завдання сервісу є передача подій певних виробників даних різній кількості споживачів у реальному часі, продемонстрованих на рисунку 2.1. Ця модель може викликати багато проблем, таких як:

- Надійність і гарантія доставки даних.
- Підключення нових споживачів
- Виробники повинні знати, якому споживачу надсилати дані.
- Підтримка.
- Інтеграція різних стеків.

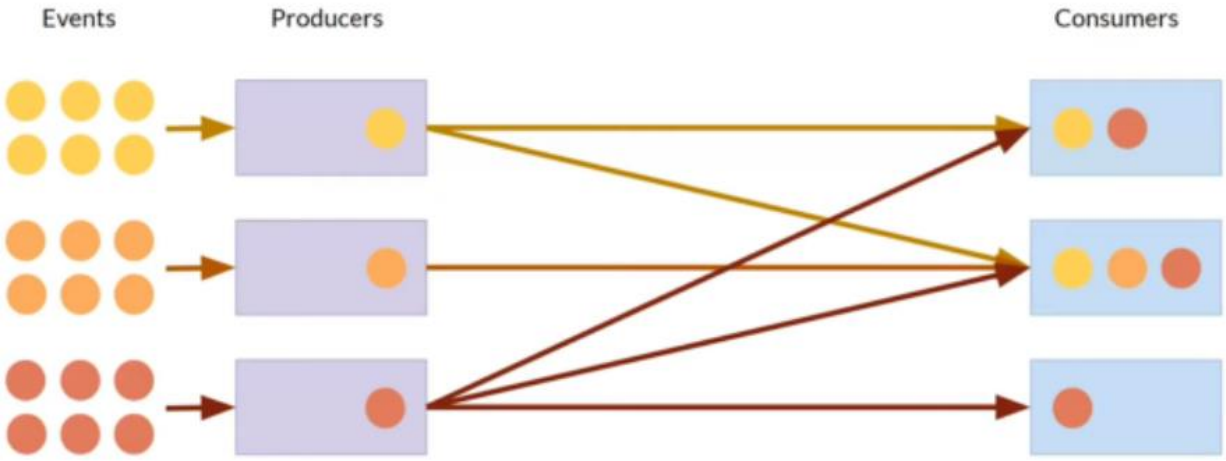


Рисунок 2.1 – Проблематика поставленої задачі

Для вирішення ряду проблем, описаних вище, існує таке рішення, як брокер. Брокер діє як сховище даних в окремих контейнерах, з яких споживач збирає дані, і в нашому випадку брокером є Кафка, приклад якого зображений на рисунку 2.2.

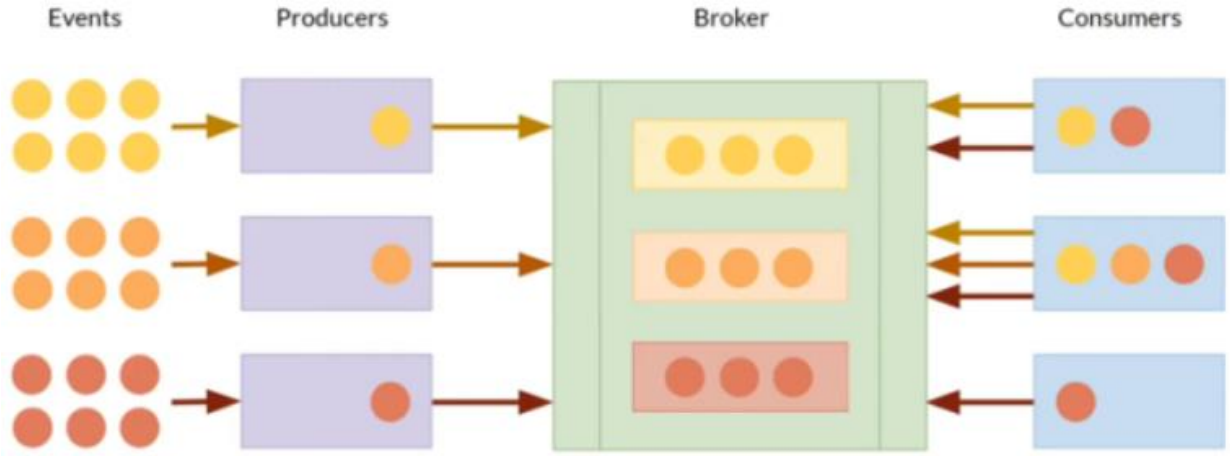


Рисунок 2.2 – Схема з посередникм (Kafka)

Основні сутності в сервісі Kafka:

– Broker.
– Zookeeper.
– Message (Record).
– Topic / Partition.

– Producer.

– Consumer.

Посередник Kafka виконує роль одержання повідомлень, збереження повідомлень і видачі повідомлень. Якби був лише один брокер, виникла б проблема надійності та продуктивності даних, оскільки з’явилися нові брокери, які розмовляють і утворюють кластер kafka, який забезпечує реплікацію та масштабованість.

Далі потрібен координатор, який зберігає стан і конфігурацію кластера, для якого використовується Zookeeper [47]. Zookeeper – це свого роду база даних, яка швидко працює на операції читання та повільно на запис, тому вона добре підходить для зберігання метаданих, таких як адреси брокерів, розділи, теми, ці дані активно використовуються під час читання та написання повідомлень у Kafka, що можна побачити на рисунку 2.3.

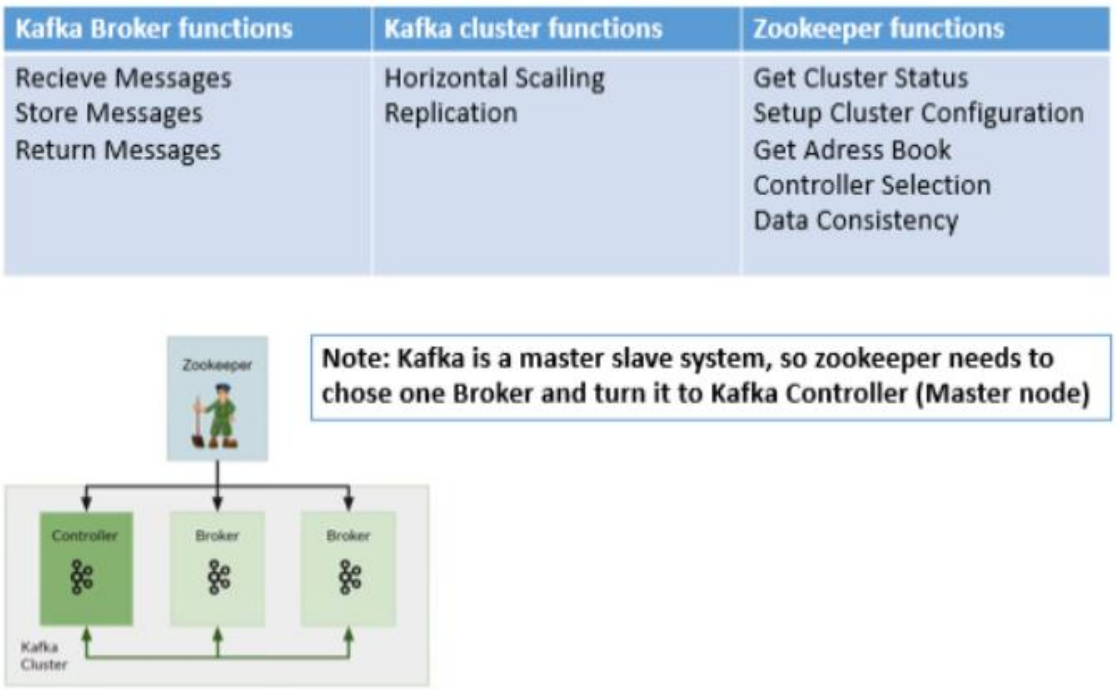


Рисунок 2.3 – Архітектура Kafka кластеру

Повідомлення в Apache Kafka – це пара ключ-значення, а також є позначка часу (timestamp) та заголовки (headers), структуру повідомлення вказано у таблиці 2.2.

Таблиця 2.2 – Структура повідомлення Apache Kafka

Поле повідомлення	**Опис**
Ключ (Key)	Використовується для розподілу повідомлень в кластері
Значення (Value)	Повідомлення в байтах
Часова мітка (Timestamp)	Час отримання повідомлення в кластері, Epoch формат
Заголовки (Headers)	Додаткові атрибути у форматі ключ значення.

Далі розглянемо теми (topic), тема – це просто потік подібних або згрупованих подій, у тому числі повідомлення теми можуть бути записані з різних джерел до черги, а потім із цієї черги. Очікуючи, споживач читає ці дані в тому самому порядку надходження цих даних [48]. Це називається FIFO (першим увійшов, першим вийшов). і дані не видаляються, якщо буде новий користувач, він зможе прочитати всі повідомлення з теми. У темі може мати кілька сегментів, їх кількість можна регулювати. Дані можуть надсилатися кількома виробниками та отримуватись із різних сегментів у кількох потоках відповідно. Як читати і писати з цих сегментів, буде розглянуто далі. FIFO також зберігається, але тільки для сегмента [41]. Приклад цього можна побачити на рисунку 2.4.

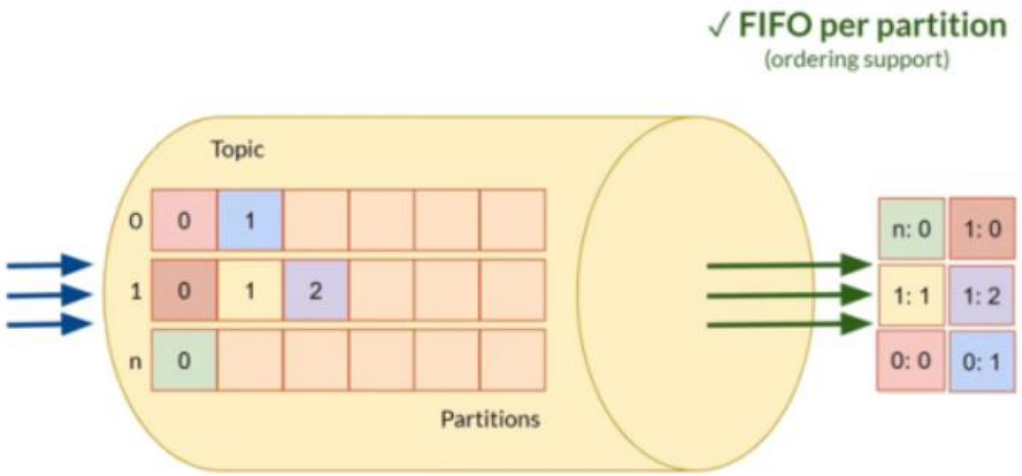


Рисунок 2.4 – Топік та партиціонування

Якщо розділи без реплік, дані будуть втрачені, якщо брокер перестає відповідати. При створенні теми є параметр replication-factor. У Kafka є таке поняття як головна репліка лідера, вона визначається головним вузлом, всі дані для конкретного розділу будуть вироблятися і споживатися тільки з розділу лідера. Також потрібно стежити, щоб лідери розділів однієї теми не були на одному брокері і тоді буде велике навантаження на брокера, якщо тема об'ємна [42]. Приклад продемонстрований на рисунку 2.5.

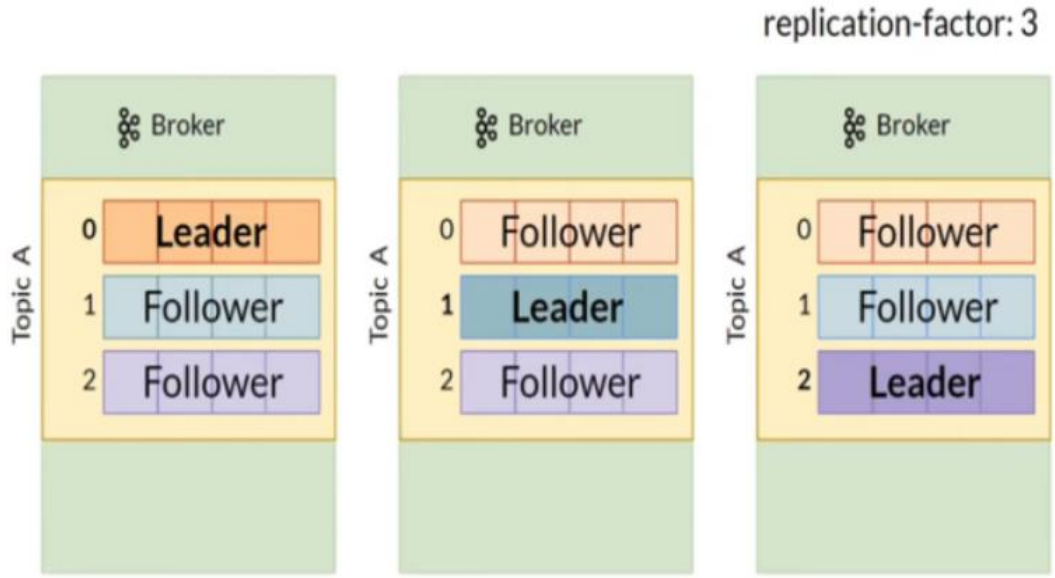


Рисунок 2.5 – Реплікація партицій між брокерами.

Також може існувати така річ, як відставання даних у підписників відносно лідера. Це пояснюється тим, що підписники періодично запитують у лідера дані. Якщо лідер падає, а підписників не встигають отримати дані, то дані будуть повністю втрачені. Щоб запобігти цьому, Kafka має параметр ISR, у якому дані ISR записуються синхронно з розділом лідером. Кількість усіх реплік ISR не повинна перевищувати кількість підписників, тому що якщо реплік ISR недостатньо, то виробник не зможе надіслати повідомлення, відповідно, якщо брокер впаде, і ми можемо не мати достатньо реплік ISR і тема просто перестає отримувати, відправляти повідомлення [43, 44]. Приклад запису продемонстрований на рисунку 2.6.

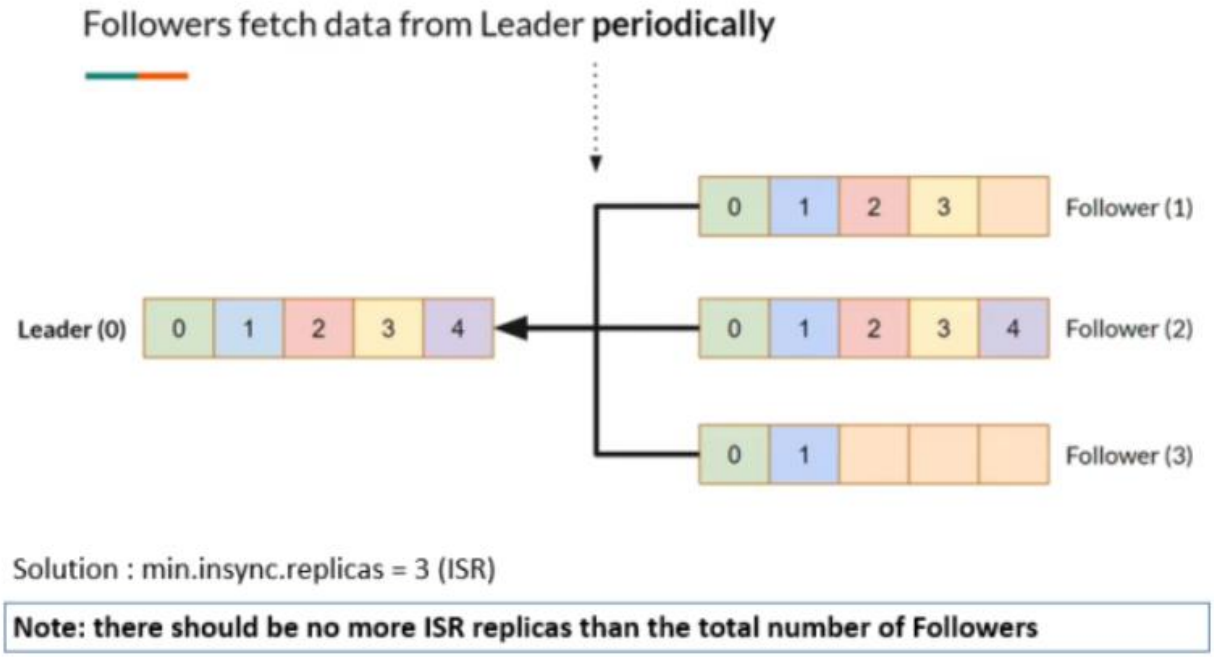


Рисунок 2.6 – Синхронний запис ISR.

Kafka також пропонує таке рішення, як група споживачів, де кожен споживач обробляє дані з кожного розділу паралельно. Також потрібно пам'ятати, що не потрібно робити більше споживачів, ніж сегментів. У групах споживачів існує поняття зміщення, якщо перший споживач обробив ряд повідомлень у розділі 0, а потім припинив роботу. Другий споживач починає читати цю частину і повинен отримати інформацію про те, які зміщення вже

оброблено, у такому випадку є таблиця взаємозаліків споживачів [45, 46]. Приклад зображений на рисунку 2.7.

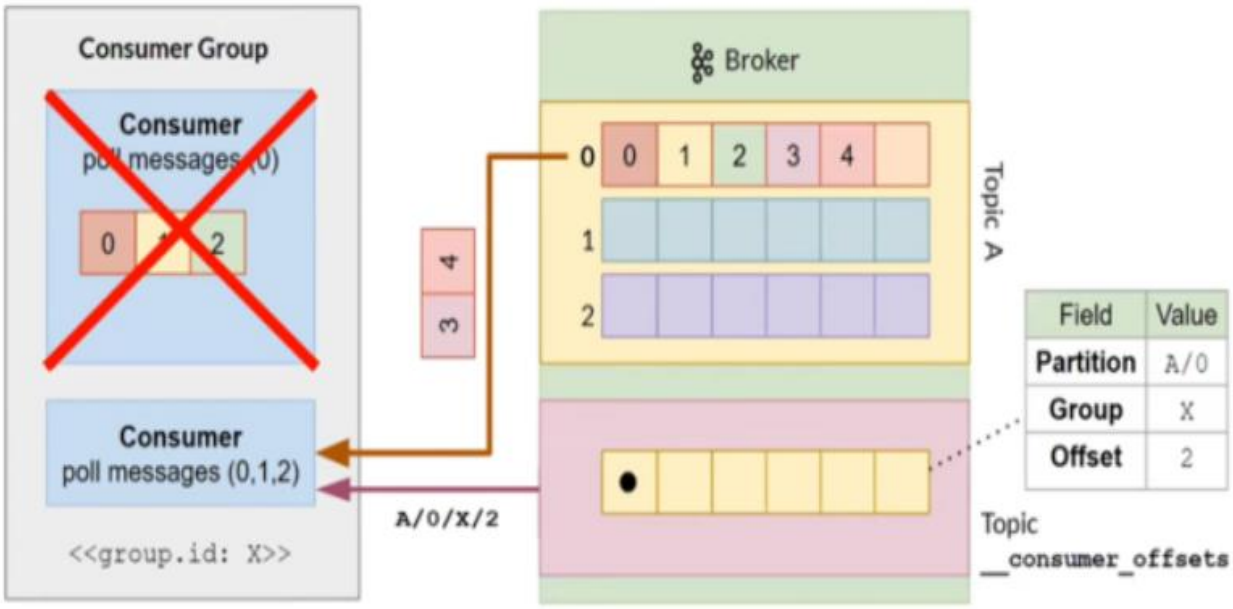


Рисунок 2.7 – Групи споживачів та офсети

Існує два типи фіксації:

– Автоматична фіксація – це коли споживач тільки почав працювати над зсувом або зміщеннями і одразу підтвердив, що він над ним працював. У цьому випадку ми можемо пропустити повідомлення, якщо споживач перестане працювати до завершення обробки.

– Ручна фіксація фіксується після завершення обробки зсуву або кількості зсувів. У цьому випадку ми можемо отримати дубльовані повідомлення, наприклад, якщо споживач починає обробку 5 зсувів і опускається на третьому, то перші два повідомлення будуть продубльовані.

Amazon Managed Streaming for Apache Kafka (Amazon MSK) – сервіс який пропонує Apache Kafka для обробки потокових даних. Amazon MSK забезпечує такі операції керування, як створення, оновлення та публікація кластерів. Це дозволяє використовувати операції в області даних Apache Kafka і зберігати дані. Він працює з відкритими версіями Apache Kafka. Це означає, що існуючі

програми, інструменти та плагіни від партнерів Apache Kafka та спільноти підтримуються без необхідності змінювати код програми. Ви можете використовувати Amazon MSK для створення кластерів, які вибирають будь-яку версію Apache Kafka, перелічену в розділі «Підтримувані версії Apache Kafka». На рисунку 2.8 показано як працює Amazon MSK [49].

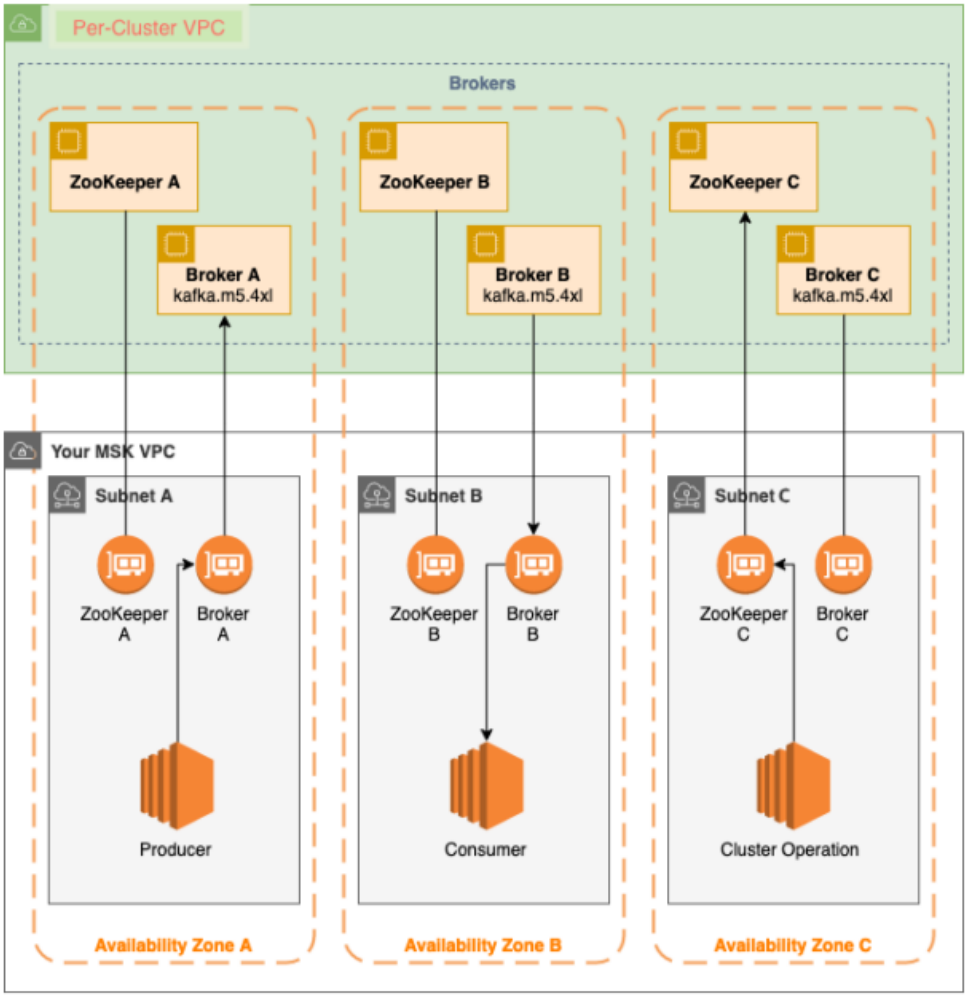


Рисунок 2.8 – Діаграма роботи Amazon MSK

На діаграмі показано взаємодію між такими компонентами:

– Вузли-посередники – під час створення кластера Amazon MSK вказується, скільки вузлів-посередників Amazon MSK має створити в кожній зоні доступності. У прикладі кластера, показаному на цій діаграмі, є один брокер на зону доступності. Кожна зона доступності має власну підмережу віртуальної приватної хмари (VPC).

– Вузли ZooKeeper – Amazon MSK також створює вузли Apache ZooKeeper. Apache ZooKeeper — це сервер із відкритим кодом, який забезпечує високонадійну розподілену координацію.

– Виробники, споживачі та розробники топіків – Amazon MSK дозволяє використовувати операції площини даних Apache Kafka для створення тем, а також для виробництва та споживання даних.

– Операції кластера. Для виконання операцій на рівні керування можна використовувати консоль керування AWS, інтерфейс командного рядка AWS (AWS CLI) або API SDK. Наприклад, можна створити або видалити кластер Amazon MSK, створити список усіх кластерів у своєму обліковому записі, переглянути властивості кластера та оновити кількість і тип агентів у кластері.

Amazon MSK виявляє та автоматично відновлює найпоширеніші сценарії збою для кластерів, щоб ваші додатки-виробники та споживачі могли продовжувати свої операції запису та читання з мінімальним впливом. Коли Amazon MSK виявляє збій брокера, він усуває збій або замінює несправного чи недоступного брокера новим. Крім того, там, де це можливо, він повторно використовує сховище від старішого брокера, щоб зменшити дані, які Apache Kafka потребує для копіювання. Вплив користувача на доступність обмежується часом, необхідним Amazon MSK для завершення виявлення та відновлення. Після відновлення програми виробника та споживача можуть продовжувати спілкуватися з тими самими IP-адресами брокера, які вони використовували до збою.

Наступним сервісом потокової передачі даних стала група сервісів AWS Kinesis Family, додаковий огляд групи сервісів подано в Додатку А .

Amazon Kinesis Data Streams – це безсерверна служба потокового передавання даних, яка дозволяє легко отримувати, обробляти та зберігати потоки даних у будь-якому масштабі. Схема його роботи продемонстрована на рисунку 2.9.

Рисунок 2.9 – Взаємодія з Kinesis Data Streams

Amazon Kinesis Data Firehose – це служба вилучення, трансформації та завантаження (ETL), яка надійно збирає, перетворює та передає дані в озера даних, сховища даних і аналітичні служби. Схема його роботи продемонстрована на рисунку 2.10.

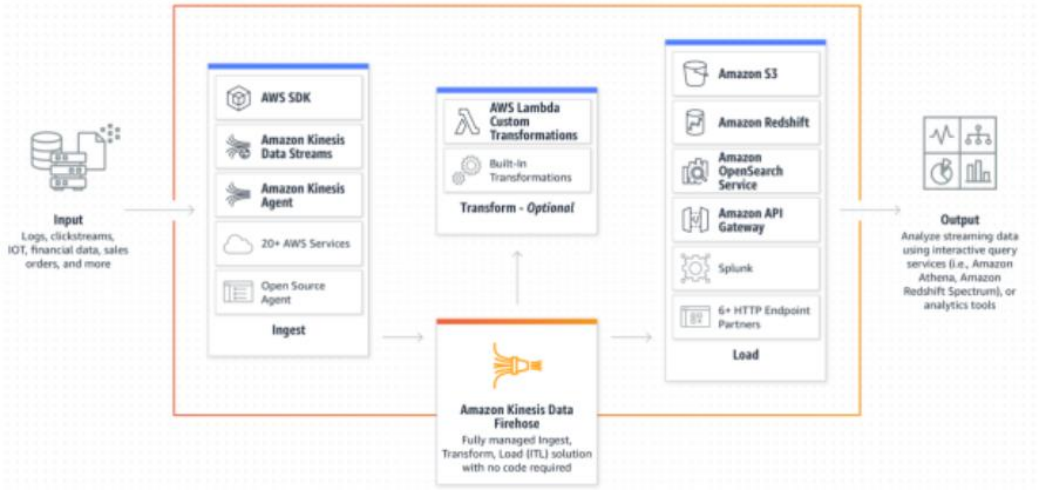


Рисунок 2.10 – Взаємодія з Kinesis Data Firehose

Amazon Kinesis Video Streams дозволяє легко та безпечно передавати відео з підключених пристроїв на AWS для аналізу, машинного навчання (ML), відтворення та іншої обробки. Kinesis Video Streams автоматично налаштовує та еластично масштабує всю інфраструктуру, необхідну для отримання потокових відеоданих з мільйонів пристроїв. Він безпечно зберігає, шифрує та індексує відеодані з ваших потоків і дає вам доступ до даних через прості у

використанні API. Kinesis Video Streams дає змогу відтворювати відео для перегляду в реальному часі та на вимогу, а також швидко створювати програми, які використовують переваги комп'ютерного зору та відеоаналітики, завдяки інтеграції з Amazon Rekognition Video та бібліотеками для фреймворків ML, таких як Apache MxNet, TensorFlow і OpenCV. Kinesis Video Streams також підтримує WebRTC, проект з відкритим вихідним кодом, який забезпечує потокове передавання медіа в реальному часі та взаємодію між веб-браузерами, мобільними програмами та підключеними пристроями через прості API. Типове використання включає відеочат і однорангову потокову передачу медіа, що розглянуто на рисунку 2.11.

Рисунок 2.11 – Взаємодія з Kinesis Video Streams

Провівши глибокий аналіз рішень, було прийнято рішення використовувати технології групи сервісів AWS Kinesis, що надасть змогу ефективно масштабувати ресурси без додаткового втручання в залежності від кількості джерел потокових даних, а також використовувати AWS Cloud як комплексне рішення, поєднуючи декілька сервісів між собою, і створюючи гнучку систему опрацювання потокових даних [50, 51].

2.4 Методи та моделі машинного аналізу зображень

2.4.1 Моделі розпізнавання об’єктів

Протягом останнього десятиліття, головною технологією в галузі комп'ютерного бачення є згорткові нейронні мережі (CNN). Після успішного вирішення завдань класифікації зображень за допомогою мережі AlexNet, з'явилось багато інших мережевих архітектур на основі CNN.

Класифікація зображень стосується опису зображень, тоді як задача по розпізнаванню об'єктів спрямована на визначення розташування цільових об'єктів у межах зображення. Завдання виявлення складається з двох підзавдань: перше – отримання інформації про клас та ймовірність належності об’єкта до нього, тобто класифікація; друге – фактичне знаходження та розташування цілей з використанням обмежувальних рамок з мітками, що є завданням позиціонування. В рамках першого виконується пошук усіх цільових об’єктів з визначенням впевненості класифікації, в рамках другого – чітке визначення меж знайдених об’єктів.

Головні сучасні методи виявлення об'єктів можна розділити на два типи: одноетапний підхід (наприклад, SSD, YOLO) та двоетапний підхід (наприклад, серія R-CNN). Двоетапний підхід спочатку генерує розріджений набір обмежувальних рамок на зображенні, а потім використовує його для поліпшення кінцевих результатів виявлення. У порівнянні з цим, одноетапний підхід безпосередньо розраховує зображення та генерує результати виявлення. Одноетапний підхід є швидшим, але менш точним, ніж двоетапний підхід [15]. Оскільки, в даній розробці нас більше цікавить швидкість, для можливості запуску моделі в режимі потокового перегляду – було оглянуто моделі, які виконують одноетапний підхід до розпізнання – моделі сімейства YOLO та SSD.

Моделі сімейства YOLO використовують CNN для виявлення всіх об’єктів в межах зображення одночасно. YOLO замість того, щоб застосовувати кілька проходів зображення для кожного цільового об'єкта окремо, розбиває зображення на сітку та використовує до нього CNN лише один раз.

Схематичне зображення YOLO архітектури зображене на рисунку 2.12.

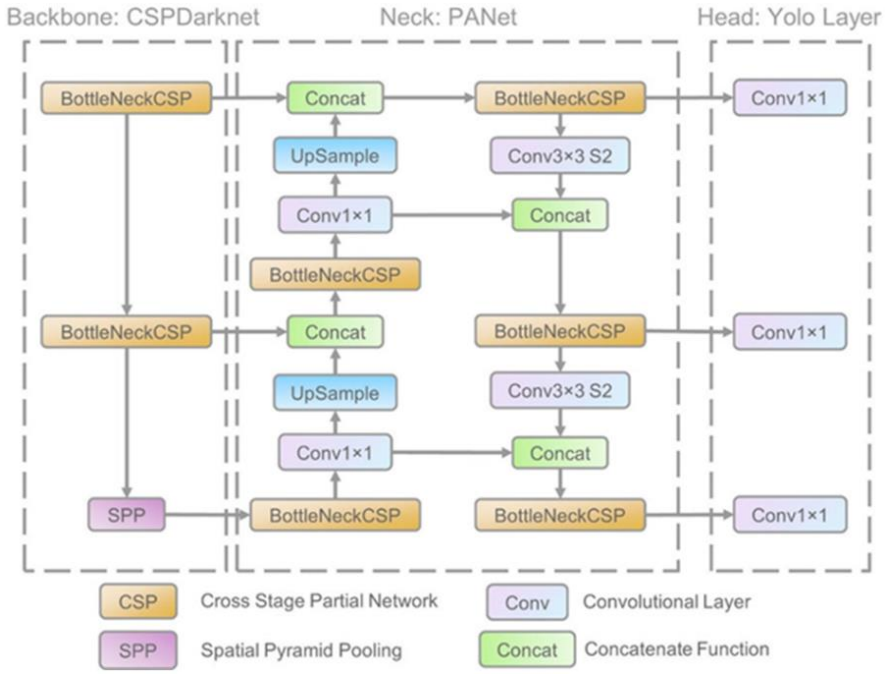


Рисунок 2.12 – Схематичне зображення YOLO архітектури

Із застосуванням цього підходу, YOLO досягає високої точності та швидкодії, здатної виявляти об'єкти в режимі реального часу на прямих відеопотоках зі швидкістю приблизно 65 кадрів в секунду. Крім того, він використовує метод Intersection over Union (IoU) для визначення кількох об'єктів одного класу на одному зображенні.

Алгоритм YOLO складається з трьох етапів: залишкового блоку (сітки, residual block), регресії обмежувальної рамки та використання методології Intersection over Union (IoU). Спочатку зображення розбивається на сітку, яку також називають залишковим блоком. Замість того, щоб використовувати кілька проходів зображення для кожного наявного об’єкта, YOLO проганяє CNN для всього зображення за один раз, перевіряючи кожну клітинку на

наявність центру обмежувальної рамки. Якщо клітинка містить центральну точку обмежувальної рамки, то модель виявляє об’єкт.

Якщо клітинка містить центри декількох класів об’єктів, модель створює інтегровану матрицю результатів. Регресія обмежувальної рамки застосовується, якщо є обмежувальні рамки, що перекриваються. Модель перевіряє, чи належать класи об’єктів обмежувальним рамкам одному або різним класам. Використання методології IoU дозволяє визначити, наскільки перекриваються обмежувальні рамки для одного класу об’єктів. Якщо перекриття перевищує 50%, додаткові обмежувальні рамки усуваються. Модель вибирає центральну точку обмежувальної рамки з середнім балом, застосовуючи немаксимальне придушення (non-max suppresion), щоб уникнути дублювання об’єктів [2, 3, 4, 15].

Приклад процесу зображений на рисунку 2.13.

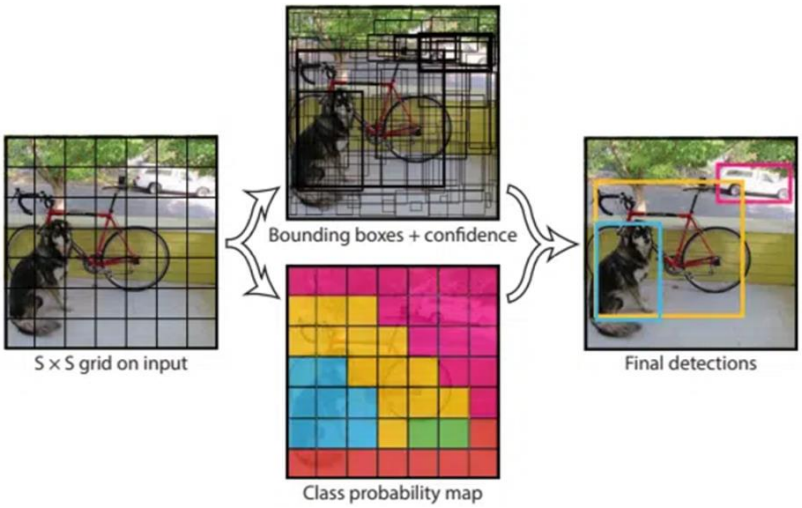


Рисунок 2.13 – Процес роботи моделі

SSD – це модель глибокого навчання для виявлення та локалізації об’єктів. Як і YOLO, він використовує один прохід вперед для розпізнавання об’єктів з усього зображення.

SSD складається з двох ключових компонентів: багатомасштабних карт ознак для виявлення об’єктів та згорткового предиктора. Багатомасштабний

екстрактор ознак є попередньо навченою моделлю для класифікації зображень, тоді як згортковий предиктор складається з набору шарів, які обробляють вхідні дані з багатомасштабних карт ознак для виявлення об’єктів.

SSD використовує сітку для розбиття зображення на клітинки, які окремо розпізнають об'єкти всередині себе. Якщо об'єкт присутній в клітинці, вихід дорівнює одиниці, в іншому випадку – нулю. Одна з ключових відмінностей між SSD та YOLO полягає у роботі з кількома обмежуючими рамками для одного об'єкту. SSD використовує пріоритети, які є заздалегідь розрахованими рамками фіксованого розміру, подібними до реальних рамок об'єктів (ground-truth), що наближаються до IoU з оцінкою більше ніж 0,5. Ці пріоритети встановлюють початкове значення для регресії обмеженої рамки [1, 2].

На рисунку 2.14 зображена схематична архітектура моделі SSD.

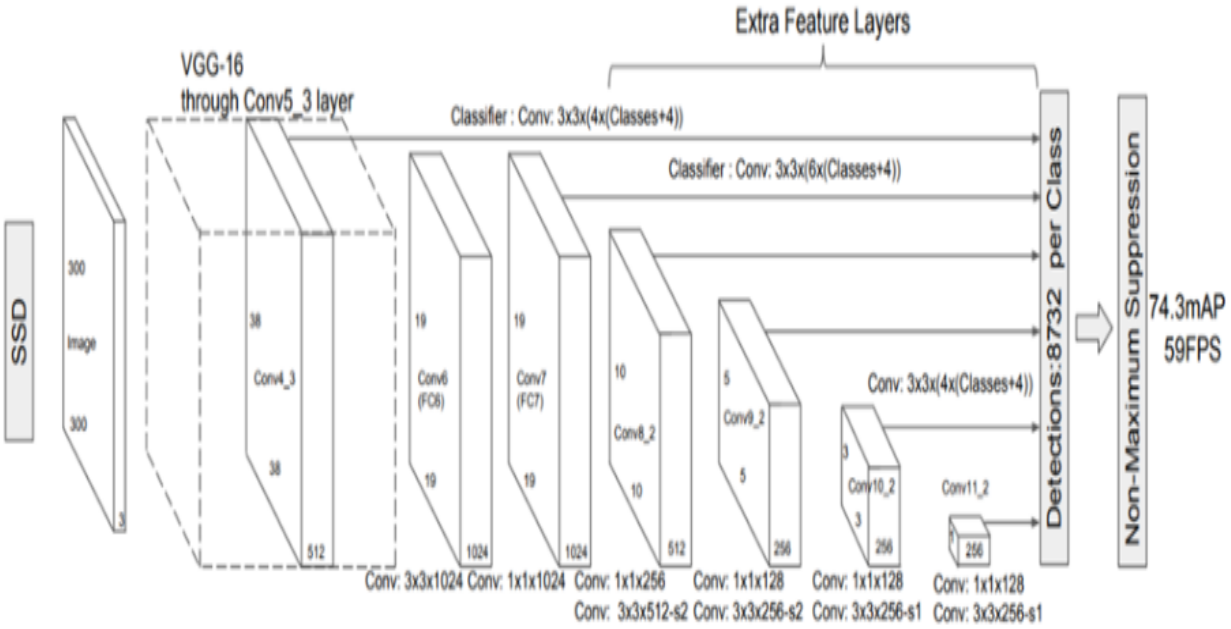


Рисунок 2.14 – Схематична архітектура моделі SSD

На рисунку 2.15 порівнюється модель YOLOv4 з іншими моделями розпізнавання об’єктів [3].

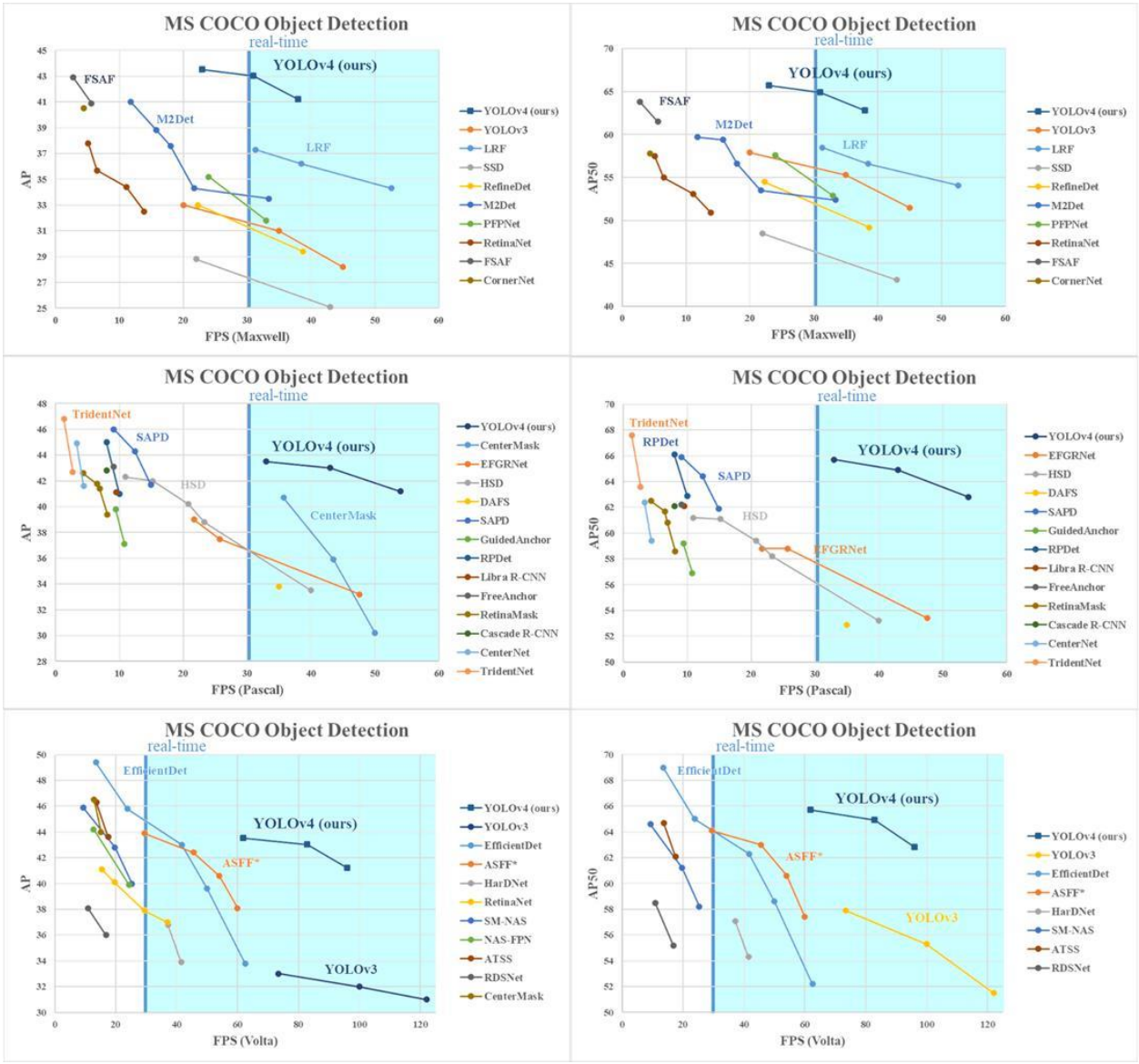


Рисунок 2.15 – Порівняння моделі сімейства YOLO з іншими моделями

Оскільки модель сімейства YOLO є найкращою у швидкому визначенні об’єктів та має досить непогану точність, для подальшої розробки була вирішено використовувати її, а саме модель YOLOv8-n.

В рамках розробки продукту від моделі вимагається виявлення лише одного клас – Людини, який є включений в перелік класів претренованої моделі YOLO на COCO датасетів від Microsoft [16]. Відповідно надалі можна використати претреновану модель без потреби збирати власний датасет та проводити тренування.

2.4.2 Системи розпізнавання лиць

Після виявлення людей на зображенні наступним кроком є використання систем з участю нейронних мереж для розпізнавання лиць, щоб класифікувати клієнта.

Сучасні системи розпізнавання обличчя використовують моделі глибокого навчання на основі згорткових нейронних мереж (CNN) для вилучення ознак із зображень обличчя та відображення їх у багатовимірному просторі ознак. Ці системи досягли надзвичайного успіху в задачах розпізнавання облич, у деяких випадках перевершуючи точність людського рівня. Вони поділяються на 4 основні компоненти:

1. Виявлення лиць – перший крок систем, який полягає у пошуку усіх облич на зображенні, їх вилученні та обрізанні для подальшої обробки. Найпопулярніші на даний момент алгоритми та моделі – Каскадне розпізнавання облич Хаара (Haar cascade), гістограма орієнтованих градієнтів (HOG) Dlib, CNN-базована мережа Dlib та MTCNN мережа.

2. Вирівнювання обличчя – оскільки лиця можуть бути повернуті під різноманітними кутами, що ускладнює їх порівняння між собою в рамках 2Д простору, важливо провести вирівнювання обличчя по одному стандарту. Переважно для цього використовується пошук ключевих точок обличчя, таких як краї очей, носу та роту. Після цього виконуються трансформації зони лиця таким чином, щоб вирівняти розташування точок.

3. Вилучення ознак – найважливіший крок, який здебільшого виконує мережа глибинного навчання. Він полягає у тому, щоб знайти та вилучити певні ознаки лиця, для утворення вбудованих векторів (embedded vectors) ознак, які будуть ідентифікувати лице.

4. Пошук та порівняння векторів – отримані вектори порівнюються з відомими векторами, і у випадку отримання співпадіння між двома векторами – лице класифікується.

На рисунку 2.16 продемонстрована схема роботи даного пайплайну. [17, 18, 19]

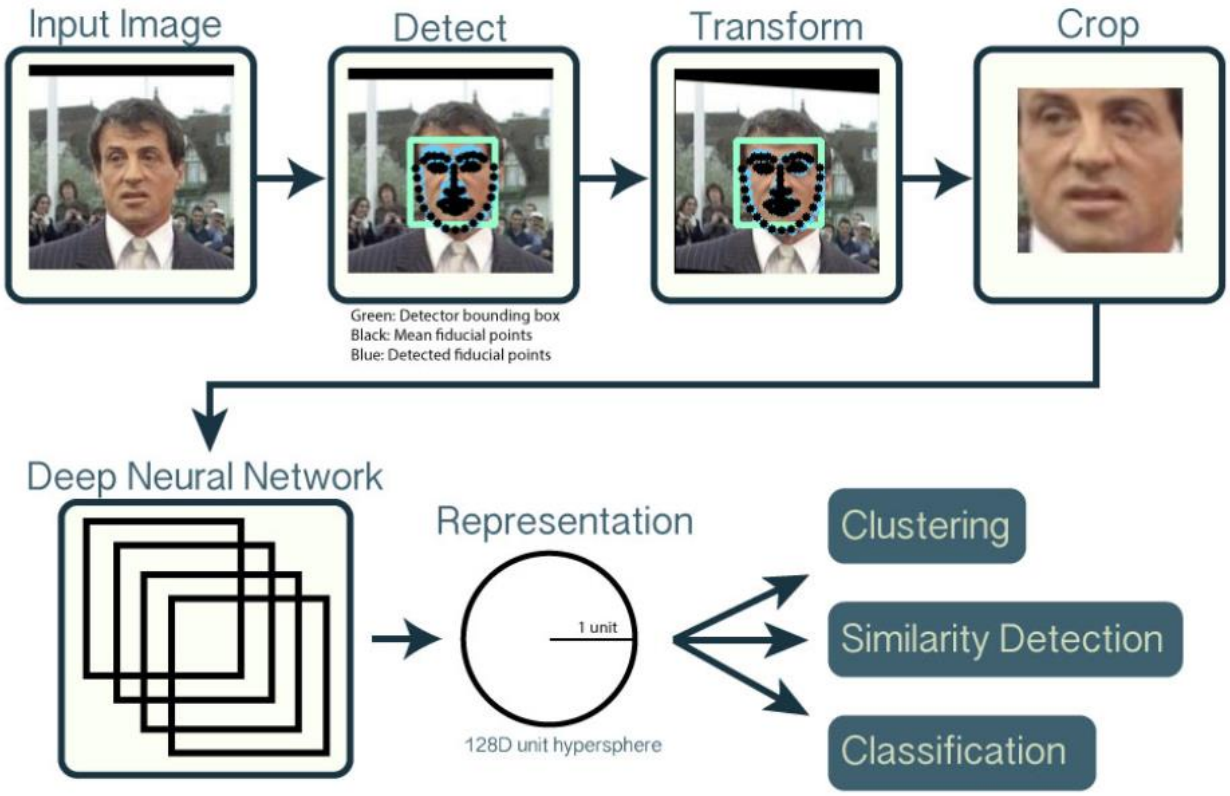


Рисунок 2.16 – Схема розпізнавання лиць

У таблиці 2.3 продемонстрований перелік сучасних моделей розпізнавання лиць з порівнянням їх точності на датасеті Labeled Faces in the Wild підготовленого Масачусетським університетом [20].

Таблиця 2.3 – Перелік моделей розпізнавання лиць з порівнянням їх точності

Модель	**Точність**
1	**2**
Facenet512	99.65%
SFace	99.60%

Продовження таблиці 2.3

1	2
ArcFace	99.41%
Dlib	99.38%
Facenet	99.20%
VGG-Face	98.78%
Люди	97.53%
OpenFace	93.80%

Оскільки точність моделей є досить високою, важливим параметром при виборі також є швидкодія розпізнавання. В результаті аналізу та експериментів була обрана модель DLib, яка має задовільну якість та швидкість.

2.4.3 Моделі оцінки пози рук

Оцінка пози руки – це завдання комп’ютерного зору, яке передбачає прогнозування положення та орієнтації руки та пальців людини на зображенні чи відео. Це завдання є важливим у багатьох програмах, таких як віртуальна та доповнена реальність, взаємодії людини з комп’ютером і розпізнавання мови жестів.

Оцінка пози руки є складною проблемою через складність руки як багатокомпонентного органа та різні пози, які вона може приймати. Традиційні методи комп’ютерного зору, такі як зіставлення шаблонів і методи на основі функцій, мають обмеження щодо точного визначення пози рук у реальних задачах. Таким чином, нещодавні досягнення в глибинному навчанні призвели до розробки моделей оцінки пози рук, які можуть вивчати складну залежність між ознаками зображення та позами рук.

Моделі оцінки пози руки зазвичай використовують згорткові нейронні мережі (CNN) для вилучення характеристик із вхідних зображень і відображення їх у тривимірному положенні суглоба або пози.

В межах роботи були розглянуті дві моделі оцінки пози рук – Adaptive Weighting Regression для 3Д оцінки пози рук та пропрієтарна модель розпізнавання рук MediaPipe Hands від Google.

Метод адаптивної вагової регресії (AWR) побудований для використовування переваги двох методів оцінки поз рук – методів на основі виявлення і на основі регресії. Перший метод дозволяє побудувати оцінку щільності знаходження руки в просторі (наприклад, у вигляді теплокарти чи векторів), після чого координати суглобів агрегуються в дискретні координати. Другий метод напряму пов’язує дані входу до параметрів 3Д поз рук чи координат суглобів.

В методі адаптивної вагової регресії координати суглобів руки оцінюються як дискретна інтеграція всіх пікселів у представленні щільності знаходження руки в просторі, що керується агрегацією скрізь адаптивні ваги. Ця операція є диференційованою, тому вона може бути вбудована в мережу для наскрізного навчання та доповнена прямим наглядом за координатами суглобів для кращого налаштування моделі. Схематичне представлення роботи моделі знаходиться на рисунку 2.17 [22].

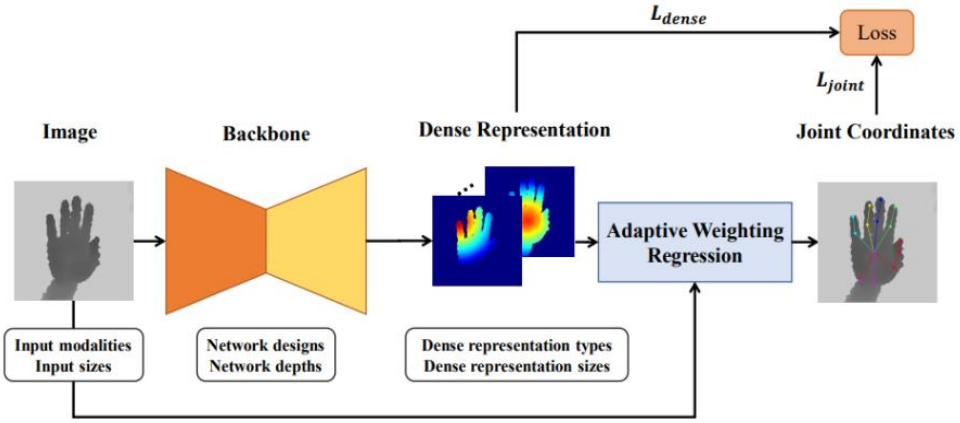


Рисунок 2.17 – Представлення роботи моделі адаптивної вагової регресії

Альтернативним методом знаходження руки є використання моделі MediaPipeHands, розробленої в Google. Дана модель насправді є ML пайплайном, що складається з двох моделей, які працюють в ансамблі [21]:

1. Детектора долоні, який працює з вхідним зображенням та визначає місцезнаходження долонь за допомогою орієнтованої обмежувальної рамки. Ця модель працює з використанням одноетапного підходу (SSD).

2. Модель знаходження орієнтирів (ключевих точок) руки, яка працює на обрізаному обмежувальному прямокутнику руки, наданому детектором долоні, і повертає орієнтири високої точності в 2,5Д вимірі. Ця модель знаходить 21 ключеві точки з їх координатами, які отримуються завдяки регресивному підході. Також модель виконує бінарну класифікацію того чи рука є правою, чи лівою.

На рисунку 2.18 показана схема роботи моделі.

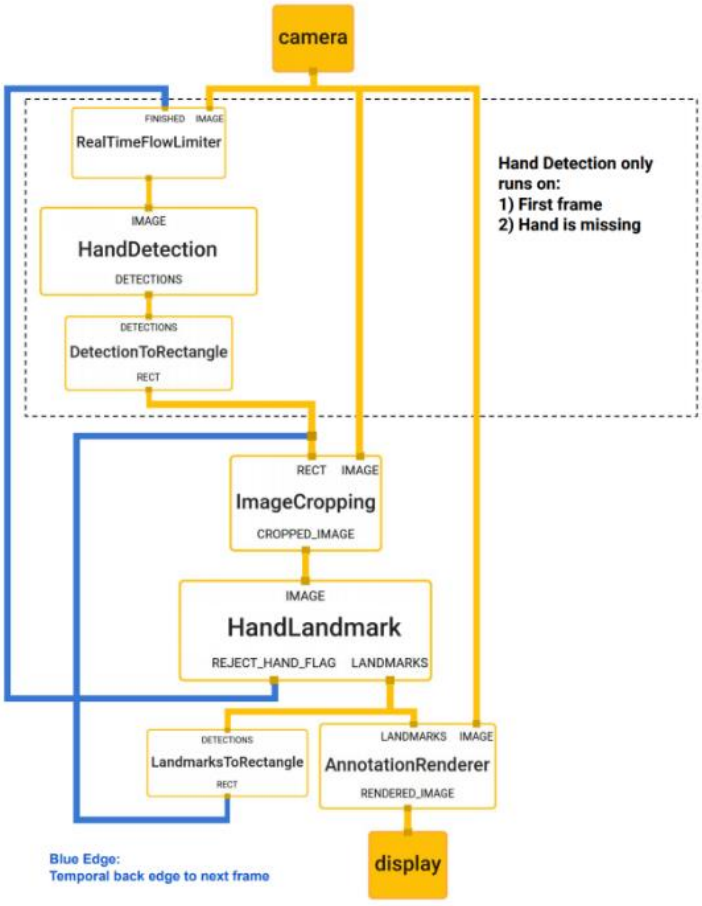


Рисунок 2.18 – Робота моделі MediaPipe Hands

Модель MediaPipe Hands показала найкращі результати в якості розпізнавання пози рук та швидкодії, саме чому і була обрана для подальшої розробки.

2.5 Висновок до другого розділу

В другому розділі кваліфікаційної роботи був проведений аналіз систем обробки потоку даних з використанням хмарних платформ, на прикладі сервісів Amazon Web Services та Google Cloud Platform як провідних провайдерів у галузі хмарних обчислень. Був складений опис продукту та вимоги до нього. Були розглянуті системи передачі та опрацювання потокових даних, Apache Kafka та група сервісів AWS Kinesis Family, на базі яких і приймалось рішення щодо використання сервісу при розробці програмного продукту. Також було проаналізовані і досліджені методи та моделі машинного аналізу зображень, які будуть виконувати основні вимоги поставлені перед продуктом. До них належать моделі розпізнавання об’єктів, засновані на одноетапному підході аналізу зображення – SSD та YOLO, системи розпізнавання обличчя, такі як Facenet, Dlib та OpenFace, і моделі оцінки пози рук – AWR та MediaPipe.

3 ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОДУКТУ

3.1 Пошук актантів та варіантів використання

На рисунку 3.1 зображена діаграма варіантів використання з акторами та прецедентами, що виникнуть у розробленому продукті машинного аналізу зображень.

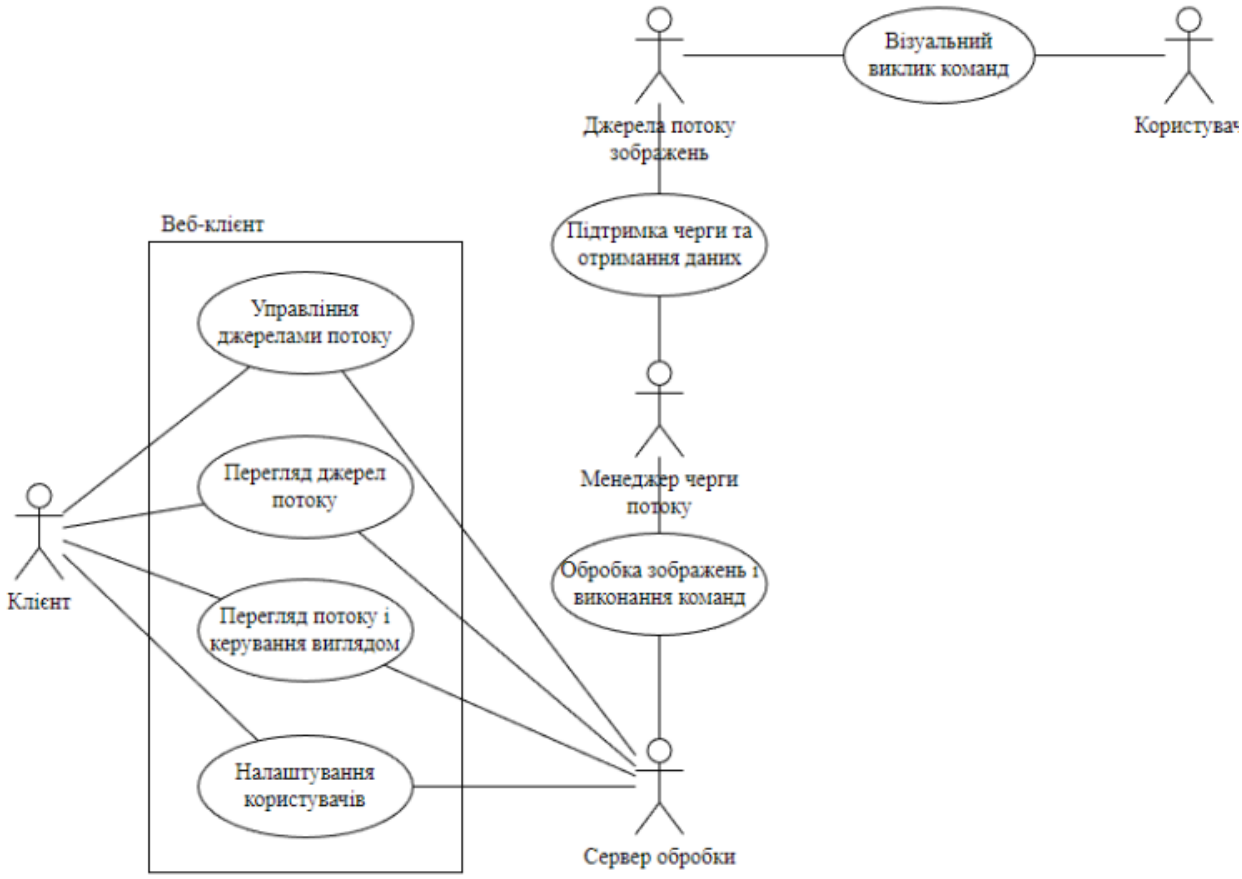


Рисунок 3.1 – Діаграма варіантів використання системи

В системі виділені п’ять основних акторів – Клієнт (актор, який взаємодіє з веб-клієнтом та налаштовує систему команд), Сервер обробки (актор, який виконує обробку запитів), Менеджер черги потоку (актор, що виконує зберігання потоку зображень та передає їх на обробку сервером обробки), Джерела потоку зображень (актори, які генерують зображення) та Користувач

(актор, який взаємодіє з Джерелами та відправляє запит на виконання програмованих команд).

Прецеденти виступають в якості основних функціональних вимог до продукту.

Користувач взаємодіє з Джерелами потоку зображень (наприклад, камерами) і подає візуальний запит на виклик конкретних програмованих команд (які були встановлені Клієнтом у веб-застосунку).

Джерела потоку зображень надсилають дані до Менеджера черги потоку, для подальшого їх збереження та обробки.

Менеджер черги потоку відправляє зображення на обробку до Сервера обробки, який запускає модулі ШІ (розпізнавання людей на зображенні, розпізнавання Користувача та розпізнавання візуальної команди) та виконує запрограмовану команду.

Клієнт, використовуючи веб-застосунок, взаємодіє з Сервером обробки, а саме отримує можливість налаштування Джерел даних, їх перегляд та налаштування Користувачів (їх створення) разом з програмуванням команд, які має виконати Сервер за умови, що певний Користувач надасть візуальну інформацію про запуск команди (номер команди).

Сервер обробки даних буде завжди обробляти зображення, які подають Менеджером черги потоку без взаємодії з Клієнтом. Проте, саме Клієнт забезпечує початкове налаштування Користувачів та програмованих дій Сервера.

3.2 Проектування архітектури продукту

Для реалізації усіх вимог та функціональних можливостей продукту необхідно створити складну систему, схема якої зображена на рисунку 3.2.

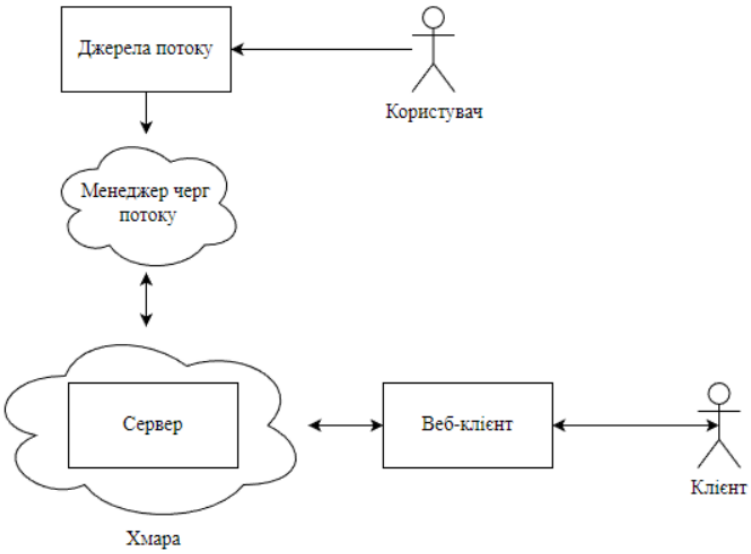


Рисунок 3.2 – Схема системи

Умовно схему можна розділити на наступні компоненти:

– Фізичні джерела потоку (камери та ін.), які будуть генерувати зображення від користувачів, та надсилати їх до Менеджера черг потоку.

– Менеджер черг потоку – хмарний сервіс, який буде мати власні розподілені ресурси для збереження та керування потоками зображень.

– Сервер обробки – сервер обробки запитів веб-клієнта та потоку зображень, для чого використовує три модулі штучного інтелекту – розпізнавання людей на зображенні, розпізнавання Користувача та розпізнавання візуальної команди. Для кращої ефективності та працездатності системи, сервер буде розміщуватися на хмарі.

– Веб-клієнт, який буде інтерфейсом взаємодії клієнта з сервером, і надавати можливість гнучкого керування та налаштування системи.

Оскільки багато елементів системи потребують безперервного двохнаправленого мережевого зв’язку в режимі реального часу буде використовуватися протокол WebSocket, який дозволить задовільнити усі вказані вимоги.

3.3 Опис програмних рішень

Створення сервера проводилось з використанням мови програмування Python та веб-фреймворку Flask. Для реалізації обміну повідомлень під протоколом WebSocket використовувалась програмна бібліотека socketio. Код реалізованого сервера знаходиться в Додатку Б.

Загалом серверну частину системи можна схематично розділити на декілька функціональних модулів:

– Модуль взаємодії з веб-клієнтом.

– Модуль підключень.

– Модуль розпізнавання людей.

– Модуль розпізнавання обличчя.

– Модуль розпізнавання рук.

– Модуль штучного інтелекту, який використовує три вищеперерахованих модулі-інтерфейси.

Модуль взаємодії з веб-клієнтом виконує обробку REST запитів, які надходять на веб-сервер. Для їх обробки були створені серверні функції, які отримують у якості параметрів адресу та HTTP-метод, за яким до них звертаються. При отриманні запиту сервер розпізнає кінцеву точку зв’язку і запускає відповідну системну функцію, яка обробить запит та поверне відповідний результат.

Перша складова модуля взаємодії – це група функцій, які виконують реєстрацію та авторизацію клієнта. Функція register() у випадку отримання GET запиту повертає сторінку реєстрації, а у випадку POST – виконує обробку форми та проводить реєстрацію клієнта на платформі. Функція login() при отриманні GET запиту повертає сторінку авторизації, у випадку POST – проводить автентифікацію клієнта на платформі, записуючи дані в сесію веб-браузера. Функція logout() очищує сесію та проводить деавторизацію. Функція login_required(view) працює по принципу шаблону Декоратор, і при виклику

перевіряє чи користувач зареєстрований, після чого дозволяє доступ до приватної сторінки веб-клієнта.

При підтвердженні автентифікації відкривається доступ до основного функціоналу веб-клієнта. Функція view_sources() отримує перелік усіх зареєстрованих джерел потоку зображень та дозволяє взаємодію з ними (створення нових, видалення, редагування та перегляд існуючих). Функція add_sources() відповідає за створення нового джерела потоку зображень. Функція source_edit(sid) при отриманні GET запиту повертає сторінку редагування джерела, а при POST – застосовує зміни та повертає до списку джерел. Функція source_remove(sid) застосовує видалення джерела з переліку існуючих. Функція view(sid) направляє на перегляд джерела потоку зображень, його користувачів та логування запуску команд користувачів. Функція change_visibility(sid) міняє наявність допоміжних відображень на зображенні потоку при перегляді з веб-клієнта. Функція create_person(sid) створює нового користувача потоку. Функція delete_person(sid, name) виконує видалення користувача з переліку. Функція edit_person(sid, name) застосовує вказані зміни до користувача.

Модуль підключень створює або реалізує підключення по протоколу WebSockets. Ці підключенні відбуваються або з веб-клієнтом, якому будуть надсилатися результати обробок, або з менеджером черг, для якого в безперервному режимі виконується отримання та обробка зображень, деталі яких будуть описані для модуля штучного інтелекту.

Модуль підключень складається з трьох функцій. Функція connect(sid, environ, auth, **kwargs) виконує перевірку автентифікації підключення. Функція join(sid, data) дозволяє створити для веб-клієнта особливий вид підключення сокетів – кімнату, яка полягає у відкритті окремого каналу зв’язку для багатьох підключень (клієнтів). Функція send(sid, data) реалізує отримання зображення, запуск модуля штучного інтелекту для обробки та відправлення

результатів у кімнату для всіх клієнтів, які підключенні до конкретного джерела.

Модуль розпізнавання людей займається попередньою обробкою зображення, його аналізом, використовуючи модель YOLOv8, для знаходження усіх людей у фреймі, допоміжним аналізом результатів розпізнавання та їх поверненням. Модуль складається з чотирьох функцій пост-обробки та класу PersonDetector, який запускає модель розпізнавання. Функція map_pred_to_dict(boxes) виконує трансформацію формату розпізнавання у формат словника. Функція get_intersection_over_box(bb1, bb2) здійснює перевірку пересічення рамок розпізнавання та повертає нормалізоване значення пересічення. Функція get_bbox_area(bbox) виконує обрахунок площі рамки. Функція get_duplicated_bboxes(bboxes, iob_threshold) здійснює пошук дублікованих рамок з розпізнавання людей за правилом пересічення рамок зі значенням більше за параметр iob_threshold. Клас PersonDetector виконує ініціалізацію моделі YOLOv8. Він містить метод detect(image), який отримує зображення, виконує розпізнавання людей на зображені, пошук і видалення дублікованих рамок та повертає результат. Та метод draw_bbox(image, bbox), який здійснює візуалізацію рамки на поданнному зображенні.

Модуль розпізнавання обличчя реалізований у вигляді класу FaceRecognizer, який використовує модель Dlib та складається з 4 методів. Метод encode_face(image) отримує фрагмент зображення, використовує модель розпізнавання лиць та вилучає ознаки лиць у вигляді вбудованих векторів. Метод encode_face_from_file(path) виконує аналогічний функціонал, що і метод encode_face, проте зчитує зображення з фізичної пам’яті. Метод compare_faces(known_encodings, unknown_encoding, tolerance) виконує порівняння невідомого обличчя з базою відомих лиць та знаходить найбільш схоже лице, яке задовільняє умову схожості меншим за параметр tolerance. Метод draw_name(image, name, left_corner_coords) здійснює візуалізацію імені знайденої людини.

Модуль оцінки пози рук реалізований у вигляді датакласу результатів HandResult, який зберігає інформацію про руку – сторону, статус кожного пальця (зігнутий чи розігнутий), число зігнутих пальців та координати орієнтирів руки, та класу LegacyFingerTracking, який складається з трьох методів. Метод transform(detection_results) отримує результати класифікації, проходиться по масиву результатів, перетворює його під формат датакласу HandResult і повертає результати в оновленому форматі. Метод detect(image) виконує розпізнавання зображення та класифікацію усіх наявних рук, після чого запускає метод transform() для перетворення формату і повертає кінцеві результати. Метод draw_landmarks_on_image(image, hands) здійснює візуалізацію орієнтирів рук на зображені.

Модуль штучного інтелекту реалізований у вигляді датакласу Person, який містить інформацію про розпізнану людину – координати обмежувальної рамки, ім’я людини (якщо була розпізнана модулем розпізнавання лиць), вбудований вектор ознак лиця, список з інформацією про руки (з типом HandResult), словник команд та число зігнутих пальців, та класу Image, який виконує обробку зображення та запускає три модулі штучного інтелекту, описаних вище. Він містить методи decode_image(image), який виконує декодування зображення, та encode_image(image), який здійснює кодування зображення в систему Base64. Захищений метод _person_detect(), який запускає метод detect класу PersonDetector, і виконує розпізнавання усіх людей на зображені. Захищений метод _recognize_faces(), який обрізає рамку для кожного з розпізнаних людей і запускає методи класу FaceRecognizer, а саме метод encode_face() для отримання вбудованого вектору ознак лиця (у випадку, якщо лице не було знайдене у рамці – присвоюється ім’я «No face detected») та виконує пошук людини методом compare_faces(), порівнюючи вбудований вектор з базою відомих лиць. Якщо лице не є відомим – присвоюється ім’я «Unknown». Захищений метод _hand_detect(), який запускає метод detect() класу FingerTracker, і виконує пошук усіх рук на зображені. Захищений метод

_draw_hand(), який запускає метод draw_landmarks_on_image() класу FingerTracker, і малює усі руки на зображені. Захищений метод _classify_hands(), який виконує присвоювання знайдених рук кожній з знайдених людей на зображені, використовуючи логіку близькості. І метод inference(), який послідовно запускає всі вище описані методи в порядку їх опису, кодує зображення методом encode_image та повертає зображення і список розпізнаних людей з їх особливостями у форматі датакласу Person.

3.4 Опис інтерфейсу веб-клієнта

Реалізація веб-клієнта здійснювалась з використанням мови розмітки HTML, таблиці стилів CSS та МП JavaScript. Веб-клієнт складається з п’яти сторінок – Авторизації, Реєстрації, Вибір джерел, Редагування джерел та Перегляд джерела.

Сторінка Авторизації зображена на рисунку 3.3 і складається з двох полей для вводу логіну та паролю, і двох кнопок – Підтвердження авторизації та Реєстрації. Перехід на сторінку Авторизації можливий або при використанні кнопки Log In у верхньому меню, або при переході на індексну сторінку веб-клієнта. У випадку, якщо клієнт не є авторизований на платформу – всі інші сторінки, крім Реєстрації, будуть недоступні та автоматично перенаправлятися на сторінку Авторизації.

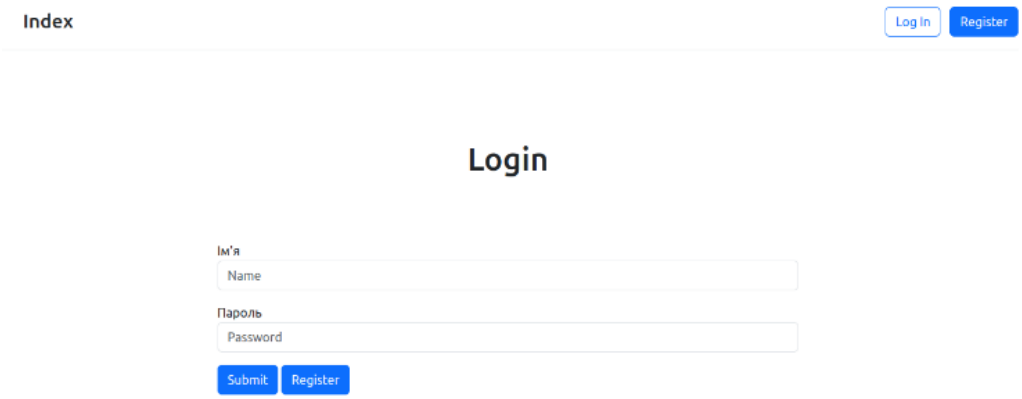


Рисунок 3.3 – Зовнішній вигляд сторінки авторизації

При натисканні на кнопку реєстрації у верхньому меню чи у формі авторизації на відповідній сторінці – клієнта перенаправить на сторінку Реєстрації, яка є зображена на рисунку 3.4. Сторінка складається з двох текстових полей для вводу – логіну та паролю і кнопки підтвердження реєстрації.

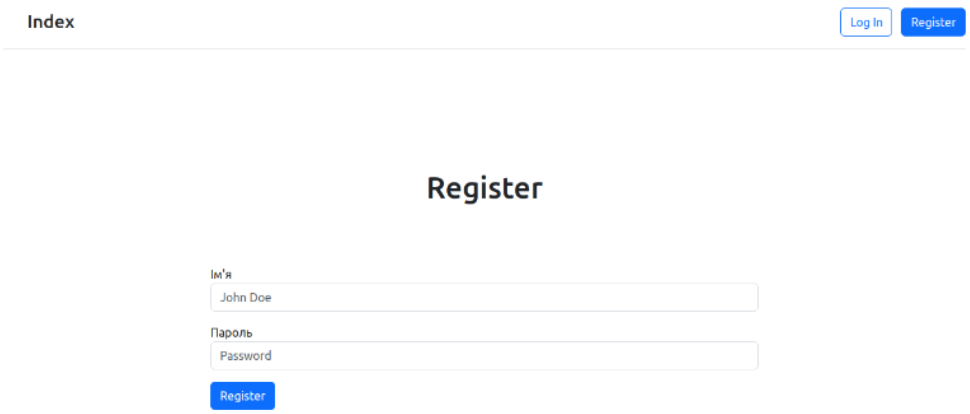


Рисунок 3.4 – Зовнішній вигляд сторінки Реєстрації

При успішній авторизації клієнта перенаправить на сторінку Вибору джерел, яка складається з переліку доступних зареєстрованих джерел, даних про них, кнопок для їх редагування та видалення і кнопки додавання нового джерела. Зовнішній вигляд сторінки Вибору джерел продемонстрований на рисунку 3.5.

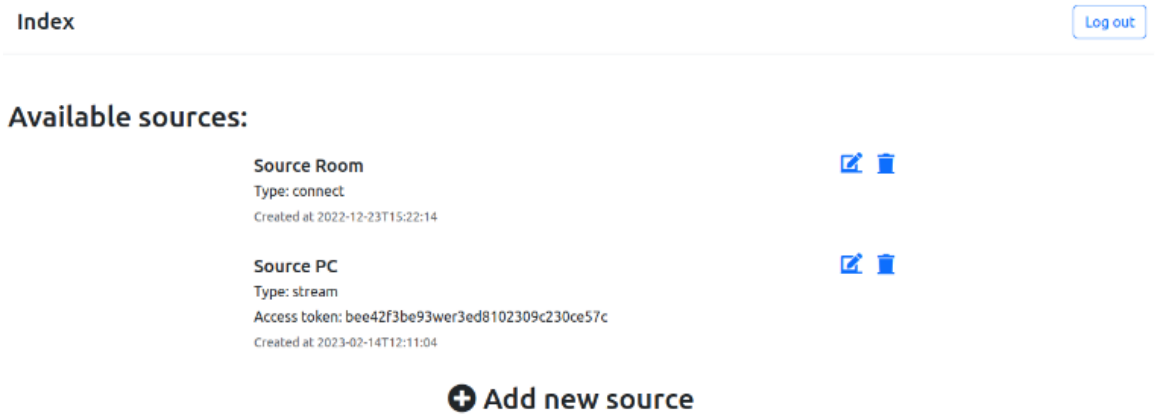


Рисунок 3.5 – Зовнішній вигляд сторінки Вибору джерел

При нажаті на кнопку Видалення джерела у вигляді корзини появиться вікно підтвердження видалення. При нажаті на кнопку Створення джерела появиться вікно Створення джерела, вигляд якого продемонстрований на рисунку 3.6. Це вікно складається з двох текстових полей для вводу – Назви джерела та Доступу до джерела, та поля вибору Типу джерела. При нажаті на кнопку Створити клієнта перенаправить на сторінку Вибору джерел з повідомленням про результати створення (успіх чи помилка).

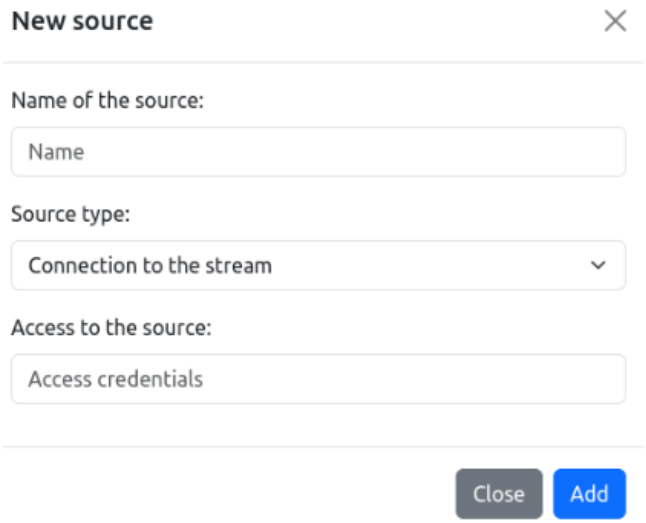
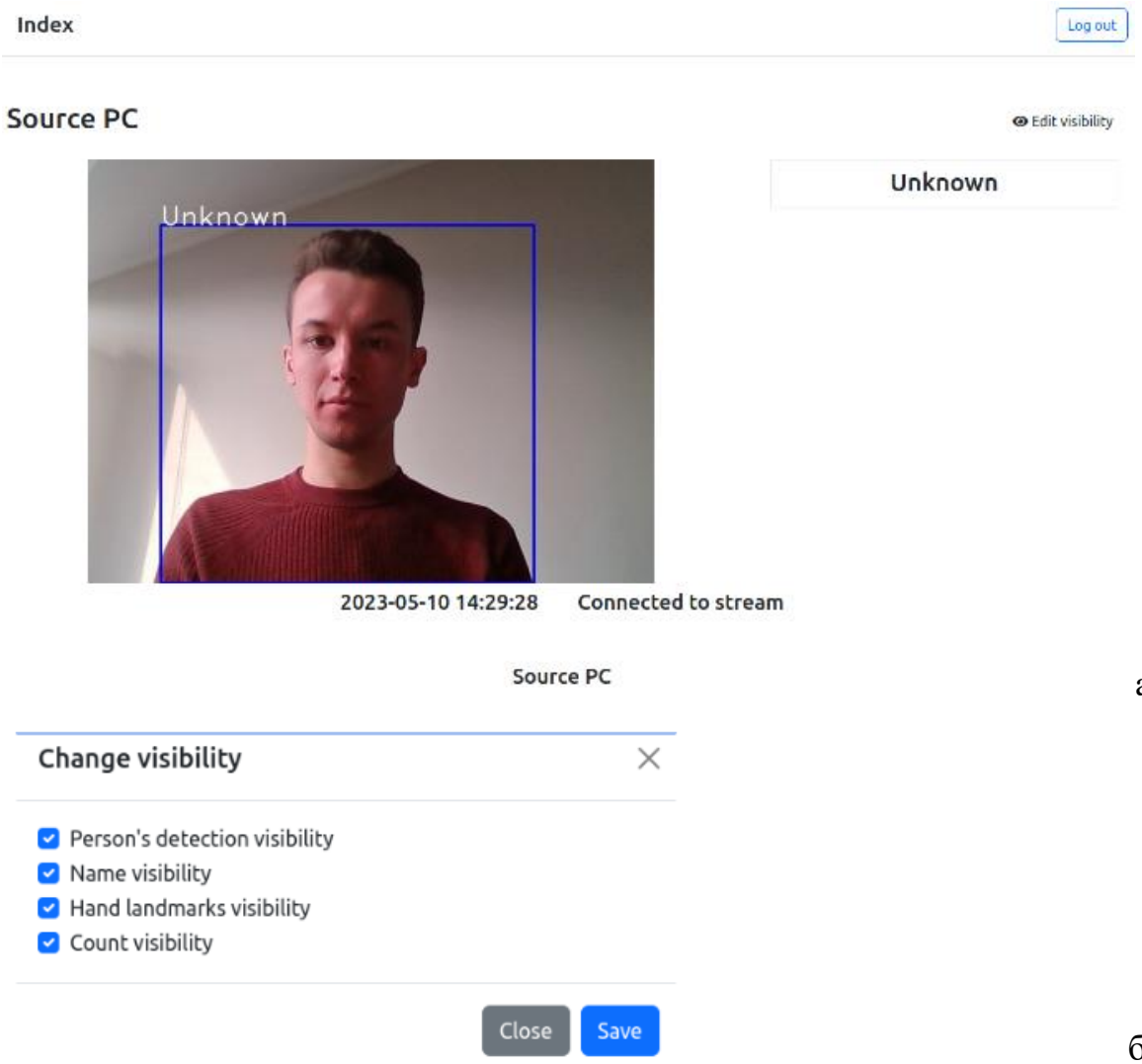


Рисунок 3.6 – Зовнішній вигляд вікна Створення джерела

При натисканні на кнопку Редагування джерела у вигляді іконки пера клієнта перенаправить на сторінку Редагування джерела, яка продемонстрована на рисунку 3.7. Вона схожа по вмісту на вікно Створення джерела, і складається з двох текстових полей для вводу – назви та токена доступу, і поля вибору. Всі дані заповнені тими значеннями, які стосуються обраного джерела. При збереженні редагування клієнта перенаправить на сторінку Вибору джерел з повідомленням про результати створення (успіх чи помилка).

Рисунок 3.7 – Зовнішній вигляд сторінки Редагування джерела

При виборі джерела клієнта перенаправить на сторінку Перегляду джерела, яка є продемонстрована на рисунку 3.8а. При переході на сторінку веб-клієнт з’єднується з каналом сервера через технологію WebSockets, який призначений для отримання потоку зображень та комунікацію з сервером. Сторінка складається з трьох блоків. Перший блок зліва – це демонстрування безперервного потоку зображень з джерела з додатковою допоміжною візуалізацією. Другий блок справа від зображень – це перелік знайдених Користувачів з можливістю реєстрації нового невідомого користувача чи редагуванням та видаленням існуючого. Над правим блоком зверху розташована кнопка зміни відображення допоміжної візуалізації у вигляді переліку опцій та поля вибору, приклад продемонстрований на рисунку 3.8б. Третій блок розташований знизу і показує допоміжну текстову інформацію про події, які зловив чи виконав сервер.



а)

б)

Рисунок 3.8 – Зовнішній вигляд сторінки Перегляду джерела (а – вигляд сторінки; б – вигляд вікна зміни допоміжної візуалізації)

При виборі Невідомого користувача у правому блоці з’являється вікно Створення Користувача, вигляд якого зображений на рисунку 3.9. Воно складається з одного текстового поля для вводу імені та кнопок Скасування і Створення. При нажаті на кнопку Створення веб-клієнт відправляє запит на створення до сервера і перенаправляє на сторінку Перегляду джерела.

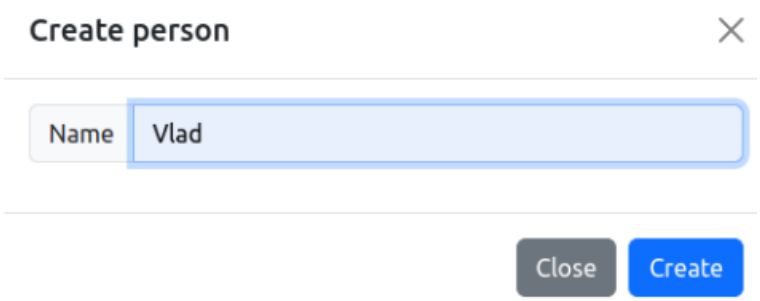


Рисунок 3.9 – Зовнішній вигляд сторінки

При знаходженні збереженого Користувача у потоці зображень сервер виконує його розпізнавання та пошук візуальної програмованої команди. На веб-клієнті це відображається як текстове повідомлення про виконання команди в блоці логування, приклад чого продемонстрований на рисунку 3.10.

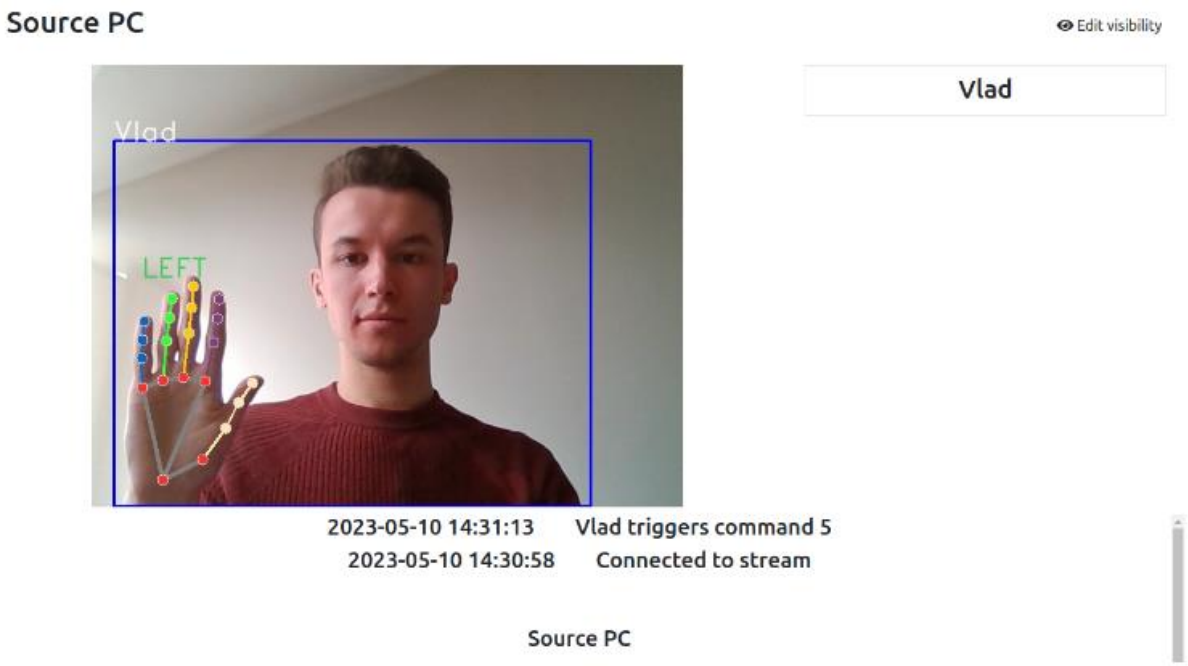


Рисунок 3.10 – Зовнішній вигляд сторінки

Також на зображені при ввімкнені відповідних допоміжних опцій з’являється відображення імені Користувача та відображення орієнтирів рук.

При натисканні на ім’я зареєстрованого Користувача у правому блоці відкривається вікно Перегляду Користувача, яке складається з полей для зміни

імені та вводу програмованих інструкцій для сервера (команд). Також є наявна можливість видалити Користувача. Зовнішній вигляд вікна перегляду Користувача продемонстрований на рисунку 3.11.

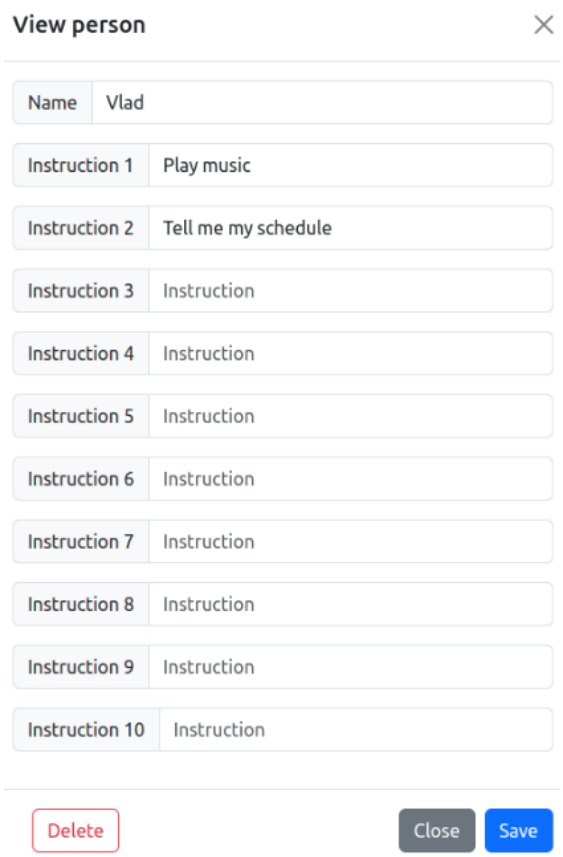


Рисунок 3.11 – Зовнішній вигляд вікна перегляду Користувача

При закриті вікна перегляду Користувача сервер оновляє для клієнта веб-сторінку перегляду джерел для застосування змін.

3.5 Розгортання системи та допоміжних сервісів на хмарній платформі

Одним із основних компонентів розробки хмарнох архітектури виступає Kinesis Video Streams. Він використовується для отримання потокового відео з різних джерел для подальшого опрацювання.

Для передачі відеоряду спочатку налаштовується сам потік на хмарі, що зображено на рисунку 3.12. Також потрібно вказати скільки часу відеопотік буде збережений для читання перед тим як видалиться (Data Retention).

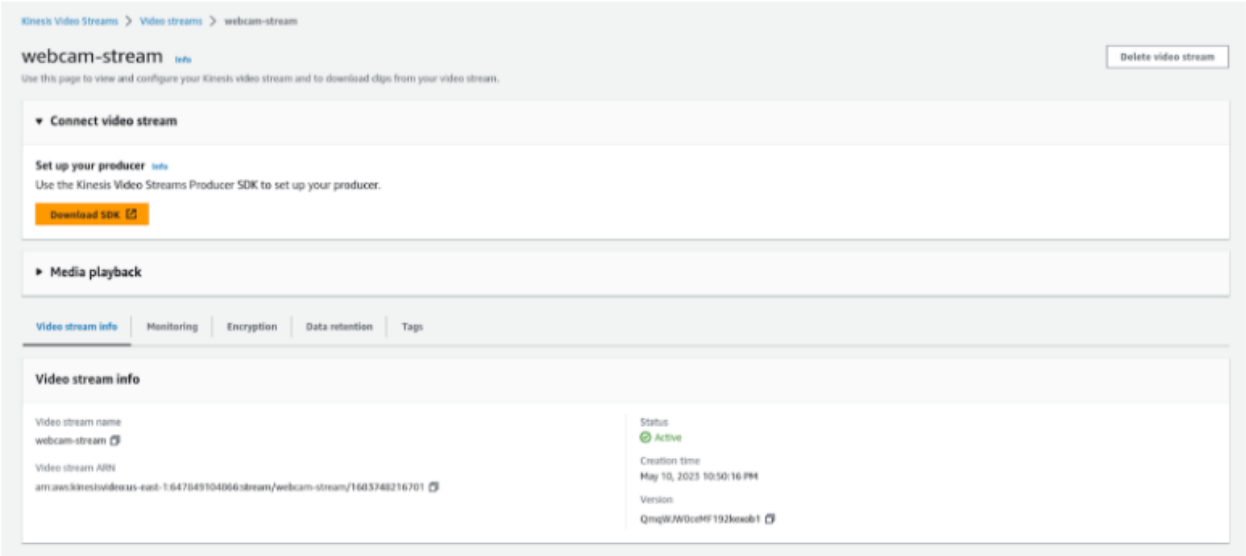


Рисунок 3.12 – Налаштований потік для отримання відео з камери

Далі, якщо камера не підтримує стрімінг даних, для відправки відеоряду завантажується спеціальне програмне забезпечення Kinesis Video Streams Producer Library. Завантаживши та встановивши програмне забезпечення, клієнт може вказувати якого розміру повинно бути відео, кількість кадрів в секунду, бітрейт, тощо. Приклад відправки відеоряду з терміналу операційної системи Linux використовуючи камеру приєднану по USB зображено на рисунку 3.13. Відправка відеоряду виконувалась за допомогою команди gst-launch-1.0 v4l2src do-timestamp=TRUE device=/dev/video0 ! videoconvert ! video/x-raw,format=I420,width=640,height=480,framerate=30/1 ! x264enc bframes=0 key-int-max=45 bitrate=500 ! video/x-h264,stream-format=avc,alignment=au,profile=baseline ! kvssink stream-name="webcam-stream" storage-size=512 access-key="access-key" secret-key="secret-key" aws-region="us-east-1". Дану команду можна розбити на декілька частин:

– gst-launch-1.0. Запускає сервіс під назвою GStreamer який і відповідає за відправку відеоряду.

– v4l2src. Опція яка відповідає за читання відеоряду з USB камери.

– device=/dev/video0. Вказується з якої камери читати відеоряд

– videoconvert ! video/x-raw,format=I420,width=640,height=480,framerate=30/1 ! x264enc bframes=0 key-int-max=45 bitrate=500 ! video/x-h264,stream-format=avc,alignment=au,profile=baseline !. Визначається якість відеоряду.

– kvssink stream-name="webcam-stream" storage-size=512 access-key="access-key" secret-key="secret-key" aws-region="us-east-1". Вказується куди саме буде відправлятись відео та передаються ключі доступу. Kvssink – це модуль, який встановлюється разом із Kinesis Video Streams Producer Library.

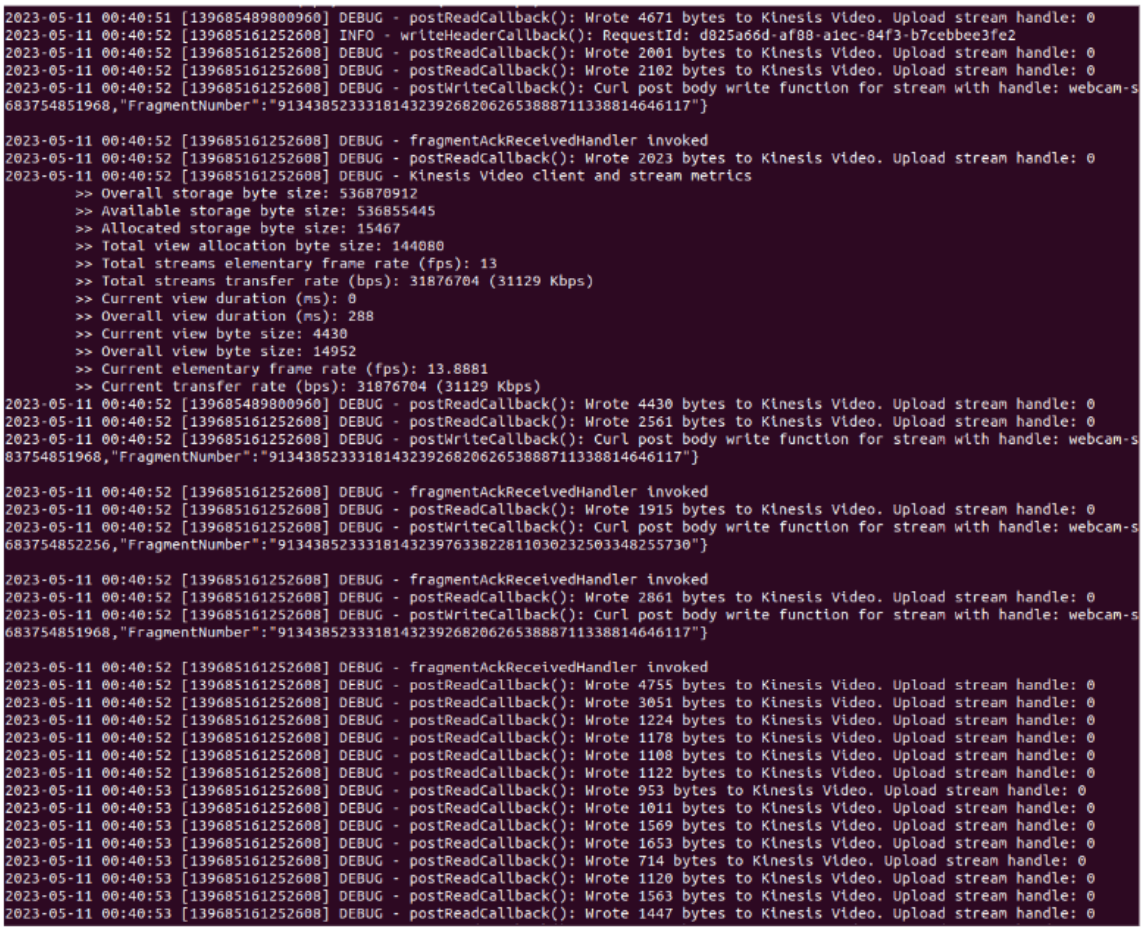


Рисунок 3.13 – Читання та відправка відеоряду з камери

При успішній відправці відеоряду з камери, його можна переглянути у самому AWS Kinesis Video Streams, у віконці Media playback Приклад продемонстрований на рисунку 3.14.

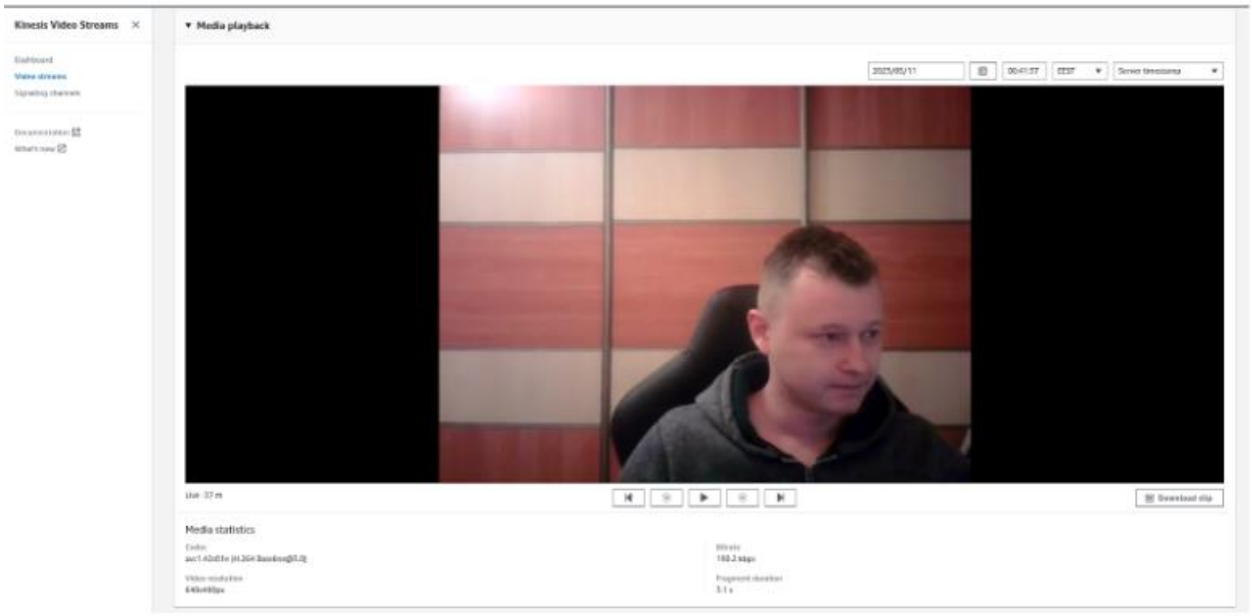


Рисунок 3.14 – Отриманий відеоряд у AWS Kinesis Video Streams

Для читання відеоряду використовується Kinesis Video Streams Parser Library. Це програмне забезпечення встановлено на машині разом з сервером, яке підключається по приватній мережі до AWS Kinesis Video Streams та читає вказаний потік. Прочитані кадри, Customer Library одразу відправляє на опрацювання серверу, який в свою чергу вже аналізує зображення та розпізнає об’єкти на ньому.

Наступним кроком у розгортанні хмарної архітектури буде хостинг сервера. Для презентації буде використовуватись Elastic Compute Cloud (EC2) instance. EC2 - є частиною платформи хмарних обчислень Amazon Web Services (AWS), яка дозволяє користувачам орендувати віртуальні комп’ютери для запуску власних програм. Для ефективного масштабування замість EC2 рекомендовано використовувати кластера з машин, для прикладу Kubernetes або AWS Fargate, це надасть можливість під кожен окремий відеопотік, підняти

відповідний сервер у вигляді контейнера який буде опрацьовувати відеопотік, та по завершненню, видаляти його, що зекономить кошти на простоюванні ресурсів. При налаштуванні EC2 було обрано тип операційної системи Linux для зручної взаємодії з машиною по протоколу SSH. Приклад виконання показаний на рисунку 3.15. Також було обрано стандартний Amazon Machine Image (AMI), він відповідає за додаткове програмне забезпечення яке встановлене на машині, для прикладу можна обрати AMI із встановленим MySQL Server.

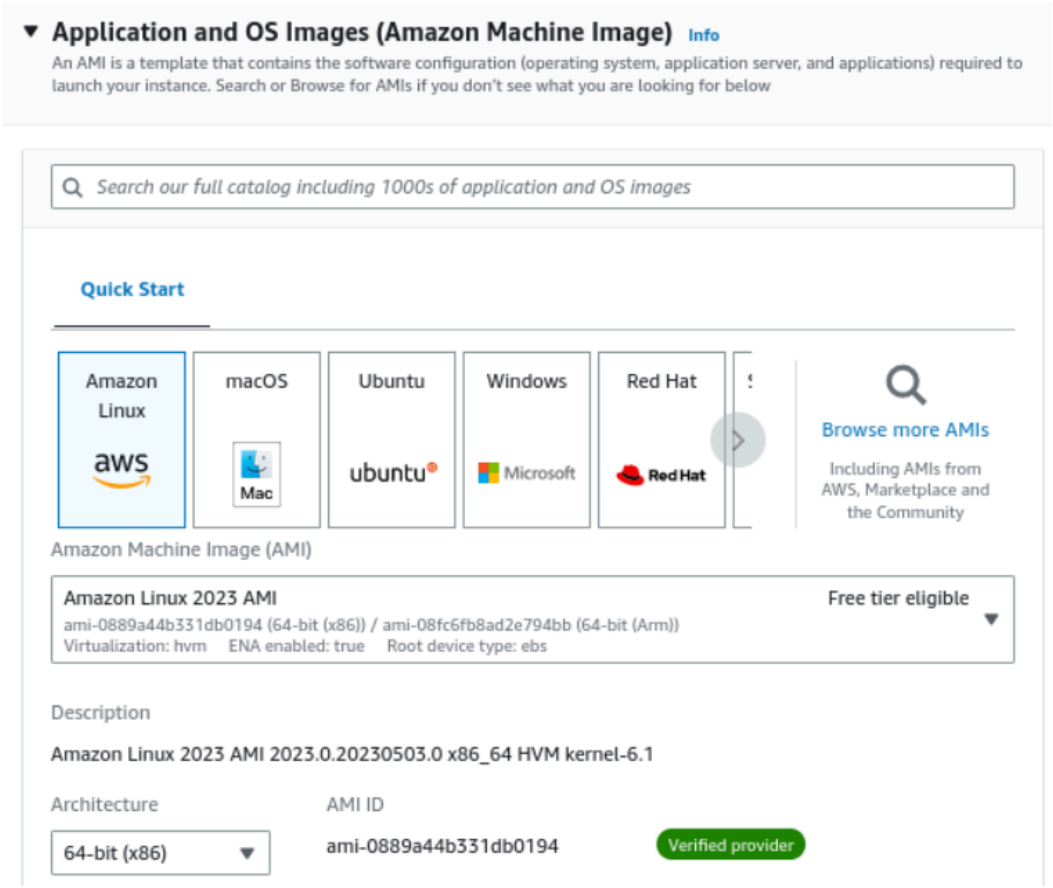


Рисунок 3.15 – Налаштування операційної системи та AMI

Наступним кроком було налаштування обчислювальних потужностей в машини, генерація ключів доступу та початкове налаштування мережі. Приклад виконання показаний на рисунку 3.16.

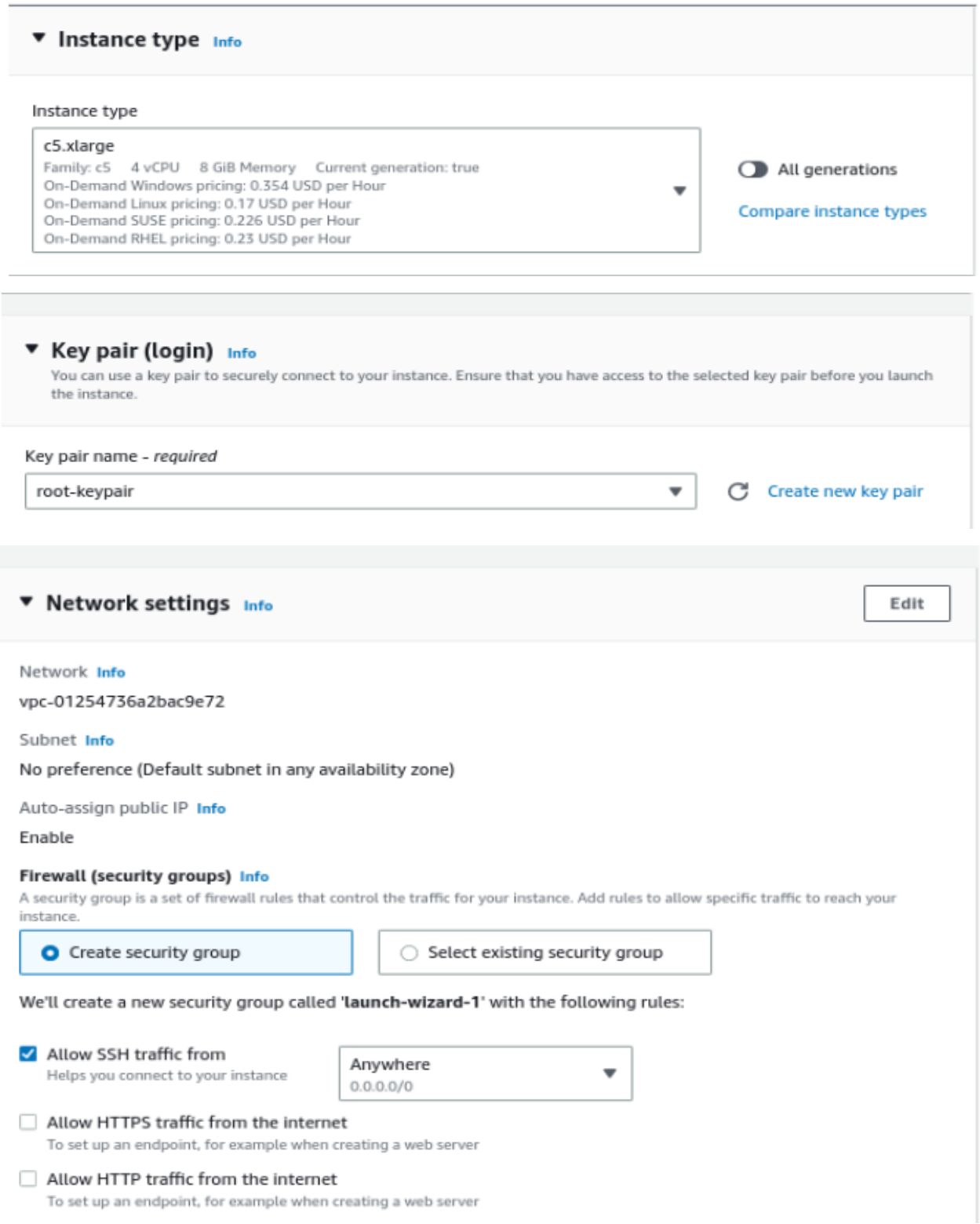


Рисунок 3.16 – Налаштування типу машини, ключів доступу та мережі

На даному етапі було обрано тим машини c5.xlarge, в якій передбачено 4 віртуальних CPU, 8gb оперативної пам’яті, а головне пропускна можливість мережі до 10 гігабіт, що дозволяє швидко опрацьовувати відеопотік без

навантажень на мережу. Наступним кроком було згенеровано ключ root-keypair.pem для доступу до машини по протоколу SSH. Останнім етапом було налаштування мережі. Вибрано VPC по замовчуванню, додано публічний IP адрес, створено нову Security Group та дозволено доступ по SSH через мережу.

Останнім етапом було налаштування об’єму та типу SSD диску(ів) на машині. Приклад виконання показаний на рисунку 3.17. Також можна додати EBS volume, додаткові зовнішні диски, вони є не обов’язкові для додавання на етапі створення машини, оскільки їх можна додавати вже під час роботи машини без додаткового перезапуску.

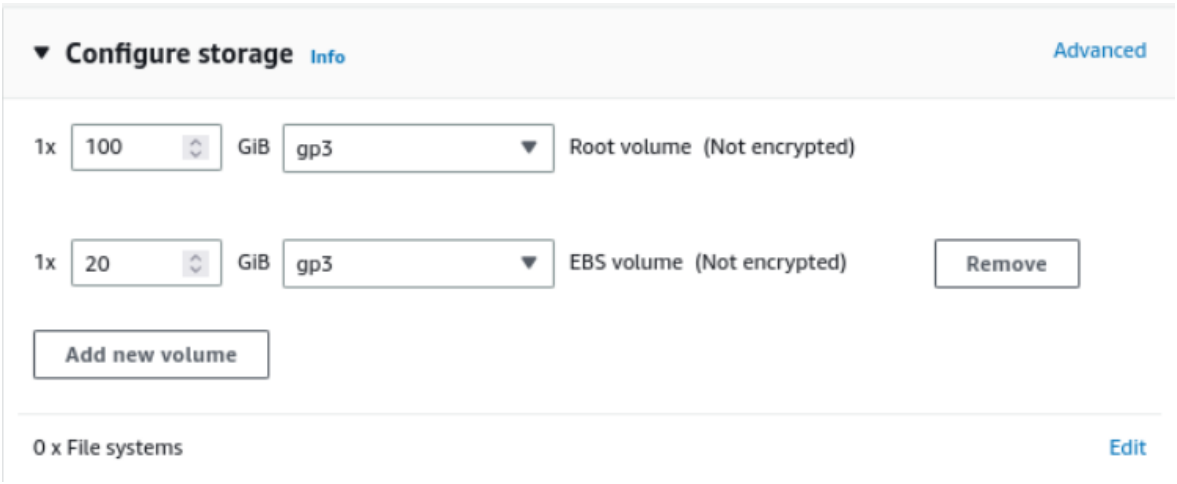


Рисунок 3.17 – Налаштування об’єму та типу SSD диску(ів)

Для підключення до машини по протоколу SSH використовується згенерований ключ root-keypair.pem. IP адрес можна переглянути в AWS веб консолі, у описі розгорнутої машини. Підключення відбувається з машини на базі операційної системи Linux використовуючи термінал, приклад чого продемонстрований на рисунку 3.18.

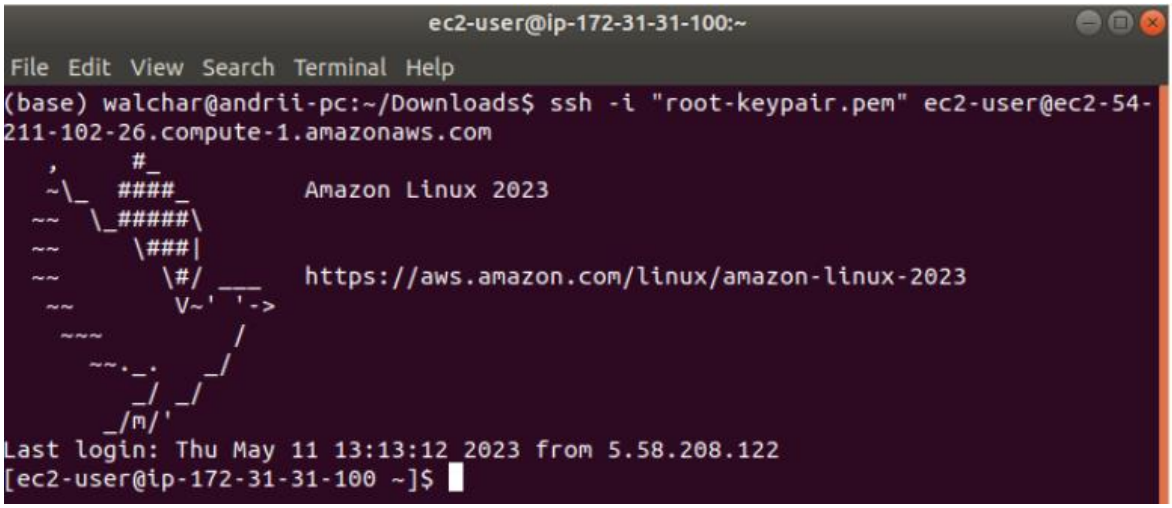


Рисунок 3.18 – Успішне підключення до віддаленої EC2 машини

Після підключення, копіюються усі файли проекту на машину використовуючи команду scp -i "root-keypair.pem" -r master/ ec2-user@ec2-54-211-102-26.compute-1.amazonaws.com:/home/ec2-user/src/.

Для запуску Python коду на сервері потрібно встановити додаткові бібліотеки перечислені в файлі /src/requirements.txt, для цього виконую команду python3 -m pip install -r /home/ec2-user/src/requirements.txt. Оскільки на сервері не буде запускатись сторонній Python код, можна обійтись без створення віртуального середовища (venv) для Python. Після встановлення всіх необхідних бібліотек залишається отримати потокові дані із Amazon Kinesis та налаштувати доступ до RDS бази даних. Для отримання потокових даних виконуються наступні кроки:

1. Клонується kinesis-video-streams-parser-library git repository https://github.com/aws/amazon-kinesis-video-streams-parser-library.
2. Використовуючи Java IDE, імпортується pom.xml файл в корені.
3. У Java IDE відкривається файл src/test/java/com.amazonaws.kinesisvideo.parser/examples/KinesisVideoRendererExampleTest.
4. Видаляється директива @Ignore з файлу.

5. Оновлюється параметр .stream назвою відеопотоку Kinesis (webcam-stream).

6. Запускається тест KinesisVideoRendererExample для перевірки читання потоку даних.

7. Після успішного читання потокових даних, сервер може виконати читання отриманих потокових даних з локального сокета.

Для успішної роботи сервера потрібно також налаштувати RDS базу даних, оскільки сервер використовує базу даних для збереження інформації про користувачів та роботи з ними. Для цього знаходимо RDS сервіс в AWS console, при створенні бази даних, на вибір буде даватись декілька різних двигунів. Для роботи з сервером було обрано найбільш поширений PostgreSQL, приклад розгортання якого показаний на рисунку 3.19.

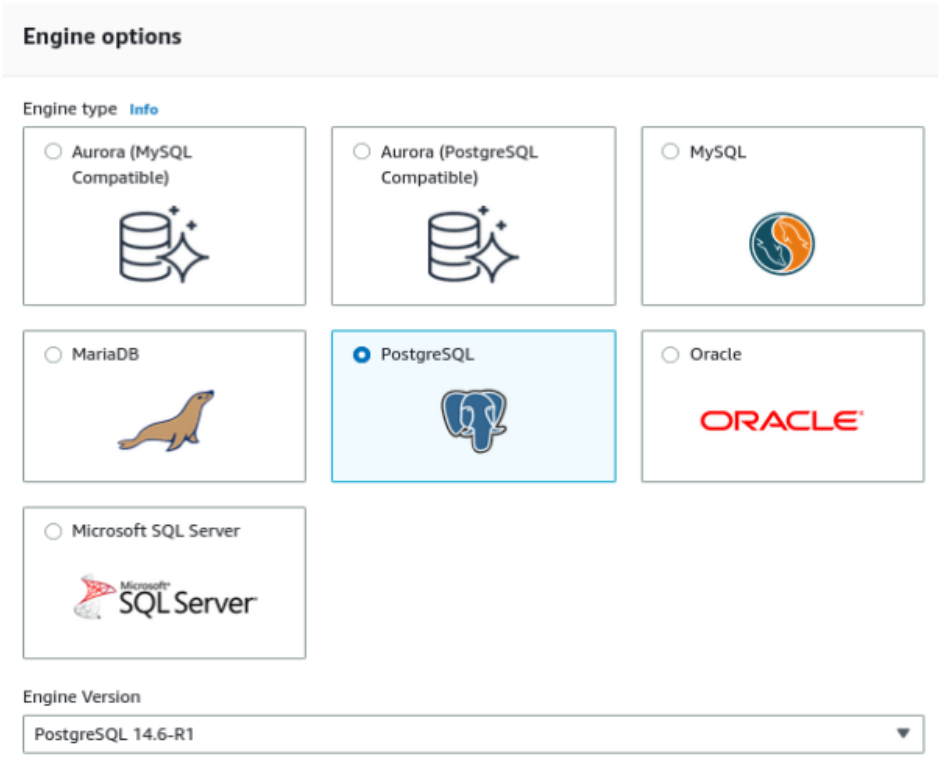


Рисунок 3.19 – Вибір двигуна бази даних в RDS

Обравши двигун, потрібно вказати ідентифікатор бази даних, а також налаштувати master користувача та пароль. Приклад виконання показаний на рисунку 3.20.

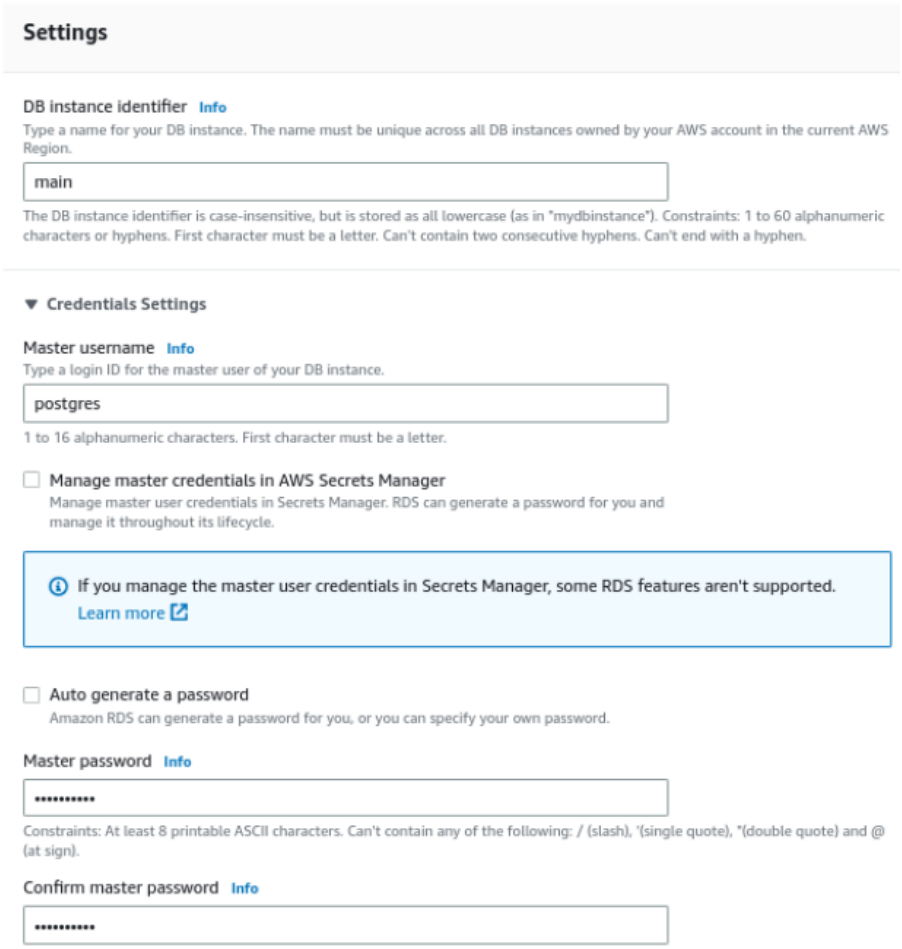


Рисунок 3.20 – Налаштування master користувача та встановлення паролю

Далі потрібно налаштувати базу даних так, щоб сервер отримав доступ до неї в приватній мережі AWS та заборонити доступ до бази даних з мережі інтернет. Для реалізації цього підходу AWS надає можливість при створенні бази даних вибрати EC2, яка повинна мати доступ до відповідної бази даних. Приклад виконання показаний на рисунку 3.21. Другий метод надання такого доступу, це використовуючи security групи в яких можна вказати з яких і до яких груп повинен бути доступ.

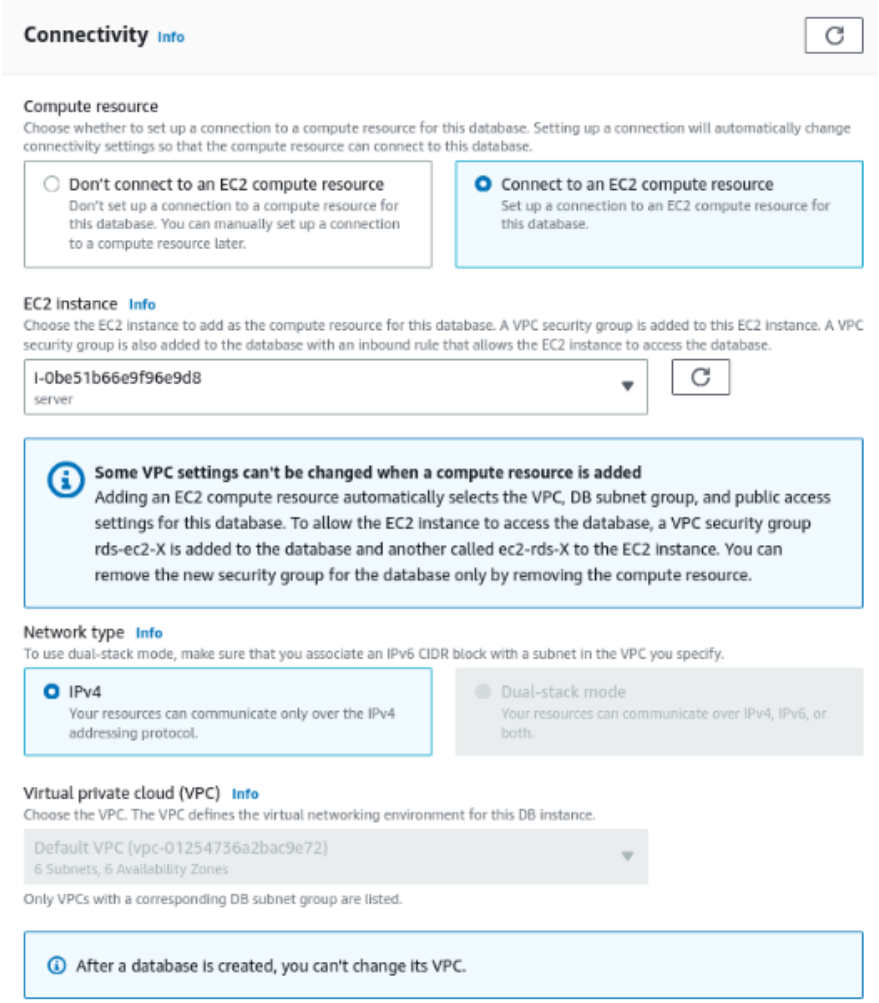


Рисунок 3.21 – Налаштування EC2 доступу до бази даних

Після створення бази, для перевірки доступу до бази даних, було запущено команду psql -h main.czsk02w3vjsy.us-east-1.rds.amazonaws.com -U postgres -d postgres –W, яка конектиться до бази даних.

Також на сервері потрібно змінити конфігураційний файл /src/server_conf.cfg та вписати правильні значення для доступу до бази даних.

Налаштувавши все, можна запускати сервер командою python3 -m main, проте на даному етапі прямого доступу до веб інтерфейсу не буде, для доступу потрібно налаштувати Security Group яка виступає фаерволом, додавши правило, яке дозволяє звертатись по порту 5000 до сервера, саме на цьому порті розгорнутий веб інтерфейс. Також потрібно обмежити доступ до сервера по

SSH надавши можливість підключатись тільки з одного комп’ютера (комп’ютера розробника).

Для тестування роботи фаерволу та сервера в цілому потрібно виконати запит по DNS адресі машини та порту 5000 на сторінку авторизації, приклад чого зображено на рисунку 3.22.

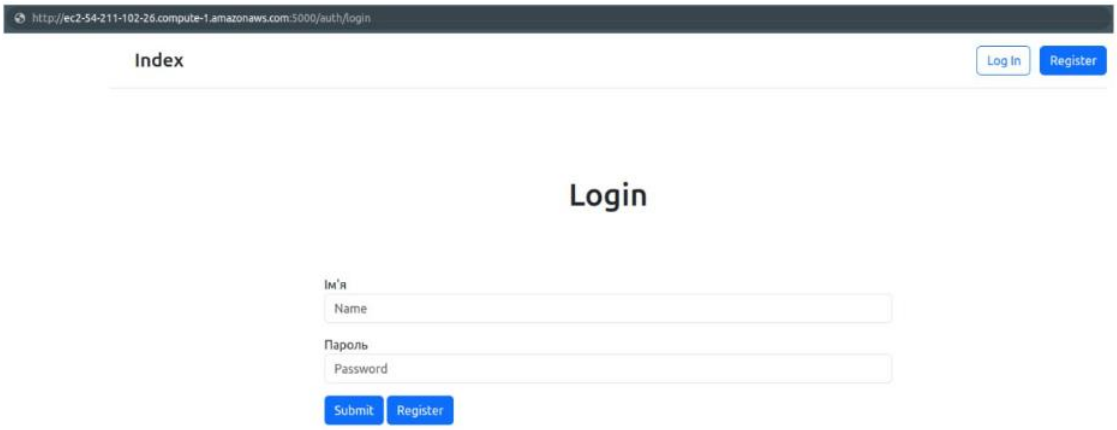


Рисунок 3.22 – Успішний запуск сервера та віддалений доступ до нього

Після проведених налаштувань, успішно отримано доступ до розгорнутого сервера на віддаленому сервері.

3.6 Висновок до третього розділу

В третьому розділі кваліфікаційної роботи був виконаний пошук та описані актанти та варіанти використання продукту. Спроектована архітектура продукту, яка складається з сервера, розташованого на хмарі, хмарного сервісу зберігання та обробки потоку зображень та веб-клієнта, що надає візуальний інтерфейс користувачам продукту. Також виконаний опис програмних рішень сервера та веб-клієнта з деталями кожного з модулів. Проведений опис інтерфейсу веб-клієнта та описані кроки розгортання системи на хмарній платформі.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Характеристика НС воєнного характеру

Надзвичайна ситуація – це стан або подія, яка виникає внаслідок непередбачуваного або надзвичайного події, і яка загрожує життю, здоров'ю, майну та навколишньому середовищу. Це може бути природна катастрофа, техногенна аварія, епідемія, терористичний акт, воєнний конфлікт або інша подібна небезпека.

Надзвичайні ситуації вимагають негайного реагування та координації з боку відповідних служб, органів управління, рятувальних команд та населення. Вони можуть призвести до евакуації, надання медичної допомоги, пошуку та порятунку, відновлення інфраструктури та інших заходів з метою мінімізації збитків і захисту людського життя.

Для ефективного управління надзвичайними ситуаціями існують спеціальні служби, такі як служби цивільного захисту, пожежна охорона, поліція, медичні команди та інші. Вони працюють впродовж доби для забезпечення безпеки, допомоги та відновлення після надзвичайних подій.

У кожній країні існують встановлені процедури та плани дій для реагування на надзвичайні ситуації, які допомагають забезпечити швидку та координовану реакцію на такі події і зменшити їхні наслідки.

Положення про класифікацію надзвичайних ситуацій за характером походження подій, котрі зумовлюють виникнення надзвичайних ситуацій на території України, розрізняє чотири класи надзвичайних ситуацій [55]:

- техногенного;
- природного;
- соціально-політичного;
- військового характеру.

Надзвичайні ситуації воєнного характеру – це ситуації, пов’язані з наслідками застосування зброї масового ураження або звичайних засобів ураження, під час яких виникають вторинні фактори ураження населення внаслідок зруйнування атомних і гідроелектричних станцій, складів і сховищ радіоактивних і токсичних речовин та відходів, нафтопродуктів, вибухівки, сильнодіючих отруйних речовин, токсичних відходів, нафтопродуктів, транспортних та інженерних комунікацій тощо [54, 55, 56].

Джерела небезпечних ситуацій у військовий час. Звісно першим і самим небезпечним джерелом є зброя. На даний час можна виділити такі види зброї:

1. Зброя масового ураження, яка в свою чергу розділяється на:

- ядерну зброю;
- хімічну зброю;
- біологічну зброю.

2. Звичайна зброя, яка застосовується при локальних і широкомасштабних бойових діях. Розрізняють багато видів звичайної зброї, але вся вона застосовується для знищення людей та матеріальних об’єктів. Наприклад, при застосуванні системи залпового вогню на площі близько 13 га будуть знищені всі споруди і майже 82% живої сили ворога.

3. Засоби радіоелектронної боротьби, які не ведуть до знищення споруд, але надзвичайно шкідливі для людини.

Іншим джерелом небезпеки є надзвичайна антисанітарна обстановка під час ведення бойових дій. Перш за все, це велика кількість трупів, які не завжди можна поховати (наприклад, у містах ведення інтенсивних бойових дій), по-друге порушується нормальна робота комунальних служб міст, що призводить до погіршення якості води, перебоїв каналізаційної системи і т. д.

Також завжди спостерігається зріст популяції гризунів і інших тварин, які завжди є переносниками хвороб. Також відчувається недостатнє медичне обслуговування, нестача медичних препаратів (більшість іде на фронт). Отже, створюється сприятлива ситуація для виникнення епідемій, особливо в теплі

місяці. Також багато людей можуть потерпати від звичайних хвороб які не зможуть ефективно лікуватися в умовах воєнного часу.

Третьою складовою є складна екологічна та техногенна обстановка. Треба враховувати, що сучасна війна не обходиться без значних руйнувань, які самі по собі являють загрозу життю людини і зазвичай супроводжуються пожежами.

Але ще більшу небезпеку несуть в собі підприємства, які й за мирних умов були джерелом небезпеки і шкідливих викидів. Хімічні підприємства, АЕС, нафтопереробні заводи у разі їх часткового або повного руйнування викличуть техногенну катастрофу і будуть становити значну небезпеку для життєдіяльності людей у районі розташування.

В сучасних умовах при виникненні широкомасштабної війни не викликає сумнівів використання зброї масового ураження і перш за все ядерної зброї. Розглянемо наслідки такого використання для життєдіяльності людини.

Ядерна зброя має декілька факторів ураження: світловий удар, тепловий удар, ударна хвиля та променеве ураження. Кожен з них становить велику небезпеку для життя і здоров’я людини. Світловий удар приводить до сліпоти, загорання одежі і предметів навколишньої обстановки, тепловий доповнює цей ефект.

Ударна хвиля руйнує будівлі і споруди, а послідуюче радіоактивне ураження робить перебування на зараженій території небезпечним для здоров’я та життя. Ядерний удар характеризується великим радіусом дії (120 км для бомби середньої потужності), великими втратами серед людей (90% в радіусі 100 км) і ще більшою кількістю опромінених. Після нього територія непридатна для проживання, а виниклі пожежі розносять за вітром радіоактивні елементи. Люди отримують опіки різного ступеня, механічні ушкодження від ударної хвилі і звичайно променеву хворобу. В залежності від її ступеня наступає смерть або розвивається лейкемія, ракові захворювання, значно послаблюється імунітет.

Звісно при ураженні поводяться евакуаційні заходи, люди переховуються в спеціальних сховищах з фільтрацією повітря і запасами води та їжі. Проводяться роботи по розбору завалів і локалізації та тушіння ядерних пожеж. Люди повинні мати засоби індивідуального захисту, які призначені для таких випадків. Район ураження локалізується і оточується для запобігання розповсюдження радіоактивного забруднення. Проводиться повна евакуація людей, особового складу формувань та техніки та їх повна дезинфекція після виведення з зон ураження. Ядерна зброя представляє найбільшу загрозу для життя і здоров'я людини.

Також можливе широке використання хімічної зброї, в основному газів і отруйних речовин для зараження водних ресурсів. Хімічні бойові речовини уражають слизові оболонки людей, очі та відкриті ділянки шкіри і маючи в основному нервово-паралітичну дію приводять до їх смерті.

Ще складніше ліквідувати наслідки дії біологічної зброї. Звичайно біологічна зброя застосовується в вигляді штамів різних хвороботворних бактерій, часто генетично змінених, які прищеплюються гризунам, комахам, рослинам для викликання епідемій і ураження живої сили противника і підриву його тилів.

Ознаки біологічної атаки виявити зразу складно і можливо піде не менше місяця до цього. Тому важливо проводити попереджуючий контроль, а при перших ознаках біологічної атаки оголошується карантин і всі сили кидаються на виявлення збудників хвороби та їх переносників. Проводиться дезинфекція міст зараження, дератизація, відлов бродячих тварин, захоронення трупів.

Можливе отруєння водних ресурсів бактеріями. В такому разі повинне бути забезпечене постачання незараженою водою з зовнішнього периметру. Складність полягає в тому, що наслідки біоатак можуть накластися на епідемії воєнного часу. Сполучення з зараженим районом строго обмежується і здійснюється тільки через КПП з відповідним санітарним контролем. Вся

техніка та люди з зараженого району мають пройти встановлений карантин і дезінфекцію.

При використанні звичайних видів зброї відбуваються значні руйнування та загибель великої кількості людей. При бомбардуванні і веденні бойових дій сучасною зброєю виникають пожежі які представляють значну загрозу життю та здоров'ю людей, особливо при застосуванні спеціальних запальних боєприпасів з напалмом та іншими горючими речовинами. Для таких випадків в підрозділах ГО та військ формуються спеціальні пожежні команди для допомоги в гасінні пожеж, недопущенні знищення матеріальних цінностей та людей.

Головним завданням таких команд є зразу ж після нанесення удару виявити, локалізувати та погасити пожежі до того як ті завдали шкоди майну і людям. Цивільне населення евакуюється з зони пожежі в першу чергу, а потім починається евакуація майна якщо зупинити вогонь на даному етапі неможливо.

Також при бойових діях і бомбардуваннях утворюються завали і під ними можуть бути заблоковані люди які не встигли сховатися в бомбосховища. Підрозділи ГО повинні розчищати завали, ліквідовувати небезпеку нових завалів, евакуювати населення з небезпечної зони. Але в першу черга повинна проводитися робота по вчасній евакуації населення в бомбосховища перед загрозою бомбардування. На цих роботах повинна в повній мірі застосовуватися спеціальна техніка МНС.

Особливу увагу слід приділяти техногенним катастрофам які можуть виникнути від руйнування у ході бойових дій екологічно небезпечних об'єктів. Візьмемо гіпотетичний приклад: бойові дії йдуть поблизу Києва. Для послаблення обороноздатності міста противних знищує Київську ГЕС. Наслідки для столиці катастрофічні – затоплені багато районів, перервані комунікації, знищені або в непрацездатному стані багато підприємств, зникла електрична енергія і звісно значні людські жертви.

Іншим прикладом може стати знищення під час бомбардування крупного хімічного підприємства з виробництва небезпечних для людей хім. речовин (тієї ж хімічної зброї). Можна тільки уявити до яких жертв це приведе. Ще більш непередбачуваними будуть наслідки удару по АЕС. Треба взяти до уваги, що противник завжди намагатиметься знищити або захопити важливі стратегічні об’єкти, як то електростанції, крупні промислові підприємства і т. п. Варто сказати що першочерговою задачею для військ є недопущення цього, але війна є війна і тому підрозділи ГО, особливо у великих промислових центрах, завжди повинні бути готові мінімізувати наслідки будь-яких таких ситуацій.

Особливою статтею слід виділити загрозу виникнення епідемій у зв'язку з антисанітарними умовами. Як було сказано вище не завжди є можливість поховати останки загиблих під час воєнних дій, що приводить до значного зростання ризику спалаху інфекційних захворювань. До того ж бродячі тварини, гризуни та комахи які мають контакт з тілами можуть стати переносниками інфекцій.

В умовах війни та неможливості дотримуватися правил гігієни, бруду спостерігається розквіт вошей, бліх та інших паразитів які є переносником такої небезпечної інфекції як тиф, свербіж і т. п. Також вода та продукти можуть бути заражені різними бактеріями у зв'язку з попаданням у них тіл та продуктів життєдіяльності зі збудниками інфекцій. Особливо небезпечною ситуація стає в теплі пори року, коли всі мікроорганізми розмножуються надзвичайно швидко.

Загроза виникнення надзвичайної ситуації будь якого класу чи рівня – це реальна загроза для життя і здоров'я людей, загроза порушення нормальних умов їх життя і діяльності або ж значних матеріальних втрат. Завданням безпеки життєдіяльності як галузі науково-практичної діяльності є захист здоров'я та життя людини і середовища її проживання від небезпек, а також розробка і реалізація відповідних засобів та заходів щодо створення і підтримки здорових та безпечних умов життя і діяльності людини. Виконання цього

завдання особливо гостро стоїть під час загрози виникнення та при виникненні надзвичайних ситуацій [55].

4.2 Оцінка стійкості роботи економіки до впливу поражаючих факторів ядерної зброї

Стійкість роботи об'єкта – це здатність його в надзвичайних ситуаціях випускати продукцію у запланованому обсязі, необхідної номенклатури і відповідної якості, а у випадку впливу на об'єкт вражаючих факторів, стихійних лих та виробничих аварій – в мінімально короткі строки відновити своє виробництво [53].

Залежить вона від таких основних факторів:

– розміщення об'єкту відносно великих міст, об'єктів атомної енергетики, хімічної промисловості, великих гідротехнічних споруд, воєнних об'єктів та ін.;

– природно-кліматичних умов, технології виробництва;

– надійності захисту працюючих, населення від впливу вражаючих факторів, наслідків стихійних лих і виробничих аварій, катастроф;

– надійності системи постачання об'єкту всім необхідним для виробництва продукції (паливом, мастилами, електроенергією, газом, водою, хімічними засобами захисту рослин, ветеринарними засобами, мінеральними добривами, запасними частинами, технікою та ін.), здатності інженерно-технічного комплексу протистояти надзвичайним ситуаціям;

– стійкості управління виробництвом і ЦО, психологічної підготовленості керівного складу, спеціалістів і населення до дій в екстремальних умовах;

– навченості командно-керівного складу ЦО об'єкту і населення правильно виконувати комплекс заходів цивільної оборони;

– масштабів і ступеня вражаючої дії стихійного лиха, виробничої аварії, катастрофи чи зброї і підготовленість об'єкту до ведення рятувальних та інших невідкладних робіт для відновлення порушеного виробництва. Основні вражаючі фактори ядерного вибуху – це ударна хвиля, світлове випромінювання, проникаюча радіація, радіоактивне зараження місцевості, електромагнітний імпульс.

Ударна хвиля – основний вражаючий фактор ядерного вибуху. Більшість руйнувань і ушкоджень споруд, будинків, а також поразки людей, як правило, обумовлені її впливом. Джерело її виникнення – величезне тиск, що утворюється в центрі вибуху і досягає у перші миті мільярдів атмосфер. Передня межа стисненого шару повітря називається фронтом ударної хвилі. Ступінь поразки ударною хвилею людей і різних об'єктів залежить від потужності і виду вибуху, а також від відстані, на якому стався вибух, рельєфу місцевості і положення об'єктів на ній.

Швидкість руху і відстань, на яку поширюється ударна хвиля, залежать від потужності ядерного вибуху. Зі збільшенням відстані від місця вибуху швидкість швидко падає. Так, при вибуху боєприпасів потужністю 20 кт ударна хвиля проходить 1 км за 2 с; 2 км – за 5 с, 3 км – за 8 с. За цей час людина після спалаху може укритися й тим зменшити ймовірність ураження ударною хвилею або взагалі уникнути поразки.

Стійкість роботи об'єкту економіки – це здатність його в надзвичайних ситуаціях випускати продукцію у запланованому обсязі, необхідної номенклатури і відповідної якості, а у випадку впливу на об'єкт вражаючих факторів, стихійних лих та виробничих аварій – в мінімально короткі строки відновити своє виробництво.

Більш підготовленими до стійкої роботи будуть ті об'єкти економіки, які реально оцінять фактори, їх несприятливий вплив на виробництво і розроблять відповідні заходи. Завчасне проведення організаційних, агрохімічних, агротехнічних, інженерно-технічних, ветеринарно-санітарних, лісотехнічних,

лісогосподарських, меліоративних та інших заходів максимально знизить результати впливу вражаючих факторів мирного і воєнного часу на людей, сільськогосподарських тварин і створить сприятливі умови для швидкої ліквідації наслідків надзвичайної ситуації.

Для розробки заходів підвищення і забезпечення стійкості роботи об'єктів у надзвичайних ситуаціях необхідно оцінити стійкість об'єкту проти випливу вражаючих факторів.

Вихідними даними для проведення розрахунків стійкості об'єкта до ураження є:

– максимальні значення параметрів можливих вражаючих факторів;

– характеристики елементів об'єкта.

Дія ударної хвилі на об'єкт характеризується складним комплексом навантажень:

– надмірним тиском;

– тиском відбивання;

– тиском швидкісного напору;

– тиском затікання.

Все це буде залежати від виду і потужності вибуху, відстані до об'єкта, конструкції й розмірів елементів об'єкта, орієнтації відносно вибуху, розміщення будівель і споруд, рельєфу місцевості. Врахувати їх разом для кожного об'єкта неможливо.

Тому опір конструкцій до дії вибухової хвилі прийнято характеризувати надмірним тиском у фронті ударної хвилі який призводить до слабких, середніх і сильних руйнувань.

Для оцінювання стійкості об’єкта до дії повітряної ударної хвилі необхідно провести розрахунок значення надлишкового тиску який є основним критерієм оцінки стійкості об’єкта до дії ударної хвилі (УХ).

Критерієм стійкості об’єкта до дії УХ є граничне значення надлишкового тиску, за якого елементи об’єкта зберігаються або отримують слабкі та середні

руйнування. Це значення надлишкового тиску називають границею стійкості об’єкта до УХ і позначають ΔРфгран .

Стійкість об’єкта оцінюють для екстремальних умов. Умови стійкості об’єкта такі:

– якщо ΔРфmax . ⩾ ΔРфгран . – об’єкт нестійкий;

– якщо ΔРф max . < ΔРфгран . – об’єкт стійкий до дії УХ Методика оцінювання стійкості об’єкта до дії УХ включає:

– розрахунок максимального значення надлишкового тиску УХ, що очікується в районі об’єкту ΔРфmax;

– розрахунок границі стійкості об’єкту до дії УХ, ΔРфгран .

Спочатку з технічної документації виділяють основні елементи об'єкта і їх характеристики. Потім визначається межа (границя) стійкості кожного з основних елементів об'єкта. Границею стійкості елемента є надмірний тиск, при якому елемент дістане середню ступінь зруйнувань. Якщо надмірний тиск, при якому елемент отримує середні руйнування, визначений не одним значенням, а діапазоном (наприклад, 20...30 кПа), то за границю стійкості приймають нижню межу діапазону (у прикладі 20 кПа). За границю стійкості об'єкта в цілому приймають границю стійкості найбільш слабкого елемента об'єкта:

– аналіз результатів оцінювання: висновок – чи стійкий об'єкт чи ні. Які з елементів найменш стійкі. До якої величини доцільно підвищувати стійкість об'єкта;

– визначення заходів щодо підвищення стійкості об'єкту.

Таким чином в даному підпункті розглянуто питання оцінки стійкості об’єктів економіки до дії ударної хвилі ядерного вибуху. На основі результатів оцінки стійкості об'єкта роблять висновки і заходи по кожному елементу і об'єкту в цілому. Такими заходами можуть бути:

– укріплення несучих конструкцій та перекриттів будівель установленням додаткових колон, ферм, контрфорсів або підкосів;

– розміщення обладнання на нижніх поверхах будівель або в підвалах, надійне закріплення на фундаменті, установлення захисних кожухів або ковпаків;

– прокладання кабельних мереж та трубопроводів під землею;

– створення резервних запасів обладнання, апаратури, матеріалів для відновлення виробництва.

4.3 Міжнародні норми соціальної відповідальності. Стандарт SA 8000 «Соціальна відповідальність»

У сьогоднішніх умовах діяльності соціальна відповідальність стає невід'ємною складовою соціальної стратегії, яка забезпечує конкурентоспроможну позицію підприємства на внутрішньому та міжнародному ринку. Найбільш актуальним є питанням формування соціальної стратегії у сфері послуг, до якої належать різного роду підприємства, адже запорукою надання якісних послуг є, в тому числі, і їх соціальна спрямованість: доступність цін, асортимент послуг, охоплення кола споживачів, урахування особливостей попиту і ринку в цілому. Крім того, не останнє місце при формуванні соціальної стратегії займають і трудові відносини, які складаються на підприємстві. До того ж всі ці аспекти регулюються як українським законодавством, так і міжнародними стандартами [52].

Аналіз останніх досліджень і публікацій. Питаннями соціальної відповідальності бізнесу на сьогоднішній день займається досить значна кількість як науковців, так і практиків в Україні та за її межами. Серед таких можна зазначити: О. Амосова, Д. Баюру, Р. Колишко, А. Куліша, О Лазоренко, О. Петроє, Н. Супрун та інших. Поглиблене наукове дослідження з даної тематики пов'язане з поєднанням стратегічного управління підприємством з його соціальною відповідальністю, а саме, вивчення можливостей формування соціально-економічної стратегій розвитку організації, яка б ґрунтувалася на

соціальній відповідальності бізнесу, що, в свою чергу, сприяло б підвищенню економічної ефективності діяльності підприємства..

Основними міжнародними стандартами, які регламентують соціальну відповідальність підприємств є стандарт SA 8000 «Соціальна відповідальність», стандарт ISO-26000 «Керівництво з соціальної відповідальності» та Глобальний договір ООН.

Стандарт SA 8000 був опублікований у 1997 році, переглянутий – у 2001 році. Мета стандарту – сприяти постійному поліпшенню умов наймання працівників і здійснення трудової діяльності, виконання етичних норм цивілізованого суспільства. Стандарт SA 8000 був створений для того, щоб компанії могли підтвердити використання соціально-відповідальних підходів у своїй діяльності. У світі вже давно прийнято, що підприємства, на яких приділяється значна увага персоналу, створенню необхідних і комфортних умов для роботи, є надійними партнерами у взаємовідносинах. Використання етичних підходів до суспільства в цілому і до своїх співробітників, зокрема створення сприятливої атмосфери в колективі, є критерієм високого рівня менеджменту. І навпаки, співробітництво з компаніями, які не виконують ці вимоги, вважається неетичним і пов'язаним з додатковими ризиками. Стандарт SA 8000 спрямований на забезпечення привабливості умов праці для співробітників, поліпшення умов їхньої праці і життєвого рівня. Компанії, у яких менеджмент здійснюється відповідно до вимог стандарту SA 8000, мають конкурентну перевагу, яка полягає у високій мотивації персоналу, що у свою чергу дозволяє ефективніше застосовувати сучасні системи менеджменту для досягнення намічених цілей, забезпечуючи при цьому постійну рентабельність. А наявність на підприємстві інших стандартів забезпечуючи основу для інтеграції в рамках загальної системи менеджменту, що веде до скорочення ризиків і підвищення прибутковості компанії.

В свою чергу Стандарт ISO 26000 – це добровільна настанова з соціальної відповідальності і не є документом, що передбачає сертифікацію, як,

наприклад, ISO 9001 та ISO 14001. Згідно ISO 26000 компанія включає такі компоненти, як захист прав людини, навколишнього природного середовища, безпеку праці, права споживачів та розвиток місцевих общин, а також організаційне управління та етику бізнесу. Тобто у керівництві враховані усі принципи, які зазначені у Глобальній ініціативі ООН (документі, до якого приєдналося 6 тисяч компаній та організацій, серед яких 130 українських).

Стандартом ISO-26000 «Керівництво з соціальної відповідальності», визначаються основні принципи соціальної відповідальності: Підзвітність, яка полягає в тому, що організація має звітувати щодо впливу від своєї діяльності на суспільство і довкілля.

Прозорість, яка означає, що організації слід бути прозорою в її рішеннях і діяльності, які впливають на інших. Організація повинна розкривати в зрозумілій, збалансованій і правдивій формі про політику, рішення та діяльність, за які вона несе відповідальність, включаючи їх фактичний і можливий вплив на суспільство і довкілля. Ця інформація має бути легкодоступною і зрозумілою для всіх заінтересованих сторін. Прозорість не має на увазі розкриття службової інформації, а також інформації, що захищена відповідно до законів або може спричинити порушення правових зобов'язань.

Етична поведінка. Організація повинна приймати і застосовувати стандарти етичної поведінки, які якнайповніше відповідають її призначенню і сфері її діяльності. Організація повинна розвивати структуру управління так, щоб вона сприяла поширенню принципів етичної поведінки як усередині організації, так і в процесі взаємодії з іншими.

Взаємодія з зацікавленими сторонами – цей принцип означає, що організації слід поважати, розглядати інтереси її заінтересованих сторін та всемірно взаємодіяти з ними.

Правові норми – У контексті соціальної відповідальності повага правових норм означає, що організація дотримується всіх чинних законів і правил, вживає заходів, аби бути обізнаною про застосовані нею закони і правила,

інформувати тих осіб в організації, хто відповідальний за дотримання законів і правил, і знати, що такі закони і правила дотримуються.

Міжнародні норми – цей принцип визначає, що організації слід поважати міжнародні норми, в тих випадках, коли ці норми є важливими для сталого розвитку і добробуту суспільства.

Права людини – цей принцип означає, що організація повинна визнавати важливість і загальність прав людини, поважати права, зазначені у всесвітній декларації з прав людини.

Цей міжнародний стандарт надає інструкції щодо основних принципів соціальної відповідальності, ключових тем та питань, що мають відношення до соціальної відповідальності, а також щодо шляхів впровадження соціально відповідальної поведінки до існуючих стратегій, систем, практик та процесів організації. Він наголошує на важливості результатів та діяльності у сфері соціальної відповідальності та її удосконалення.

Цей міжнародний стандарт розроблявся як такий, що мав би бути корисним для всіх типів організацій у приватному, державному та неприбутковому секторах, для компаній великих і малих, діючих у розвинутих країнах або країнах, що розвиваються. Хоча не всі розділи цього міжнародного стандарту будуть однаково корисними для всіх типів організацій, всі основні аспекти є придатними для кожного типу організацій. Кожна організація самостійно визначає, що є придатним та важливим для неї шляхом власних оцінок. Використання цього міжнародного стандарту підтримує кожну організацію у прагненні стати більш соціально відповідальною, брати до уваги інтереси її членів, відповідати чинному законодавству та поважати міжнародні норми поведінки.

4.4 Підвищення стійкості роботи підприємств приладобудівної галузі в воєнний час

Підвищення стійкості роботи підприємств приладобудівної галузі в воєнний час є критично важливим завданням для забезпечення безпеки та ефективності воєнних операцій. Це означає здатність підприємств витримувати негативні впливи воєнних конфліктів та забезпечувати безперебійне функціонування в умовах збройних конфліктів, мобілізації та воєнного стану.

В останні роки поряд із заходами, що спрямовані на забезпечення охорони природного середовища, проводиться робота, яка передбачає не тільки підготовку до ліквідації НС, а й підвищення стійкості роботи об’єкта господарської діяльності (ОГД) у НС.

Необхідно підкреслити, що складність виробництва сучасного озброєння потребує широкої кооперації галузей виробництва. При цьому підвищується залежність підприємства одного від другого. Значні руйнування на ОГД і великі втрати серед населення можуть стати причиною різкого зменшення випуску сільськогосподарської і військової продукції, що в свою чергу знизить боєздатність Армії.

Тому, підвищення стійкості роботи ОГД і забезпечення стійких виробничих зв’язків в умовах НС (війни) є особливо важливим питанням і складає одне із головних завдань ЦЗ [57].

Під стійкістю роботи об’єкта розуміють здатність його під час НС випускати продукцію у запланованому обсязі і номенклатурі, а у випадках впливу на об’єкт вражаючих факторів – в мінімально короткі терміни відновити своє виробництво.

Для об’єктів, що не виробляють матеріальні цінності (транспорт, зв’язок), під стійкістю їх роботи розуміють здатність виконувати свої функції у НС.

На стійкість роботи промислового об’єкта в умовах НС впливають такі фактори:

– надійність захисту робітників і службовців від сучасних засобів ураження;

– здатність інженерно-технічного комплексу (ІТК) протистояти дії вражаючих факторів ядерної зброї;

– надійність систем постачання всім необхідним для виготовлення запланованої продукції (паливо, вода, газ, електроенергія та інше);

– стійкість управління виробництвом;

– готовність підприємства до проведення робіт по відновленню порушеного виробництва.

Основні шляхи підвищення стійкості роботи промислових об’єктів у надзвичайних умовах мирного і воєнного часів:

– забезпечення надійного захисту робітників і службовців від ЗМУ (засобів масового ураження);

– захист виробничих фондів від вражаючих факторів ЗМУ, в тому числі і від вторинних;

– підвищення надійності і оперативності управлінням виробництвом і ЦЗ;

– забезпечення стійкості постачання підприємства електроенергією, газом, водою та інше;

– підготовка об’єкта до проведення відновлювальних робіт.

Підвищення стійкості роботи промислових підприємств в умовах НС мирного і воєнного часів досягається завчасним проведенням комплексу інженерно-технічних, технологічних і організаційних заходів.

Інженерно-технічні заходи (ІТЗ) включають комплекс робіт по підвищенню міцності і надійності будинків, споруд комунально-енергетичних систем, матеріально-технічних запасів.

Технологічні заходи спрямовані на підвищення стійкості виробництва шляхом заміни існуючого технологічного режиму роботи на такий, що виключає можливість виникнення вторинних вражаючих факторів.

Організаційні заходи передбачають розробку і планування дій в умовах НС керівного складу об’єкту, штабу, служб та невоєнізованих формувань ЦЗ по захисту робітників і службовців, проведення рятувальних робіт та відновлення порушеного виробництва.

Підвищення стійкості об'єкта досягається посиленням найбільш слабких (вражаючих) елементів і ділянок об'єкта. Для цього на кожному ОГД завчасно на основі досліджень планують і проводять відповідні організаційні й інженерно-технічні заходи.

Досягнення науки і техніки дозволяють реалізувати такі рішення, при яких підприємство буде стійке до впливу дуже значних надлишкових тисків, однак це пов'язано з великими витратами засобів і матеріалів і може бути виправдано лише при захисті унікальних, особливо важливих елементів об'єкта. Заходи будуть економічно обґрунтовані, якщо вони максимально узгоджені із завданнями, які розв'язуються в мирний час для забезпечення безаварійної роботи, поліпшення умов праці, удосконалювання виробничого процесу. Тому підвищення характеристик міцності проводять, якщо:

– окремі особливо важливі будинки і спорудження значно слабші за інші і їхню міцність доцільно довести до прийнятої для даного підприємства межі стійкості;

– необхідно зберегти деякі важливі ділянки (цехи), які можуть самостійно функціонувати при виході з ладу інших і забезпечать випуск особливо цінної продукції.

Особливо велике значення має розробка інженерно-технічних заходів при новому будівництві, бо у процесі проектування, як відзначалося раніше, у багатьох випадках можна домогтися логічного поєднання загальних

інженерних рішень із захисними заходами ЦЗ, що знизить витрати на їх реалізацію.

На існуючих об'єктах заходи щодо підвищення стійкості доцільно проводити в процесі реконструкції чи виконання інших ремонтно-будівельних робіт.

Підвищення стійкості роботи промислових об'єктів передбачає:

– захист робітників та службовців у надзвичайних ситуаціях мирного і воєнного часу;

– підвищення міцності і стійкості найважливіших елементів і удосконалювання технологічного процесу;

– підвищення стійкості матеріально-технічного постачання;

– підвищення стійкості управління об'єктом;

– розробку заходів щодо зменшення імовірності виникнення вторинних факторів ураження і збитків від них;

– підготовку до відновлення виробництва після ураження об'єкта.

Особлива увага повинна бути приділена забезпеченню укриттям всіх працюючих у захисних спорудженнях. З цією метою розробляється план нагромадження і будівництва необхідної кількості захисних споруджень; у випадку недостачі сховищ, які відповідають сучасним вимогам, у ньому передбачається укриття робітників та службовців у швидкостворюваних сховищах.

При проектуванні і будівництві нових цехів підвищення стійкості може бути досягнуто застосуванням для несучих конструкцій високоміцних і легких матеріалів (легованих сталей, алюмінієвих сплавів).

Заходи щодо підвищення стійкості технологічного і верстатного устаткування повинні бути спрямовані на забезпечення його збереження для випуску продукції після надзвичайної ситуації. Однак підвищити стійкість устаткування можна, підсилюючи його найбільш слабкі елементи і створюючи

запаси цих елементів, окремих вузлів і деталей, матеріалів та інструментів для ремонту і відновлення пошкоджень.

Важке устаткування розміщують, по можливості, на нижніх поверхах виробничих будівель. Велике значення має міцне закріплення на фундаментах верстатів і установок, які мають велику висоту і малу площу опори; використання розтяжок і додаткових опор підвищить їх стійкість до перекидання. Прилади бажано встановлювати на закріплених підставках, тумбах, столах. Особливо цінне й унікальне устаткування потрібно розміщувати в заглиблених підземних чи спеціально побудованих приміщеннях підвищеної міцності і на випадок виникнення надзвичайних ситуацій розробити спеціальні індивідуальні енергогасильні пристрої.

При удосконалюванні технологічних процесів виробництва слід вживати і заходи для підвищення їх стійкості, пам'ятаючи, що найбільш важливі умови надійності – стійкість системи управління і безперебійність забезпечення усіма видами енергопостачання. У випадку виходу з ладу автоматичних систем управління повинен бути передбачений перехід на ручне управління процесом у цілому чи окремими його ділянками.

На випадок значних руйнувань повинна бути передбачена заміна складних технологічних процесів більш простими з використанням найбільш стійких типів устаткування і контрольно-вимірювальних приладів, які збереглись. Необхідно заздалегідь розробити можливі зміни в технології з метою заміни дефіцитних матеріалів, деталей і сировини на більш доступні.

На всіх об'єктах розробляються способи безаварійної зупинки виробництва за сигналом "Повітряна тривога" ("ПТ"). У кожній зміні призначаються люди, які повинні відключати джерела живлення і технологічні установки. Якщо за умовами технологічного процесу зупинити окремі ділянки виробництва, агрегати, печі і т. д. не можна, їх переводять на знижений режим роботи; ті, що спостерігають за безупинною роботою цих елементів, повинні

бути забезпечені індивідуальними укриттями, спорудженими в безпосередній близькості від робочого місця.

Одним із найважливіших заходів по забезпеченню сталого, безперервного на всіх етапах управління у надзвичайних ситуаціях є розподіл всього персоналу об'єкта на дві групи: працююча зміна (перебуває на об'єкті) і відпочиваюча (перебуває у заміській зоні або по дорозі між заміською зоною та об'єктом). До того ж створюються дві-три групи управління, які, крім керівництва виробництвом, повинні бути готові будь-якої миті взяти на себе організацію і керівництво проведенням рятувальних та ремонтних робіт [58].

Для забезпечення надійного управління діяльністю об'єкта у надзвичайних ситуаціях мирного і воєнного часу в одному із сховищ обладнується пункт управління. Диспетчерські пункти і радіовузли розміщують по можливості у найміцніших спорудах і підвальних приміщеннях. Повітряні лінії зв'язку до найважливіших виробничих ділянок переводять на підземно-кабельні.

Стабільно працююче підприємство повинно бути здатним безперебійно випускати продукцію за рахунок наявних запасів до відновлення зв'язків з поставок або до одержання необхідного від нових постачальників.

1. Основні аспекти, що варто враховувати для підвищення стійкості роботи підприємств приладобудівної галузі в воєнний час, включають:

2. Резервування та розподілення виробничих потужностей: Розподілення виробничих потужностей між різними регіонами або підприємствами допомагає зменшити ризик втрати виробництва в разі атаки на один з об'єктів. Резервування критичного обладнання та запасних частин також гарантує продовження виробництва в разі пошкоджень або втрати обладнання.

3. Забезпечення криптографічної безпеки та захисту інформації: Важливо захищати конфіденційні дані, інтелектуальну власність та комунікації від несанкціонованого доступу та кібератак. Використання шифрування,

аутентифікації та захисту мережевих систем допомагає забезпечити цю безпеку.

4. Диверсифікація постачальників та поставок: Залежність від одного постачальника або ринку може бути ризиком в разі обмежень поставок через воєнний конфлікт. Диверсифікація постачальників, альтернативних ринків та поставок допомагає знизити цей ризик та забезпечити продовження виробництва.

5. Планування та гнучкість: Розробка планів для надзвичайних ситуацій, а також гнучкість у виробництві та поставках можуть допомогти швидко адаптуватися до змінних умов воєнного часу. Це може включати швидку зміну виробництва, резервування робочої сили та планування доставок.

6. Співпраця з військовими та урядовими організаціями: Важливо підтримувати співпрацю та комунікацію з військовими та урядовими організаціями для обміну інформацією, встановлення пріоритетів та отримання допомоги в разі надзвичайних ситуацій.

Врахування цих аспектів допоможе підприємствам приладобудівної галузі забезпечити стійкість своєї роботи в умовах воєнних конфліктів, зберегти виробничі потужності та забезпечити безперебійність постачання необхідних продуктів та послуг.

4.5 Висновок до четвертого розділу

У розділі «Охорона праці та безпека в надзвичайних ситуаціях» проаналізовано надзвичайні ситуації воєнного характеру, досліджено міжнародні норми соціальної відповідальності за стандартом SA 8000, описано підвищення стійкості роботи підприємства приладобудівної галузі в воєнний час та визначено як проводити оцінку стійкості роботи економіки до впливу поражаючих факторів ядерної зброї.

ВИСНОВКИ

У даній кваліфікаційній роботі було досліджено велику кількість різних підходів до роботи з потоковими даними, а саме потоковим відеорядом. Розроблено програмне рішення стійке до навантажень та оптимізоване для роботи з великою кількістю користувачів одночасно. Основа програмного рішення складається з передових технологій машинного навчання та комп'ютерного бачення. Проведено інтеграцію сервісу із хмарним рішенням Amazon Web Services.

В першому розділі кваліфікаційної роботи освітнього рівня «Магістр»:

– Проаналізовано використання хмарної архітектури для обробки зображень.

– Ознайомлено з доступними методами опрацювання потоку даних.

– Висвітлено можливі джерела безперервного потоку зображень.

– Досліджено підхід інженерії даних для зберігання і опрацювання великих об'ємів потокових даних.

– Поверхнево розглянуто глибоке навчання та комп'ютерне бачення для обробки зображень.

В другому розділі кваліфікаційної роботи:

– Проведено опис продукту та його вимог.

– Проаналізовано хмарні рішення Amazon Web Services та Google Cloud Platform, проведено їхнє порівняння, та було обрано AWS як основне хмарне рішення для розгортання продукту.

– Досліджено популярні системи передачі та опрацювання потокових даних, обрано AWS Kinesis головним сервісом для опрацювання відеоряду.

– Описано методи машинного аналізу зображень, визначено моделі для розпізнавання об'єктів, розпізнавання лиць, оцінки пози рук.

В третьому розділі кваліфікаційної роботи:

– Проведено пошук актантів та варіантів використання продукту.

– Спроектовано архітектуру продукту.

– Сформовано опис програмних рішень сервера та веб-клієнта.

– Описано інтерфейс веб-клієнта

– Розгорнуто опрацювання безперервного потоку даних та встановлено розроблену систему у хмарному рішенні AWS.

У розділі «Охорона праці та безпека в надзвичайних ситуаціях» проаналізовано надзвичайні ситуації воєнного характеру, досліджено міжнародні норми соціальної відповідальності за стандартом SA 8000, описано підвищення стійкості роботи підприємства приладобудівної галузі в воєнний час та визначено як проводити оцінку стійкості роботи економіки до впливу поражаючих факторів ядерної зброї.

ПЕРЕЛІК ДЖЕРЕЛ

1 SSD: Single Shot MultiBox Detector [Електронний ресурс] – Режим доступу до ресурсу: https://arxiv.org/pdf/1512.02325v5.pdf – Дата доступу: 14.04.2023.

2 YOLO vs SSD: Which One is a Superior Algorithm [Електронний ресурс] – Режим доступу до ресурсу: https://algoscale.com/blog/yolo-vs-ssd-which-one-is-a-superior-algorithm – Дата доступу: 14.04.2023.

3 YOLOv4: Optimal Speed and Accuracy of Object Detection [Науково-технічний ресурс] – Режим доступу до ресурсу: https://arxiv.org/abs/2004.10934 – Дата доступу: 14.04.2023. Дата публікації: 21.04.2020.

4 YOLOv5 Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.ultralytics.com/ – Дата доступу: 14.04.2023.

5 Хмарні обчислення – Вікіпедія [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Хмарні_обчислення – Дата доступу: 01.03.2023.

6 Характеристики хмарних обчислень за NIST [Електронний ресурс] – Режим доступу до ресурсу: https://timesofcloud.com/cloud-tutorial/characteristics-of-cloud-computing-as-per-nist – Дата доступу: 01.03.2023.

7 Характеристики хмарних обчислень [Електронний ресурс] – Режим доступу до ресурсу: https://coggle.it/diagram/WNtfITPg0gABaMkZ/t/характеристики-хмарних-обчислень – Дата доступу: 01.03.2023.

8 Хмарна піраміда: IaaS, PaaS і SaaS [Електронний ресурс] – Режим доступу до ресурсу: https://gigacloud.ua/blog/navchannja/hmarna-piramida-iaas-paas-i-saas – Дата доступу: 01.03.2023.

9 Чотири найкращі хмарні моделі розгортання [Електронний ресурс] – Режим доступу до ресурсу: https://www.sam-solutions.com/blog/four-best-cloud-deployment-models-you-need-to-know– Дата доступу: 01.03.2023.

10 Streaming Video over the Internet: Approaches and Directions [Електронний ресурс] – Режим доступу до ресурсу: https://web.engr.oregonstate.edu/~thinhq/teaching/ece599/papers/wu01streaming.pdf – Дата доступу: 01.03.2023.

11 Cloud-Based Video Streaming Services: A Survey [Науково-технічний ресурс] – Режим доступу до ресурсу: https://arxiv.org/abs/2011.14976 – Дата доступу: 14.04.2023. Дата публікації: 30.11.2020.

12 Correcting the IoT History [Електронний ресурс] – Режим доступу до ресурсу: http://www.chetansharma.com/correcting-the-iot-history – Дата доступу: 01.03.2023.

13 Smart networks for control [Електронний ресурс] – Режим доступу до ресурсу: https://ieeexplore.ieee.org/document/284793 – Дата доступу: 01.03.2022.

14 The smart home: a glossary guide for the perplexed [Електронний ресурс] – Режим доступу до ресурсу: https://www.t3.com/features/the-smart-home-guide – Дата доступу: 01.03.2023.

15 Machine Learning with Applications: Volume 6 [Науково-технічний ресурс] – Режим доступу до ресурсу: https://doi.org/10.1016/j.mlwa.2021.100134 – Дата доступу: 14.04.2023. Дата публікації: 15.12.2021.

16 Microsoft COCO: Common Objects in Context [Електронний ресурс] – Режим доступу до ресурсу: https://arxiv.org/abs/1405.0312 – Дата доступу: 01.03.2023.

17 Face Recognition Pipeline Clearly Explained [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/backprop-labs/face-recognition-pipeline-clearly-explained-f57fc0082750 – Дата доступу: 01.03.2023.

18 DeepFace: Closing the Gap to Human-Level Performance in Face Verification [Електронний ресурс] – Режим доступу до ресурсу: https://research.facebook.com/file/266870805034649/deepface-closing-the-gap-to-human-level-performance-in-face-verification.pdf – Дата доступу: 01.03.2022.

19 OpenFace: A general-purpose face recognition library with mobile applications [Електронний ресурс] – Режим доступу до ресурсу: http://reports-archive.adm.cs.cmu.edu/anon/anon/2016/CMU-CS-16-118.pdf – Дата доступу: 01.03.2023.

20 Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments [Електронний ресурс] – Режим доступу до ресурсу: http://vis-www.cs.umass.edu/lfw/lfw.pdf – Дата доступу: 01.03.2023.

21 MediaPipe Hands: On-device Real-time Hand Tracking [Науково-технічний ресурс] – Режим доступу до ресурсу: https://arxiv.org/abs/2006.10214– Дата доступу: 14.04.2023. Дата публікації: 18.06.2020.

22 AWR: Adaptive Weighting Regression for 3D Hand Pose Estimation [Науково-технічний ресурс] – Режим доступу до ресурсу: https://arxiv.org/abs/2007.09590 – Дата доступу: 14.04.2023. Дата публікації: 19.07.2020.

23 Детально про черги повідомлень (Message queue) [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Черга_повідомлень – Дата доступу: 05.03.2023.

24 Обробка потокової інформації [Електронний ресурс] – Режим доступу до ресурсу: http://apeps.kpi.ua/downloads/обробка%потоко вої%інформації.pdf – Дата доступу: 14.04.2023.

25 Cloud Solutions, cloud architectures [Електронний ресурс] – Режим доступу до ресурсу: https://www.suse.com/suse-defines/definition/cloud-solutions/ – Дата доступу: 05.01.2023.

26 What is the internet of things [Електронний ресурс] – Режим доступу до ресурсу: https://www.techtarget.com/iotagenda/definition/Internet-of-Things-IoT – Дата доступу: 10.01.2023.

27 Internet of things [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Internet_of_things – Дата доступу: 02.03.2023.

28 Інженерія даних (Data Enginering) [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Data_engineering – Дата доступу: 23.03.2023.

29 What Is a Data Engineer? [Електронний ресурс] – Режим доступу до ресурсу: https://www.coursera.org/articles/what-does-a-data-engineer-do-and-how-do-i-become-one – Дата доступу: 01.01.2023.

30 Data Engineering Concepts, Processes, and Tools [Електронний ресурс] – Режим доступу до ресурсу: https://www.altexsoft.com/blog/datascience/what-is-data-engineering-explaining-data-pipeline-data-warehouse-and-data-engineer-role/ – Дата доступу: 06.03.2023.

31 How to Become a Data Engineer [Електронний ресурс] – Режим доступу до ресурсу: https://www.datacamp.com/blog/how-to-become-a-data-engineer – Дата доступу: 13.01.2023.

32 Машинне навчання (Machine learning) [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Machine_learning – Дата доступу: 13.01.2023.

33 Глибоке начання (Deep learning) [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Deep_learning – Дата доступу: 15.02.2023.

34 What Is Deep Learning? [Електронний ресурс] – Режим доступу до ресурсу: https://www.mathworks.com/discovery/deep-learning.html – Дата доступу: 11.02.2023.

35 Хмаро орієнтовані платформи, засоби і послуги [Науково-технічний ресурс] – Режим доступу до ресурсу: https://lib.iitta.gov.ua/716589/1/012-024_Damniskaya.pdf – Дата доступу: 17.04.2023. Дата публікації: 01.02.2019.

36 Amazon Web Services [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Amazon_Web_Services – Дата доступу: 11.01.2023.

37 Google Cloud Platform [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Google_Cloud_Platform – Дата доступу: 11.04.2023.

38 Google Cloud vs AWS [Електронний ресурс] – Режим доступу до ресурсу: https://www.guru99.com/google-cloud-vs-aws.html. – Дата доступу: 02.05.2023.

39 Apache kafka [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Apache_Kafka – Дата доступу: 14.04.2023.

40 Kafka introduction [Електронний ресурс] – Режим доступу до ресурсу: https://kafka.apache.org/intro – Дата доступу: 14.02.2023.

41 What is a Kafka Topic [Електронний ресурс] – Режим доступу до ресурсу: https://hevodata.com/learn/kafka-topic/ – Дата доступу: 14.02.2023.

42 Реплікація Кафка (Kafka Replication) [Електронний ресурс] – Режим доступу до ресурсу: https://hevodata.com/learn/kafka-replication/. – Дата доступу: 15.02.2023.

43 What does In-Sync Replicas in Apache Kafka Really Mean? [Електронний ресурс] – Режим доступу до ресурсу: https://www.cloudkarafka.com/blog/what-does-in-sync-in-apache-kafka-really-mean.html – Дата доступу: 13.01.2023.

44 How ISR work in Apache Kafka [Електронний ресурс] – Режим доступу до ресурсу: https://www.conduktor.io/blog/how-replication-and-isr-work-in-kafka/ – Дата доступу: 14.01.2023.

45 What is a consumer group in Kafka? [Електронний ресурс] – Режим доступу до ресурсу: https://dev.to/de_maric/what-is-a-consumer-group-in-kafka-49il – Дата доступу: 21.02.2023.

46 Understanding Kafka Consumer Offset [Електронний ресурс] – Режим доступу до ресурсу: https://dattell.com/data-architecture-blog/understanding-kafka-consumer-offset/ – Дата доступу: 27.04.2023.

47 Apache ZooKeeper service [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Apache_ZooKeeper – Дата доступу: 27.04.2023.

48 FIFO, Exactly-Once, and Other Costs [Електронний ресурс] – Режим доступу до ресурсу: https://dzone.com/articles/fifo-exactly-once-and-other-costs – Дата доступу: 28.04.2023.

49 Amazon Managed Streaming for Apache Kafka (MSK) [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/msk/?nc1=h_ls – Дата доступу: 11.04.2023.

50 Amazon Kinesis [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/kinesis/ – Дата доступу: 05.05.2023.

51 What is AWS Kinesis (Amazon Kinesis Data Streams)? [Електронний ресурс] – Режим доступу до ресурсу: https://k21academy.com/amazon-web-services/amazon-kinesis/ – Дата доступу: 14.01.2023.

52 Міжнародні стандарти впровадження принципів соціальної відповідальності [Електронний ресурс] – Режим доступу до ресурсу: https://dut.edu.ua/uploads/p_1010_91031876.pdf – Дата доступу: 05.05.2023.

53 Підвищення стійкості об’єктів господарської діяльності в умовах НС [Електронний ресурс] – Режим доступу до ресурсу: http://opcb.kpi.ua/wp-content/uploads/2014/09/Лекц_я-4.pdf – Дата доступу: 04.05.2023.

54 Дії населення в умовах надзвичайних ситуацій воєнного характеру [Електронний ресурс] – Режим доступу до ресурсу: https://dsns.gov.ua/uk/abetka-bezpeki/diyi-naselennya-v-umovax-nadzvicainix-situacii-vojennogo-xarakteru – Дата доступу: 05.05.2023.

55 Надзвичайні ситуації та їх класифікація. Реферат [Електронний ресурс] – Режим доступу до ресурсу: https://osvita.ua/vnz/reports/bjd/22895/ – Дата доступу: 05.05.2023.

56 Класифікатор надзвичайних ситуацій [Електронний ресурс] – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/va457609-10#Text – Дата доступу: 05.05.2023.

57 Стійкість роботи промислових об’єктів у надзвичайних ситуаціях [Електронний ресурс] – Режим доступу до ресурсу: https://org2.knuba.edu.ua/pluginfile.php/114575/mod_resource/content/1/Лекція_№4.pdf – Дата доступу: 05.05.2023.

58 Тема 14. Підвищення стійкості роботи об'єкта господарювання [Електронний ресурс] – Режим доступу до ресурсу: https://studfile.net/preview/5563647/ – Дата доступу: 05.05.2023.

ДОДАТКИ

Додаток А

Тези на наукову конференцію XI Міжнародної науково-практичної конференції молодих учених та студентів «Актуальні задачі сучасних технологій»

УДК 004.6

В. Лісовський, А. Зелінський, О. Сороківський
Тернопільський національний технічний університет імені Івана Пулюя, Україна

АНАЛІЗ ЗАДАЧ МАШИННОГО АНАЛІЗУ ЗОБРАЖЕННЯ ТА СУЧАСНИХ МЕТОДІВ ЇХ ВИРІШЕННЯ

V. Lisovskyi, A. Zeliniskyi, O. Sorokivskyi

ANALYSIS OF MACHINE IMAGE ANALYSIS TASKS AND THEIR MODERN SOULTIONS

З кожним роком машинний аналіз покриває все більшу кількість сфер та аспектів буденного життя людей. Одним з значних проривів у сфері інформатики стала розробка моделей машинного навчання та штучного інтелекту, які дозволяють автоматизувати велику кількість задач. Комп'ютерне бачення, яке полягає у аналізі візуальної інформації машиною засобами штучного інтелекту, є яскравим прикладом сучасних методів аналізу зображень.

Задачі машинний аналізу зображень поділяються на наступні категорії:

- обробка зображень;
- розпізнавання об'єктів (також відома як класифікація об'єктів);
- сегментація зображень;
- оцінка пози.

Обробка зображень, в порівнянні з іншими, є простішою задачею аналізу, оскільки вона полягає в проведенні статистичного аналізу характеристик зображення, та застосуванні простих алгоритмічних змін на основі отриманої інформації. До цієї категорії належать такі техніки як вирівнювання гістограм тональностей (використовуючи гістограми розподілу зображення), зниження шуму (використовуючи або морфологічні фільтри, або лінійні-нелінійні

фільтри, або нейронні мережі), фільтри зглаження, фільтри різкості та фільтри виявлення контурів. Методи з категорії обробки зображень переважно використовуються як початковий крок з покращення зображення в підготовці його до більш глибокого машинного аналізу.

Розпізнавання об’єктів – це складна система, яка полягає у зборі сукупності статистичних та візуальних характеристик зображення та їх подальшому аналізі на основі схожості набору характеристик до певних шаблонів (цільових об’єктів). Сучасним методом у розпізнаванні об’єктів є використання глибинних нейронних мереж, що використовують згорткові шари (фільтри). Їх використовують у таких задачах як:

- класифікація зображень, де проводиться огляд зображення в цілому та його класифікація до цільових класів (сучасні підходи до вирішення задачі – архітектури ResNet, EfficientNet, MobileNet, Inception та ін.);
- розпізнавання об’єктів, де проводить аналіз елементів зображення, знаходження окремих складових (цільових об’єктів) та їх класифікація до цільових класів (сучасні підходи до вирішення задачі – архітектури SSD, R-CNN, RetinaNet, YOLO та ін.).

Сегментація зображень полягає у розділенні зображення на полігональні сегменти, кожен з яких представляє собою певний об’єкт, який візуально відрізняється від інших сегментів. До методів, які проводять подібний аналіз зображень, належать пороговий аналіз, кластерний аналіз, контурний аналіз, аналіз гістограм розподілу та використання глибнних нейронних мереж. Сегментація поділяється на три групи задач – семантична (кожен піксель класифікується до цільового класу), об’єктна (instance) (кожен піксель класифікується до цільового класу з розподілом на різні об’єкти) та комбінована (panoptic) (включає у себе особливості обох груп). До нейронних мереж, що розв’язують першу групу здача, входять U-Net, FCN, SegNet, DeepLab, EfficientDet та інші. До нейронних мереж другої групи входять Mask R-CNN, HTC, ConInst та інші.

Оцінка пози – це задача, направлена на аналіз позиції та положення об’єкта у просторі. Переважно це виконується завдяки розпізнаванню складових (орієнтирів) об’єкта на зображенні, після чого проводиться їх об’єднання у цільовий об’єкт, для якого вираховується агрегована позиція та положення у просторі. До моделей, які вирішують цю проблему, належать OpenPose, MoveNet, PoseNet, AlphaPose та інші.

Отже, великий спектр задач пов’язаних з аналізом зображення сьогодні доволі успішно розв’язуються машиною з використанням глибинних нейронних мереж. Кожна категорія задач має свій широкий набір методів для розв’язку, які мають свої недоліки, проте значно полегшують процес автоматизації виконання задач та знижують постійну потребу у людському нагляді.

Перелік використаної літератури:

1. Yağmur Çiğdem Aktaş. A Comprehensive Guide to Image Processing: Fundamentals, 2021. https://towardsdatascience.com/image-processing-4391c5bcef78

2. Neetu Rani. Image Processing Techniques: A Review, 2017. https://www.researchgate.net/publication/345364858_Image_Processing_Techniques_A_Review

3. Salwa Khalid Abdulateef, Mohanad Dawood Salman. A Comprehensive Review of Image Segmentation Techniques, 2021. https://www.researchgate.net/publication/354846947_A_Comprehensive_Review_of_Image_Segmentation_Techniques

4. Rohit Josyula, Sarah Ostadabbas. A Review on Human Pose Estimation, 2021. https://arxiv.org/abs/2110.06877

УДК 004.6

А. Зелінський, В. Лісовський

Тернопільський національний технічний університет імені Івана Пулюя, Україна

ПРОЕКТУВАННЯ ХМАРНОГО РІШЕННЯ ДЛЯ АНАЛІЗУ ПОТОКОВИХ ДАНИХ НА ОСНОВІ AWS KINESIS

A. Zeliniskyi, V. Lisovskyi

DESIGN OF A CLOUD SOLUTION FOR ANALYSIS OF STREAMING DATA BASED ON AWS KINESIS

У теперішній інформаційний час, щодня генерується більше ніж 1000 петабайт даних починаючи від мобільних пристроїв і закінчуючи розумним холодильником. Велика частка цих даних генеруються неперервним потоком. Потокові дані - це дані, які постійно генеруються різними джерелами. Такі дані слід обробляти поступово за допомогою методів потокової обробки без доступу до всіх даних. Крім того, слід враховувати, що в даних може статися дрейф концепції, що означає, що властивості потоку можуть змінюватися з часом. Зазвичай потік даних використовується в контексті великих даних, де вони генеруються багатьма різними джерелами з високою швидкістю.

Потік даних також можна пояснити як технологію, яка використовується для доставки вмісту на пристрої через Інтернет, і вона дозволяє користувачам отримати доступ до вмісту негайно, а не чекати, поки його завантажать. Великі дані змушують багато організацій зосереджуватися на витратах на зберігання, що викликає інтерес до озер даних і потоків даних. Озеро даних відноситься до зберігання великої кількості неструктурованих даних і є корисним через збільшення великих даних, оскільки їх можна зберігати таким чином, щоб фірми могли зануритися в озеро даних і отримати те, що їм потрібно в даний

момент. Тоді як потік даних може виконувати аналіз потокових даних у реальному часі, і він відрізняється від озер даних швидкістю та безперервним характером аналізу без необхідності попереднього зберігання даних [1].

Для обробки великої кількості потокових даних без розгортання власних серверів використано сервіс Kinesis на хмарній платформі Amazon Web Services.

Amazon Kinesis - хмарний сервіс для обробки великої кількості розподілених потоків даних у режимі реального часу. Сервіс входить в інфраструктуру Amazon Web Services. Дозволяє розробникам витягувати будь-яку кількість даних з будь-якої кількості джерел, збільшуючи або зменшуючи кількість джерел при необхідності. Має деяку схожість по функціоналу з Apache Kafka [2].

До сервісів AWS Kinesis входить:

1. Amazon Kinesis Data Streams
2. Amazon Kinesis Video Streams
3. Amazon Kinesis Data Firehose
4. Amazon Kinesis Data Analytics

Kinesis включає в собі набір сервісів які можна поєднувати між собою та з іншими сервісами утворюючи послідовність опрацювання даних починаючи від вхідного потоку і закінчуючи озером даних, кінцевим користувачем, структурованим сховищем даних, або все одразу [3] (див. рис. 1).

Рисунок 1 – Приклад використання сервісів Kinesis

Отже, використовуючи сервіси Amazon Kinesis, можна досягти ефективного опрацювання потоку даних, його масштабованість та стабільність оскільки частину роботи по підтримці на себе бере хмарне рішення.

Перелік використаної літератури:

1.What is streaming data? https://aws.amazon.com/streaming-data/

2.Ultimate Guide to Data Streaming with AWS Kinesis. https://www.udemy.com/course/ultimate-guide-to-data-streaming-with-aws-kinesis

3.Amazon Kinesis. Easily collect, process, and analyze video and data streams in real time, 2016. https://aws.amazon.com/kinesis/

Додаток Б

Програмний код сервера і веб-клієнта

Лістинг файлу main.py

```
from collections import Counter
from flask import Flask, render_template, request, flash, url_for,
redirect, g, jsonify, Response
from uuid import uuid1
import datetime
import auth
from db import db, init_app
import cv2
import socketio
import eventlet.wsgi
from flask_cors import CORS
from AIModule import Image
from engineio.payload import Payload
from time import time
from models import SavedPerson

Payload.max_decode_packets = 500
QUEUE_CLEAR_SECONDS = 5
QUEUE_SIZE = 12
QUEUE_CONFIDENCE = 6


app = Flask(__name__, static_url_path='/static')
CORS(app)
seo = socketio.Server()
app.wsgi_app = socketio.WSGIApp(seo, app.wsgi_app)
app.register_blueprint(auth.bp)
app.config.from_pyfile("server_conf.cfg")
app.config['SQLALCHEMY_DATABASE_URI']                                    =
f"{app.config['DRIVER']}://{app.config['USERNAME']}:{app.config['P
ASSWORD']}" \

f"@{app.config['HOST']}:{app.config['PORT']}/{app.config['DB_NAME'
]}"
init_app(app)
db.init_app(app)
app.jinja_env.add_extension('jinja2.ext.loopcontrols')

app.add_url_rule('/login', endpoint='auth.login')
app.add_url_rule('/auth', endpoint='auth.login')

NOT_AVAILABLE = cv2.imread("static/media/unavailable-image.jpg")
NO_FRAME = Image.encode_image(NOT_AVAILABLE)

USERS = {}
```

```
@auth.login_required
@app.route('/', methods=['POST', 'GET'])
def index():
    return view_sources()

@auth.login_required
@app.route('/sources', methods=['GET'])
def view_sources():
    if g.user.id not in USERS:
        USERS[g.user.id]  =  {"SOURCES":  {},  "VISIBILITY":  {},
"QUEUES": {}, "PERSONS": {}}
    user_info = USERS.get(g.user.id)
    return render_template('index.html', user_info=user_info)

@auth.login_required
@app.route('/add_source', methods=['POST'])
def add_source():
    user = USERS.get(g.user.id)
    info = dict(**request.form)
    sid = str(uuid1())
    info['date']                                                    =
datetime.datetime.now().isoformat(timespec='seconds')
    if info['type'] == 'stream' and not info['access'].strip():
        info['access'] = str(uuid1()).replace('-', '')
    user['SOURCES'][sid] = info
    flash("New source added", 'success')
    return redirect(url_for('view_sources'))

@auth.login_required
@app.route("/source/edit/<sid>", methods=('GET', 'POST'))
def source_edit(sid):
    user_info = USERS.get(g.user.id)
    source = user_info['SOURCES'][sid]
    if request.method == 'POST':
        for name, item in request.form.items():
            source[name] = item
        if    (source['type']    !=    request.form['type'])    and
(source['type'] == 'stream'):
            source['access'] = str(uuid1()).replace('-', '')
        flash("Data changed", "success")
        return redirect(url_for('index'))
    return render_template('source_edit.html', source=source)

@auth.login_required
@app.route("/source/remove/<sid>", methods=('GET', 'POST'))
def source_remove(sid):
    user_info = USERS.get(g.user.id)
    user_info["SOURCES"].pop(sid)
    flash("Source removed", "success")
    return redirect(url_for('index'))
```

```
@auth.login_required
@app.route("/source/view/<sid>", methods=('GET', 'POST'))
def view(sid):
    user_info = USERS.get(g.user.id)
    return          render_template('view.html',          sid=sid,
visibility=user_info['VISIBILITY'], uid=g.user.id,
                          source=user_info['SOURCES'][sid])

@auth.login_required
@app.route("/visibility/<sid>", methods=['POST'])
def change_visibility(sid):
    user_info = USERS.get(g.user.id)
    visibility = user_info['VISIBILITY']
    for name, val in visibility.items():
        visibility[name] = False if name not in request.form else
True
    return redirect(url_for('view', sid=sid))

@app.route("/video_feed/<uid>/<sid>", methods=['GET', 'POST'])
def video_feed(uid, sid):
    return     url_for('static',     filename='media/unavailable-
image.jpg')

def get_user_by_token(token):
    user = None
    for user_id, info in USERS.items():
        token_list   =   {data['access']   for   sid,   data   in
info['SOURCES'].items()}
        if token in token_list:
            user = user_id
            break
    return user

@seo.event
def connect(sid, environ, auth, **kwargs):
    print(sid)
    if 'HTTP_ACCESS_TOKEN' in environ :
        user_id = get_user_by_token(environ['HTTP_ACCESS_TOKEN'])
        if user_id is None:
            seo.disconnect(sid)
        else:
            seo.emit('send_data',                          {'token':
environ['HTTP_ACCESS_TOKEN']}, to=sid)

def denoise_queue(user_id, name, new_elem):
    person: SavedPerson = USERS[user_id]["PERSONS"][name]
    if (time() - person.queue_time) > QUEUE_CLEAR_SECONDS:
        person.queue_time = time()
        person.queue = [new_elem]
        return 0
    queue = person.queue
    queue.insert(0, new_elem)
```

```
    queue = queue[:QUEUE_SIZE]
    # print('Queue', queue)
    elem, cnt = Counter(queue).most_common(1)[0]
    person.queue_time = time()
    if cnt < QUEUE_CONFIDENCE:
        return 0
    person.queue = list()
    return elem

@seo.event
def send(sid, data):
    user_id = get_user_by_token(data['token'])
    if user_id is None:
        seo.disconnect(sid)
        return
    encoding_data = dict(face_encodings=list(), face_names=list())
    for person in USERS[user_id]['PERSONS'].values():
        encoding_data['face_encodings'].append(person.encoding)
        encoding_data['face_names'].append(person.name)
    image = Image(data['image'], USERS[user_id]['VISIBILITY'],
                  encoding_data)
    res_image, persons = image.inference()
    for person in persons:
        if person.name in USERS[user_id]["PERSONS"].keys():
            person.instructions                                    =
USERS[user_id]["PERSONS"][person.name].instructions
            current_count    =    sum(hand.count    for    hand    in
person.hands)
            common_elem   =   denoise_queue(user_id,   person.name,
current_count)
            person.total_count = common_elem
    seo.emit('response',    {'token':    data['token'],    'image':
res_image,
                             'persons': [person.dict() for person in
sorted(persons, key=lambda x: x.name)]},
             to=data['token'])
    seo.sleep(0)
    seo.emit('send_data', {'token': data['token']}, to=sid)

@seo.event
def join(sid, data):
    seo.enter_room(sid, data['token'])

@auth.login_required
@app.route('/edit/<sid>/person/<name>', methods=['POST'])
def edit_person(sid, name):
    form_data = request.form.to_dict()
    person: SavedPerson = USERS.get(g.user.id)['PERSONS'][name]
    if name != form_data['name'].strip():
        USERS.get(g.user.id)['PERSONS'].pop(name)
        name = form_data['name']
        person.name = name
```

```
        form_data.pop('name')
    for name, val in form_data.items():
        index = int(name.removeprefix('instr-'))
        if not val.strip():
            continue
        person.instructions[index] = val
    USERS.get(g.user.id)['PERSONS'][name] = person
    return redirect(url_for('view', sid=sid))

@auth.login_required
@app.route('/create/<sid>/person', methods=['POST'])
def create_person(sid):
    user_info = USERS.get(g.user.id)
    form_data = request.form.to_dict()
    person              =              SavedPerson(form_data['name'],
eval(form_data['face_encoding']))
    user_info["PERSONS"][person.name] = person
    return redirect(url_for('view', sid=sid))

@auth.login_required
@app.route('/delete/<sid>/person/<name>', methods=['GET'])
def delete_person(sid, name):
    user_info = USERS.get(g.user.id)['PERSONS']
    user_info.pop(name)
    return redirect(url_for('view', sid=sid))

if __name__ == '__main__':
     eventlet.wsgi.server(eventlet.listen(('0.0.0.0', 5000)), app)
```

Лістинг файлу AIModule.py

```
import time
from FingerTracker import LegacyFingerTracker, HandResult
import cv2
import base64
import numpy as np
from dataclasses import dataclass, field
from PersonDetector import PersonDetector
from FaceRecognizer import FaceRecognizer

finger_tracker = LegacyFingerTracker()
person_detector = PersonDetector()
face_recognizer = FaceRecognizer()

WRIST = finger_tracker.mp_hands.HandLandmark.WRIST
MIDDLE_FINGER_MCP                                                    =
finger_tracker.mp_hands.HandLandmark.MIDDLE_FINGER_MCP
HAND_LABELS = ("RIGHT", "LEFT")

@dataclass
class Person:
```

```
    boxes: np.array
    name: str = str()
    encoding: list = field(default_factory=list)
    hands: list[HandResult] = field(default_factory=list)
    instructions: dict[int, str] = field(default_factory=dict)
    total_count: int|None = None

    def __post_init__(self):
        self.xyxy = self.boxes[:4]

    def dict(self):
        return {"name": self.name, "encoding": self.encoding,
"instructions": self.instructions, "hands": [{"count": hand.count,
"label": hand.label, 'status': hand.status} for hand in
self.hands], "total_count": self.total_count}

class Image:
    hand_results: list[HandResult]
    persons: list[Person]

    def __init__(self, encoded_source, visibility, face_encodings)
-> None:
        self.encoded_source = encoded_source
        self.source = self.decode_image(encoded_source)
        self.original = self.source.copy()
        self.visibility = visibility
        self.face_encodings = face_encodings
        self.persons = list()

    @staticmethod
    def decode_image(encoded_image: np.ndarray) -> np.ndarray:
        jpg_original = base64.b64decode(encoded_image)
        jpg_as_np = np.frombuffer(jpg_original, dtype=np.uint8)
        img = cv2.imdecode(jpg_as_np, cv2.IMREAD_COLOR)
        return img

    @staticmethod
    def encode_image(image: np.ndarray) -> np.ndarray:
        encoded_img = cv2.imencode('.jpg', image)[1].tobytes()
        encoded_img = base64.encodebytes(encoded_img).decode("utf-
8")
        return encoded_img

    def inference(self) -> tuple[np.ndarray, list[Person]]:
        self._person_detect()
        persons_count = len(self.persons)
        if persons_count:
            self._recognize_faces()
            self._hand_detect()
            if self.visibility['hand_visibility']:
                self._draw_hand()
            if persons_count > 1 or len(self.hand_results) > 2:
```

```
                self._classify_hands()
            else:
                self.persons[0].hands = self.hand_results
        return self.encode_image(self.source), self.persons

    def _person_detect(self):
        detections = person_detector.detect(self.original)
        for detect in detections:
            person = Person(boxes=detect)
            self.persons.append(person)
            if self.visibility['person_visibility']:
                person_detector.draw_bbox(image=self.source,
bbox=person.xyxy)

    def _recognize_faces(self):
        unknown = 0
        for person in self.persons:
            x1, y1, x2, y2 = map(int, person.xyxy)
            cropped_img = self.original[y1:y2, x1:x2, :]
            face_encoding                                          =
face_recognizer.encode_face(cropped_img)
            if face_encoding.size == 0:
                person.name = 'No face detected'
                if self.visibility['name_visibility']:
                    face_recognizer.draw_name(self.source,
person.name, (x1, y1))
                continue
            face_index                                             =
face_recognizer.compare_faces(self.face_encodings['face_encodings'
], face_encoding)
            name  =  self.face_encodings['face_names'][face_index]
if face_index is not None else f'Unknown{" #"+str(unknown) if
unknown>0 else ""}'
            if face_index is None:
                unknown += 1
            if self.visibility['name_visibility']:
                face_recognizer.draw_name(self.source,  name,  (x1,
y1))
            person.name = name
            person.encoding = face_encoding.tolist()

    def _hand_detect(self):
        self.hand_results = finger_tracker.detect(self.original)

    def _draw_hand(self):
        self.source                                                =
finger_tracker.draw_landmarks_on_image(self.source,
self.hand_results)

    def _classify_hands(self):
        if not self.hand_results:
            return
```

```
        height, width, _ = self.source.shape
        for person in self.persons:
            if    person.name.startswith(('Unknown',    'No    face
detected')):
                continue
            x1, y1, x2, y2 = person.xyxy
            edges = [(x1, y1), (x2, y1), (x2, y2), (x1, y2)]
            sides  =  [(np.array(edges[i]),  np.array(edges[(i+1)%
len(edges)])) for i in range(len(edges))]
            for label in HAND_LABELS:
                hands = [hand for hand in self.hand_results if
hand.label == label]
                assigned = None
                distances = list()
                for hand in hands:
                    wrist_coord = hand.landmarks.landmark[WRIST]
                    x,       y       =       wrist_coord.x*width,
wrist_coord.y*height
                    if (x1 <= x and x <= x2) and (y1 <= y and y <=
y2):
                        assigned = hand
                        self.hand_results.remove(hand)
                        break
                    wrist_point = np.array([x, y])
                    distances.append(min((np.cross(p2-
p1,wrist_point-p1)/np.linalg.norm(p2-p1) for p1, p2 in sides)))
                if distances and assigned is None:
                    hand = hands[np.argmin(distances)]
                    assigned = hand
                    self.hand_results.remove(hand)
                if assigned is not None:
                    person.hands.append(assigned)
```

Лістинг файлу FaceRecognizer.py

```
import numpy as np
import cv2
import face_recognition
from PIL import Image

class FaceRecognizer:
    @staticmethod
    def encode_face(image) -> np.ndarray:
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        res = face_recognition.face_encodings(image)
        return res[0] if res else np.array([])

    @staticmethod
    def encode_face_from_file(path):
        image = face_recognition.load_image_file(path)
        return face_recognition.face_encodings(image)
```

```
    @staticmethod
    def      compare_faces(known_encodings,      unknown_encoding,
tolerance=0.6):
        if not isinstance(known_encodings, np.ndarray):
            known_encodings = np.array(known_encodings)

        best_index = None
        matches  =  face_recognition.compare_faces(known_encodings,
unknown_encoding, tolerance)
        if True in matches:
            best_index = matches.index(True)

        return best_index

    @staticmethod
    def      draw_name(image,      name,      left_corner_coords,
color=(255,255,255)):
        return    cv2.putText(image,    name,    left_corner_coords,
cv2.FONT_HERSHEY_DUPLEX, 1, color 1, cv2.LINE_AA)
```

Лістинг файлу FingerTracker.py

```
import cv2
import mediapipe as mp
import numpy as np
from dataclasses import dataclass, field

MARGIN = 10  # pixels
FONT_SIZE = 1
FONT_THICKNESS = 1
HANDEDNESS_TEXT_COLOR = (88, 205, 54)  # vibrant green
mp_drawing = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles
finger_names = ['THUMB', 'INDEX', 'MIDDLE', 'RING', 'PINKY']

def gen_finger_status():
    return {name: False for name in finger_names}

@dataclass
class HandResult:
    label: str = field(default_factory=str)
    status:             dict[str,             bool]             =
field(default_factory=gen_finger_status)
    count: int = field(default=0)
    landmarks: list = field(default_factory=list)

class LegacyFingerTracker:
    mp_hands = mp.solutions.hands
    hands = mp_hands.Hands(
        model_complexity=0,
```

```
        min_detection_confidence=0.5,
        min_tracking_confidence=0.5,
        max_num_hands=10)
    fingers_tips_ids = [
        mp_hands.HandLandmark.INDEX_FINGER_TIP,
        mp_hands.HandLandmark.MIDDLE_FINGER_TIP,
        mp_hands.HandLandmark.RING_FINGER_TIP,
        mp_hands.HandLandmark.PINKY_TIP,
    ]

    def detect(self, image):
        results = self.hands.process(image)
        if not results.multi_hand_landmarks:
            return list()
        return self.transform(results)

    def transform(self, detection_results):
        hands = list()
        for hand_index, hand_info in
enumerate(detection_results.multi_handedness):
            hand_label = hand_info.classification[0].label.upper()
            hand_landmarks =
detection_results.multi_hand_landmarks[hand_index]
            hand = HandResult(label=hand_label,
landmarks=hand_landmarks)
            for tip_index in self.fingers_tips_ids:
                finger_name = tip_index.name.split("_")[0]
                if (hand_landmarks.landmark[tip_index].y <
hand_landmarks.landmark[tip_index - 2].y):
                    hand.status[finger_name] = True
            thumb_tip_x =
hand_landmarks.landmark[self.mp_hands.HandLandmark.THUMB_TIP].x
            thumb_mcp_x =
hand_landmarks.landmark[self.mp_hands.HandLandmark.THUMB_TIP -
2].x
            if (hand_label == "RIGHT" and (thumb_tip_x <
thumb_mcp_x)) or (hand_label == "LEFT" and (thumb_tip_x >
thumb_mcp_x)):
                hand.status["THUMB"] = True
            hand.count = sum(hand.status.values())
            hands.append(hand)
        return hands

    @staticmethod
    def draw_landmarks_on_image(image, hands:list[HandResult]):
        if not hands:
            return image
        height, width, _ = image.shape
        for hand in hands:
            mp_drawing.draw_landmarksimage, hand.landmarks,
LegacyFingerTracker.mp_hands.HAND_CONNECTIONS,
mp_drawing_styles.get_default_hand_landmarks_style(),
```

```
mp_drawing_styles.get_default_hand_connections_style())
            x_coord    =    min(landmark.x    for    landmark    in
hand.landmarks.landmark)
            y_coord    =    min(landmark.y    for    landmark    in
hand.landmarks.landmark)
            x_coord,       y_coord       =       int(x_coord*width),
int(y_coord*height - MARGIN)
            cv2.putText(image,   hand.label,   (x_coord,   y_coord),
cv2.FONT_HERSHEY_DUPLEX,     FONT_SIZE,     HANDEDNESS_TEXT_COLOR,
FONT_THICKNESS, cv2.LINE_AA)
        return image
```

Лістинг файлу PersonDetector.py

```
import cv2
import numpy as np
from ultralytics import YOLO
from itertools import combinations

attr = ['x1', 'y1', 'x2', 'y2']

def map_pred_to_dict(boxes: list):
    return [dict(zip(attr, bbox)) for bbox in boxes]

def get_intersection_over_box(bb1, bb2):
    x_left = max(bb1['x1'], bb2['x1'])
    y_top = max(bb1['y1'], bb2['y1'])
    x_right = min(bb1['x2'], bb2['x2'])
    y_bottom = min(bb1['y2'], bb2['y2'])
    if x_right < x_left or y_bottom < y_top:
        return 0.0
    intersection_area = (x_right - x_left + 1) * (y_bottom - y_top
+ 1)
    bb1_area, bb2_area = get_bbox_area(bb1), get_bbox_area(bb2)
    iob1,     iob2     =     (intersection_area     /     bb1_area),
(intersection_area / bb2_area)
    return max(iob1, iob2)

def get_bbox_area(bbox):
    return   (bbox['x2']   -   bbox['x1']   +   1)   *   (bbox['y2']   -
bbox['y1'] + 1)

def get_duplicated_bboxes(bboxes, iob_threshold=.75):
    to_remove = list()
    for bbox1, bbox2 in combinations(bboxes, 2):
        iob = get_intersection_over_box(bbox1, bbox2)
        if iob >= iob_threshold:
            bb1_area,     bb2_area     =     get_bbox_area(bbox1),
get_bbox_area(bbox2)
            to_remove.append(bboxes.index(bbox1)   if   bb1_area   <=
bb2_area else bboxes.index(bbox2))
```

```
    return set(to_remove)

class PersonDetector:
    model = YOLO("yolov8n.pt", task='detect')
    model.to('cuda')

    def detect(self, image: np.ndarray):
        result = self.model.predict(image, classes=[0],
verbose=False)[0].boxes.data.cpu().numpy()
        boxes = map_pred_to_dict(result[:,:4])
                if len(boxes) > 1:
            duplicates = get_duplicated_bboxes(boxes)
            result = np.delete(result, list(duplicates), axis=0)
        return result

    def draw_bbox(self, image: np.ndarray, bbox: np.ndarray,
font=cv2.FONT_HERSHEY_DUPLEX, font_scale=0.5, box_thickness=2,
text_weight=1, color=(255, 0,0)) -> np.ndarray:
        x1, y1, x2, y2 = bbox
        p1, p2 = (int(x1), int(y1)), (int(x2), int(y2))
        image = cv2.rectangle(image, p1, p2, color, box_thickness)
        return image
```

Лістинг файлу auth.py

```
import functools
from db import db
from flask import (Blueprint, flash, g, redirect, render_template,
request, session, url_for)
from werkzeug.security import check_password_hash,
generate_password_hash
from sqlalchemy.exc import IntegrityError
import models

bp = Blueprint('auth', __name__, url_prefix='/auth')

@bp.route('/login', methods=('GET', 'POST'))
def login():
    if g.user and g.user is not None:
        return redirect(url_for('index'))
    if request.method == 'POST':
        username = request.form['name']
        password = request.form['password']
        error = None
        user = models.User.query.filter_by(name=username).first()
        if user is None:
            error = 'Incorrect username'
        elif not check_password_hash(user.password, password):
            error = 'Incorrect password'
        elif user.valid == 0:
            error = "Account not valid yet"
```

```
        if error is None:
            session.clear()
            session['user_id'] = user['id']
            return redirect(url_for('index'))
        flash(error, "error")
    return render_template('login.html')

@bp.route('/register', methods=("GET", "POST"))
def register():
    if request.method == "POST":
        print(request.form)
        password = request.form['password']
        name = request.form["name"]
        error = None
        if len(password) < 5:
            error = "Short password. Minimum 5 characters
required!"
        try:
            if error is None:
                user = models.User(name=name,
password=generate_password_hash(password))
                print(user)
                db.session.add(user)
                db.session.commit()
                flash("Registered", "success")
                return redirect(url_for('auth.login'))
        except IntegrityError:
            flash("User already exists", "error")
        except Exception as e:
            flash(str(e), "error")
    return render_template("register.html")

@bp.route('/logout')
def logout():
    session.clear()
    return redirect(url_for('auth.login'))

def login_required(view):
    @functools.wraps(view)
    def wrapped_view(**kwargs):
        if g.user is None:
            return redirect(url_for("auth.login"))
        return view(**kwargs)
    return wrapped_view

@bp.before_app_request
def load_logged_in_user():
    user_id = session.get("user_id")
    if user_id is None:
        g.user = None
    else:
        g.user = (models.User.query.get(user_id))
```

Лістинг файлу view.html

```
{% extends 'base.html' %}
{% block title%} View {% endblock %}
{% block content %}
<div class="container" style="margin: auto">
  <div class="container d-flex justify-content-between">
    <h2 class="pb-2 ps-2">Source {{source['name']}}</h2>
    <h2 class="pb-2 ps-2">
      <button    class="btn"    data-bs-toggle="modal"    data-bs-
target="#visibilityModal">
        <i class="fas fa-eye"></i> Edit visibility
      </button>
    </h2>
  </div>
  <div class="container text-center">
    <div class="row">
      <div class="col-8 p-2" style="min-height: 480px; min-width:
400px">
        <img
          src="{{url_for('static',     filename='media/unavailable-
image.jpg')}}"
          alt=""
          id="image"
        />
      </div>
      <div class="col-4 p-2">
        <div          class="list-group          list-group-flush"
id="personsList"></div>
      </div>
    </div>
    <div class="row">
      <div class="col" style="max-height: 200px; overflow: auto;
white-space: pre;" id="loggerDiv">
        <h4>Source {{source['name']}}</h4>
      </div>
    </div>
  </div>
</div>
<div class="modal fade" id="visibilityModal" tabindex="-1" aria-
labelledby="modalLabel" aria-hidden="true">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h1    class="modal-title    fs-5"    id="modalLabel">Change
visibility</h1>
        <button    type="button"    class="btn-close"    data-bs-
dismiss="modal" aria-label="Close" ></button>
      </div>
      <form  method="post"  action="{{url_for('change_visibility',
sid=sid)}}">
        <div class="modal-body">
```

```
          <div class="form-check">
            <input     class="form-check-input"     type="checkbox"
value="1"  id="person_visibility"  name="person_visibility"  {%  if
visibility['person_visibility'] %}checked {% endif %}>
            <label                          class="form-check-label"
for="person_visibility"> Person's detection visibility</label>
          </div>
          <div class="form-check">
            <input     class="form-check-input"     type="checkbox"
value="1"   id="name_visibility"   name="name_visibility"   {%   if
visibility['name_visibility'] %}checked {% endif %}>
            <label class="form-check-label" for="name_visibility">
              Name visibility
            </label>
          </div>
          <div class="form-check">
            <input     class="form-check-input"     type="checkbox"
value="1"
            id="hand_visibility" name="hand_visibility" {% if
            visibility['hand_visibility'] %}checked {% endif %}>
            <label class="form-check-label" for="hand_visibility">
              Hand landmarks visibility
            </label>
          </div>
          <div class="form-check">
            <input     class="form-check-input"     type="checkbox"
value="1"
            id="count_visibility" name="count_visibility" {% if
            visibility['count_visibility'] %}checked {% endif %}>
            <label                          class="form-check-label"
for="count_visibility">
              Count visibility
            </label>
          </div>
        </div>
        <div class="modal-footer">
          <button
            type="button"
            class="btn btn-secondary"
            data-bs-dismiss="modal"
          >
            Close
          </button>
          <button        type="submit"        class="btn        btn-
primary">Save</button>
        </div>
      </form>
    </div>
  </div>
</div>
```

```
<div class="modal fade" id="personModal" tabindex="-1" aria-
labelledby="modalLabel" aria-hidden="true">
  <div class="modal-dialog">
    <div class="modal-content" id="personModalContent">
    </div>
  </div>
</div>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/4.0.1/socket
.io.js"
  integrity="sha512-
q/dWJ3kcmjBLU4Qc47E4A9kTB4m3wuTY7vkFJDTZKjTs8jhyGQnaUrxa0Ytd0ssMZh
bNua9hE+E7Qv1j+DyZwA=="
  crossorigin="anonymous"
></script>
 <script type="text/javascript" charset="utf-8">
  var loggerDiv = document.getElementById('loggerDiv');
  var createPersonAction = '{{url_for("create_person",
sid=sid)}}';
  var currentNames = Array();
  var access = '{{source["access"]}}';
  var connectType = '{{source["type"]}}';
  const imageElement = document.getElementById("image");
  var socket = io();
  socket.on("connect", function (msg) {
    socket.emit("join", { token: access });
    logToPage('Connected to stream');
  });

  socket.on("response", function (msg) {
    if (msg.token == access) {
      var personsList = document.getElementById("personsList");
      personsList.innerHTML = "";
      imageElement.src = "" + msg.image;
      msg.persons.forEach((element) => {
        createPersonItem(element);
        if (element.total_count > 0) {
          logToPage(`${element.name} triggers command
${element.total_count}`);
        }
      });
    }
  });
  function createPersonItem(person) {
    if (person.name == "No face detected") return false;
    var personsList = document.getElementById("personsList");
    var holder = document.createElement("div");
    holder.classList.add("container", "d-flex", "flex-row", "mb-
3");
    var trigger = document.createElement("a");
    trigger.classList.add("list-group-item", "list-group-item-
action");
```

```
    trigger.addEventListener("click", function () {
      processPerson(person);
      return false;
    });
    trigger.href = "#";
    holder.appendChild(trigger);
    var name = document.createElement("h3");
    name.className = "mb-1";
    name.textContent = person.name;
    trigger.appendChild(name);
    personsList.appendChild(holder);
  }
  function processPerson(person) {
    var                        modalContent                              =
document.getElementById('personModalContent');
    modalContent.innerHTML = '';
    var modalHeader = document.createElement('div');
    modalHeader.className = 'modal-header';
    var create = person.name.startsWith("Unknown");
    modalHeader.innerHTML = `
      <h1  class="modal-title  fs-5">${create?  'Create':  'View'}
person</h1>
        <button
          type="button"
          class="btn-close"
          data-bs-dismiss="modal"
          aria-label="Close"
        ></button>
    `;
    var form = document.createElement('form');
    form.action      =      create?      createPersonAction      :
'{{url_for("edit_person",                                  sid=sid,
name="holder_name")}}'.replace('holder_name', person.name);
    form.method = 'POST';
    var modalBody = document.createElement('div');
    modalBody.className = 'modal-body';
    var name = document.createElement('div');
    name.classList.add('input-group', 'mb-3');
    name.innerHTML = `
    <span class="input-group-text" id="name-addon1">Name</span>
    <input  type="text"  class="form-control"  placeholder="Name"
aria-label="Name"
        aria-describedby="name-addon1"                  name='name'
value='${create? "": person.name}' required>
    `;
    modalBody.appendChild(name);
    if(!create){
      for (let index = 1; index < 11; index++) {
        var instruction = document.createElement('div');
        instruction.classList.add('input-group', 'mb-3');
        var text = person.instructions[index];
        instruction.innerHTML = `
```

```
        <span          class="input-group-text"          id="instr-
addon${index}">Instruction ${index}</span>
        <input           type="text"           class="form-control"
placeholder="Instruction" aria-label="Instruction"
          aria-describedby="instr-addon${index}" value='${text !==
undefined? text : ""}' name='instr-${index}'>
        `;
        modalBody.appendChild(instruction);
      }
    } else{
      var face = document.createElement('input');
      var json_arr = JSON.stringify(person.encoding);
      face.type = 'text';
      face.value = json_arr;
      face.hidden = true;
      face.name = 'face_encoding';
      modalBody.appendChild(face);
    }

    var modelFooter = document.createElement('div');
    modelFooter.className = 'modal-footer';
    modelFooter.innerHTML = `
    <button
      type="button"
      class="btn btn-secondary"
      data-bs-dismiss="modal"
    >
      Close
    </button>
    <button    type="submit"    class="btn    btn-primary">${create?
'Create': 'Save'}</button>
    `;
    if (!create){
      var dltButton = document.createElement('button');
      dltButton.textContent = 'Delete'
      dltButton.type = 'button';
      dltButton.classList.add("btn", "btn-outline-danger");
      dltButton.value    =    "{{url_for('delete_person',    sid=sid,
name='holder_name')}}".replace('holder_name', person.name);
      dltButton.style.marginRight = '50%';
      dltButton.addEventListener("click", function (item) {
        approve(dltButton);
      })
      modelFooter.insertBefore(dltButton, modelFooter.firstChild);
    }
    form.append(modalBody, modelFooter);
    modalContent.append(modalHeader, form);
    const myModal = new bootstrap.Modal('#personModal', {});
    myModal.show();
  }
  function logToPage(msg) {
```

```
    var time = new Date().toISOString().replace('T', ' ').slice(0,
19);
    var item = document.createElement('h4');
    item.textContent = time + '\t' + msg;
    loggerDiv.prepend(item);
    while (loggerDiv.childElementCount >= 300){
      loggerDiv.removeChild(loggerDiv.lastChild);
    }
  }
  function approve(elem){
    console.log(elem.value);
      if (window.confirm('Really delete this person?')){
          window.location = elem.value;
      }
  }

</script>

{% endblock %}
```