

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
ЧЕРНІВЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ  
ІМЕНІ ЮРІЯ ФЕДЬКОВИЧА**

**Навчально-науковий інститут фізико-технічних та комп'ютерних наук  
кафедра комп'ютерних наук**

**Веб-сервіс розпізнавання обличчя з Face-API.JS та штучним інтелектом**

**Курсова робота**

**Рівень вищої освіти – перший (бакалаврський)**

***Виконав:***

студент 4 курсу, 444б групи

**Горбан О.Ю.**

***Керівник:***

кандидат фізико-математичних наук,

асистент **Карачевцев А.О.**

**Чернівці – 2024**

**Чернівецький національний університет імені Юрія Федьковича**

Навчально-науковий інститут фізико-технічних та комп'ютерних наук

Кафедра Комп'ютерних наук

Спеціальність Комп'ютерні науки

Освітній ступінь Бакалавр

Форма навчання денна курс 4 група 444б

**ЗАТВЕРДЖУЮ**

Завідувач кафедри \_\_\_\_\_

( підпис)

Ушенко Ю.О.

(ініціали, прізвище)

\_\_\_\_\_ 2024 р.

**ЗАВДАННЯ**

**НА КУРСОВУ РОБОТУ СТУДЕНТА**

Горбана Олександра Юрійовича

(прізвище , ім'я, по батькові)

**1. Тема роботи**

Веб-сервіс розпізнавання обличчя з Face API JS та штучним інтелектом

затверджена протоколом засідання кафедри від «\_»\_\_2024 року № 1

**2. Термін подання студентом закінченої роботи 11.11.2024**

**3. Вхідні дані до роботи**

Visual Studio Code, Face-API.js

**4. Зміст розрахунково-пояснювальної записки.**

Вступ

Розділ I. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області. обґрунтування необхідності використання пз в предметній області

1.2 Аналіз програм-аналогів

1.3 Постановка задачі на розробку ПЗ

РОЗДІЛ II. РОЗРОБКА ПЗ ДЛЯ КУРСОВОЇ РОБОТИ

2.1 Вибір інструментарію для реалізації ПЗ для курсової роботи

2.2 Проектування та реалізація системи зберігання даних

2.3 Реалізація ПЗ для курсової роботи

2.4 Демонстрація роботи програми

Висновки

Список використаних джерел

Додатки

5. Перелік графічного, наочного матеріалу

Скріншоти інтерфейсу, скріншоти результатів розпізнавання, скріншоти програм аналогів

6. Консультант(и) курсової роботи

**КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ КУРСОВОЇ РОБОТИ**

№	Назва етапів курсової роботи	Термін виконання етапів роботи	Примітки
1	Отримання завдання на курсову роботу		виконано
2	Аналіз предметної області, дослідження літератури та матеріалів на задану тему		виконано
3	Аналіз існуючих аналогів програмного забезпечення		виконано
4	Постановка задачі за темою курсової роботи		виконано
5	Вибір інструментальних засобів розробки системи		виконано
6	Проектування структури та алгоритму роботи розроблюваної системи		виконано
7	Формування інструкції користувача		виконано
8	Написання розділів пояснювальної записки		виконано
9	Представлення закінченої роботи на перевірку		виконано
10	Захист курсової роботи		виконано

Студент \_\_\_\_\_ Горбан О.Ю.  
(підпис)

Науковий керівник \_\_\_\_\_ Карачевцев А.О.  
(підпис)

«\_\_\_» \_\_\_\_\_ 2024 р.

## АНОТАЦІЯ

У курсовій роботі на тему «Веб-сервіс розпізнавання обличчя з Face-API.JS та штучним інтелектом» досліджено сучасні методи обробки та розпізнавання облич з використанням JavaScript. Визначено вимоги до програмного забезпечення, обрано інструменти для реалізації розпізнавання облич у веб-середовищі та розроблено інтерфейс для користувача. Сервіс інтегрує нейромережеву модель розпізнавання облич на основі Face-API.JS, яка забезпечує швидке й точне ідентифікування зображень.

Курсова робота містить результати власних досліджень. Використання ідей, результатів і текстів наукових досліджень інших авторів мають посилання на відповідне джерело.

## **ABSTRACT**

In the course project titled "Face Recognition Web Service with Face-API.js and Artificial Intelligence," modern methods of face processing and recognition using JavaScript are examined. Software requirements were defined, tools were selected for implementing face recognition in a web environment, and a user interface was developed. The service integrates a neural network-based face recognition model using Face-API.js, providing quick and accurate image identification.

The course project includes results of original research. Use of ideas, findings, and texts from other authors' scientific studies is cited accordingly.

## ЗМІСТ

ВСТУП.....	7
РОЗДІЛ І. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ .....	9
1.1 Опис предметної області. Обґрунтування необхідності використання ПЗ в предметній області .....	9
1.2 Аналіз програм-аналогів.....	10
1.3 Постановка задачі.....	15
РОЗДІЛ ІІ. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ .....	16
2.1 Вибір інструментарію для реалізації ПЗ для курсової роботи.....	16
2.2 Проектування та реалізація системи зберігання даних .....	16
2.3 Реалізація ПЗ для курсової роботи .....	21
2.4 Демонстрація роботи програми .....	24
ВИСНОВКИ .....	28
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ .....	30
ДОДАТКИ .....	31

## ВСТУП

У курсовій роботі на тему «Веб-сервіс розпізнавання обличчя з Face API JS та штучним інтелектом» планується розробка веб-сервісу, який використовує технології розпізнавання облич за допомогою бібліотеки Face API JS. Цей сервіс дозволяє ідентифікувати осіб на зображеннях, що завантажуються користувачем, із застосуванням нейромережевих моделей та штучного інтелекту.

Для реалізації цього проєкту я обрав стек технологій, включаючи HTML, CSS та JavaScript. HTML і CSS забезпечують структуру та дизайн веб-сторінок, а JavaScript із бібліотекою Face API JS відповідає за обробку зображень та реалізацію розпізнавання облич.

Веб-сервіс не потребує бази даних, оскільки всі зображення для розпізнавання будуть завантажуватися безпосередньо з мого GitHub репозиторію. В цьому проєкті використовуються різні моделі для розпізнавання обличчя, що дозволяє ефективно ідентифікувати обличчя та порівнювати їх з наявними даними.

Метою курсової роботи є створення інтерфейсу для зручного завантаження зображень, обробки їх за допомогою моделей машинного навчання, а також відображення результатів розпізнавання у вигляді маркованих прямокутників на зображеннях.

Перший розділ курсової роботи розглядає теоретичні основи роботи з технологією розпізнавання обличчя, аналізує програмні аналоги та визначає завдання для реалізації проєкту. У другому розділі детально описуються вибір інструментів для розробки веб-сервісу, а також реалізація функціоналу для роботи з зображеннями та інтерфейсом користувача.

Основними завданнями курсової роботи є:

- Провести аналіз предметної області та обґрунтувати необхідність використання технології розпізнавання облич.
- Здійснити огляд і аналіз аналогічних сервісів.
- Поставити завдання для розробки веб-сервісу.

- Обрати та обґрунтувати вибір інструментарію для створення програмного забезпечення.
- Інтегрувати Face API JS для розпізнавання облич у веб-середовище.
- Розробити веб-сервіс для розпізнавання облич.
- Підготувати інструкцію користувача для роботи з веб-сервісом.



## РОЗДІЛ І. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

### 1.1 Опис предметної області. Обґрунтування необхідності використання ПЗ в предметній області

Цей веб-сервіс призначений для розпізнавання облич на основі зображень, завантажених з репозиторію Github. Використовуючи технології розпізнавання облич і штучного інтелекту, система ідентифікує осіб за зображеннями, що зберігаються у відкритому доступі на GitHub. Це дозволяє автоматично виявляти обличчя на зображеннях та здійснювати порівняння з наявними зразками, що дає можливість виконувати точну ідентифікацію осіб.

#### **Значення системи:**

- **Розпізнавання облич:** Система дозволяє автоматично виявляти обличчя на зображеннях, що завантажуються користувачами. Це важливий етап для подальшої ідентифікації осіб.
- **Ідентифікація облич:** Використовуючи зображення, завантажені з GitHub репозиторію, система порівнює виявлені обличчя з вже наявними зразками, що дозволяє визначати, чи є співпадіння з обличчями, які зберігаються в системі. Це дає можливість виконувати автоматичну ідентифікацію осіб на основі їхніх облич, що може бути використано в різних галузях, наприклад, для перевірки особистості чи для організації віртуальних зустрічей.

#### **Основні функції системи:**

1. **Завантаження зображень:** Користувач може завантажувати зображення для подальшого аналізу.
2. **Розпізнавання облич на зображеннях:** Після завантаження система автоматично виявляє обличчя на зображеннях, виділяючи їх та готуючи до порівняння з існуючими даними.
3. **Ідентифікація облич за допомогою Face API JS:** Використовуючи зразки облич, що зберігаються в репозиторії GitHub, система порівнює виявлені обличчя з цими зразками для точного визначення особи.

4. **Порівняння облич і видача результатів:** Після виявлення та порівняння облич система виводить результат і визначає найкраще співпадіння для кожного зображення.
5. **Інтерфейс для зручності користувачів:** Програмний інтерфейс забезпечує зручне завантаження зображень, перегляд результатів розпізнавання та взаємодію з іншими функціями системи.

**Необхідність використання ПЗ в предметній області:** Цей веб-сервіс дозволяє вирішити низку задач, пов'язаних з автоматичним розпізнаванням осіб на зображеннях, що є важливим для застосувань у сфері безпеки, інтерфейсів користувача, а також для розпізнавання та аналізу облич у великій кількості зображень. Технології штучного інтелекту і машинного навчання, що використовуються в цьому проекті, дозволяють ефективно і точно здійснювати розпізнавання, значно підвищуючи зручність і ефективність таких систем.

## 1.2 Аналіз програм-аналогів

Для аналізу програм-аналогів були використані наступні сервіси та додатки для розпізнавання облич:

1. **FaceApp** (<https://www.faceapp.com>)
2. **Reface** (<https://www.reface.app>)
3. **FindFace** (<https://findface.pro>)

### FaceApp

FaceApp – це популярний мобільний додаток для редагування фотографій, який використовує алгоритми розпізнавання обличчя для змінювання зображень. Окрім базового розпізнавання облич, додаток дозволяє застосовувати фільтри, змінювати вік, емоції, а також здійснювати заміну облич на зображеннях.

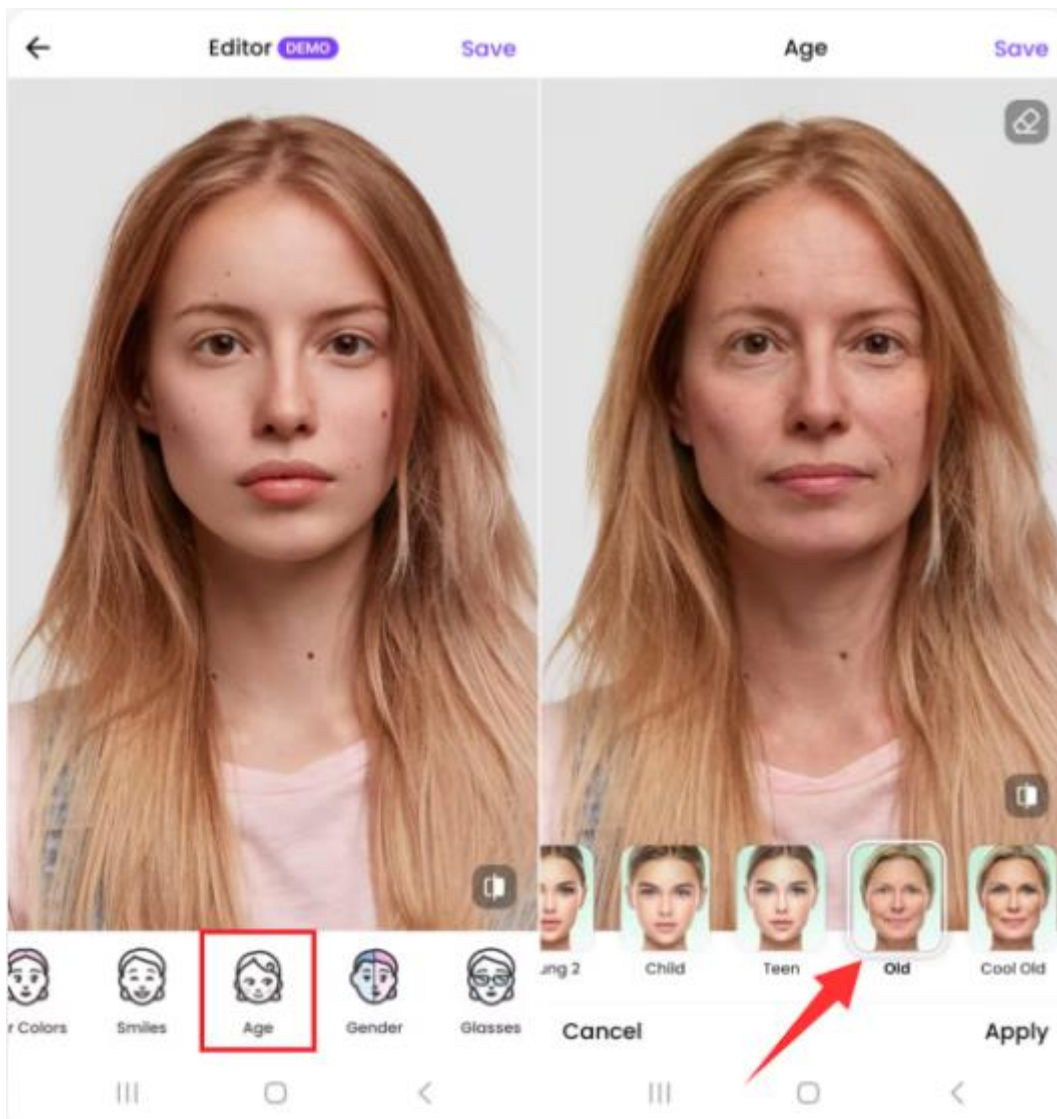


Рисунок 1 – Демонстрація одного з фільтрів додатку FaceApp

**Плюси:**

- Широкий спектр функцій, таких як зміна віку, статі, емоцій, а також можливість вставляти обличчя на інші зображення.
- Висока точність розпізнавання облич.
- Доступний на мобільних платформах iOS і Android.

**Мінуси:**

- Обмежена безкоштовна версія, багатфункціональність доступна лише після покупки преміум-версії.
- Можливі питання щодо конфіденційності даних користувачів, оскільки додаток зберігає фото на своїх серверах.

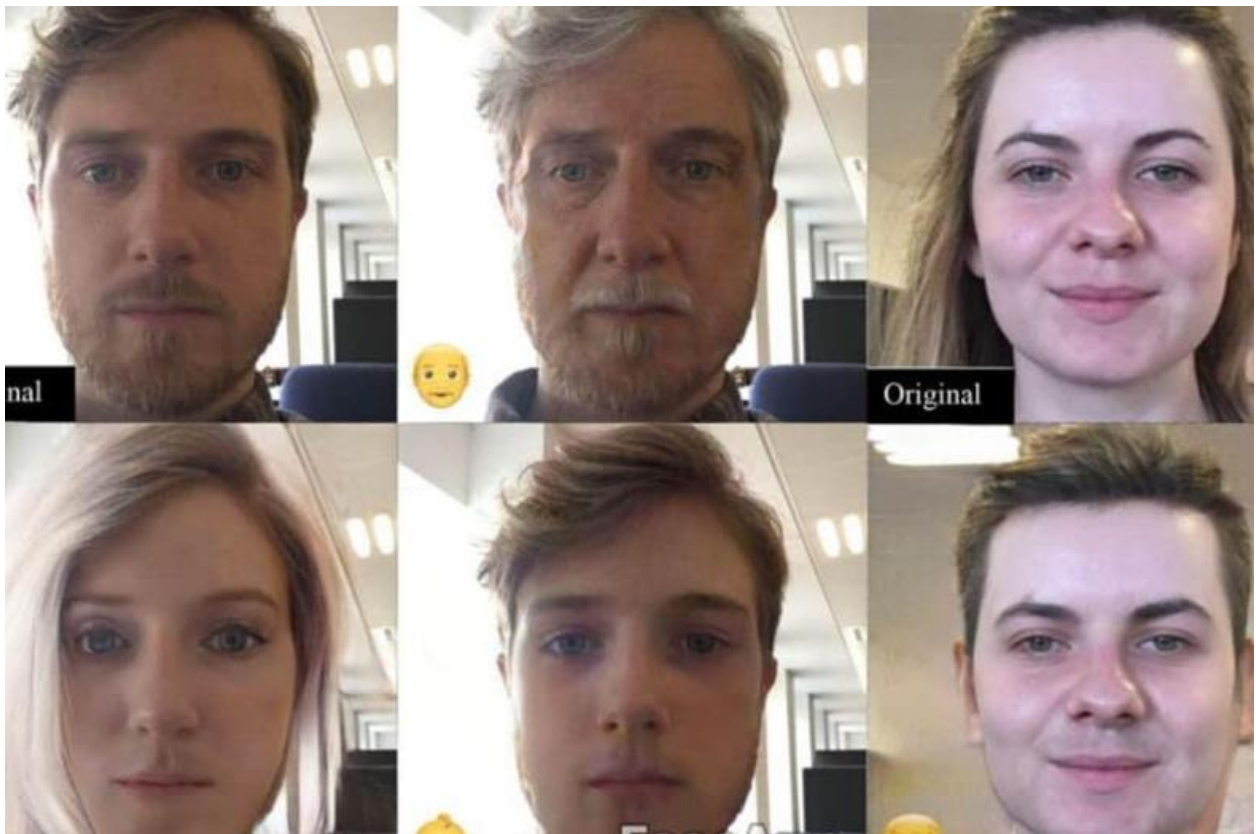


Рисунок 2 – Приклади використання фільтрів в додатку FaceApp

## Reface

Reface – це український мобільний додаток для обміну обличчями на відео та зображеннях, який використовує алгоритми глибокого навчання для точного перенесення обличчя людини на фото або відео з іншими людьми.

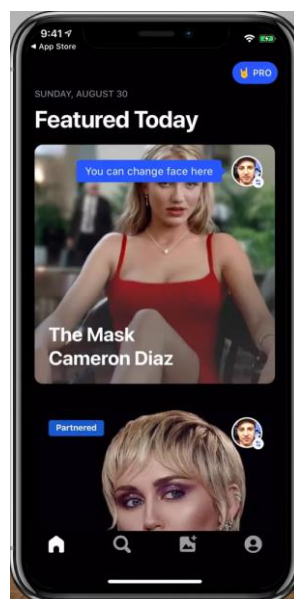


Рисунок 3 – Головна сторінка додатку Reface

### Плюси:

- Швидке і точне перенесення обличчя на відео або фото.
- Підтримка великої кількості фільтрів та можливостей для креативного використання.
- Легкий у використанні, інтуїтивно зрозумілий інтерфейс.

### Мінуси:

- Не знайшов.

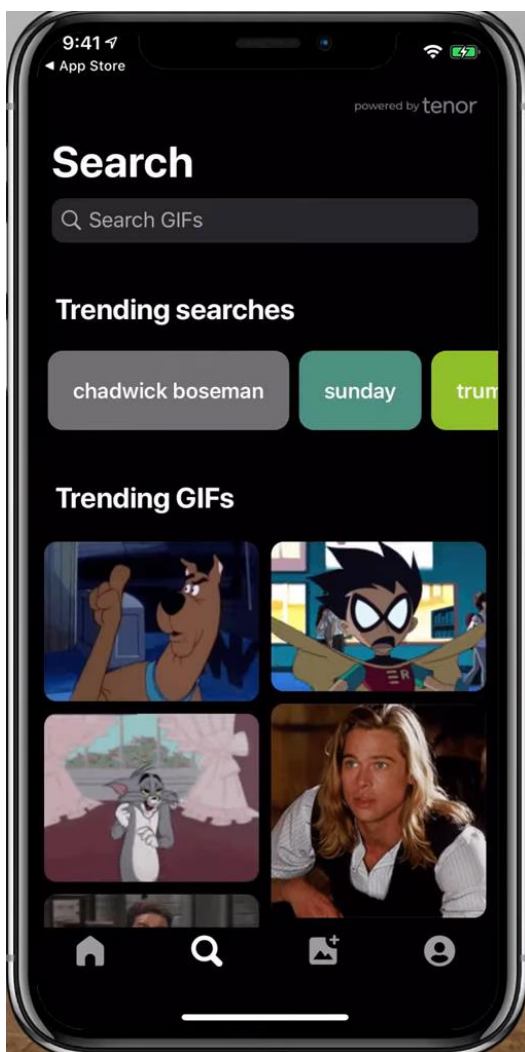


Рисунок 4 – Сторінка пошуку додатку Reface

### FindFace

FindFace – сервіс для розпізнавання облич, який використовується для пошуку людей за їх фотографіями в соціальних мережах та інших онлайн-ресурсах. Цей



сервіс орієнтований на безпеку та ідентифікацію осіб у публічних місцях або онлайн.

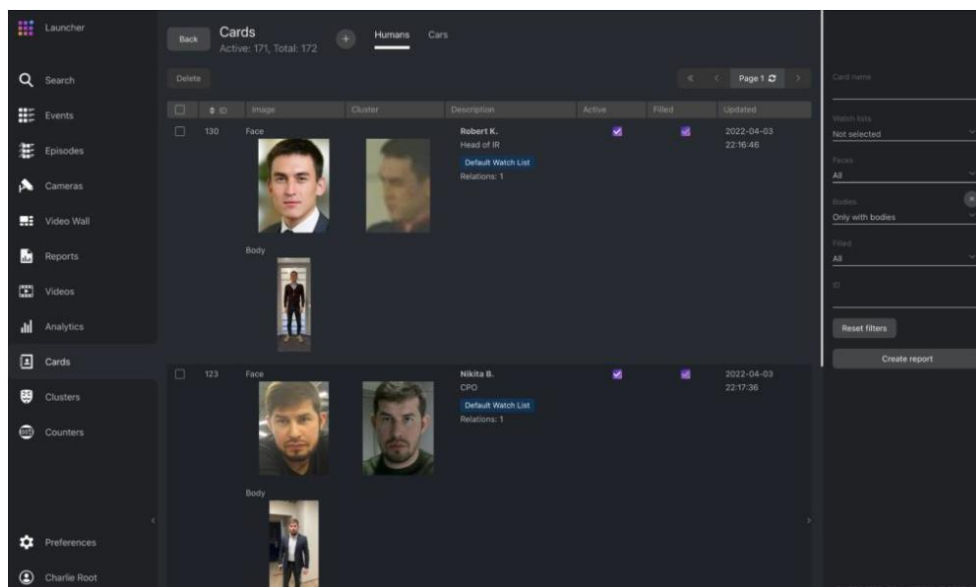


Рисунок 5 – Сторінка додатку FindFace

#### Плюси:

- Висока точність розпізнавання облич навів при поганій якості зображень.
- Можливість порівняти обличчя на фотографіях з базами даних.
- Використовується для пошуку осіб у великих масивах даних.

#### Мінуси:

- Платна підписка на використання з більш високим рівнем доступу до функцій.

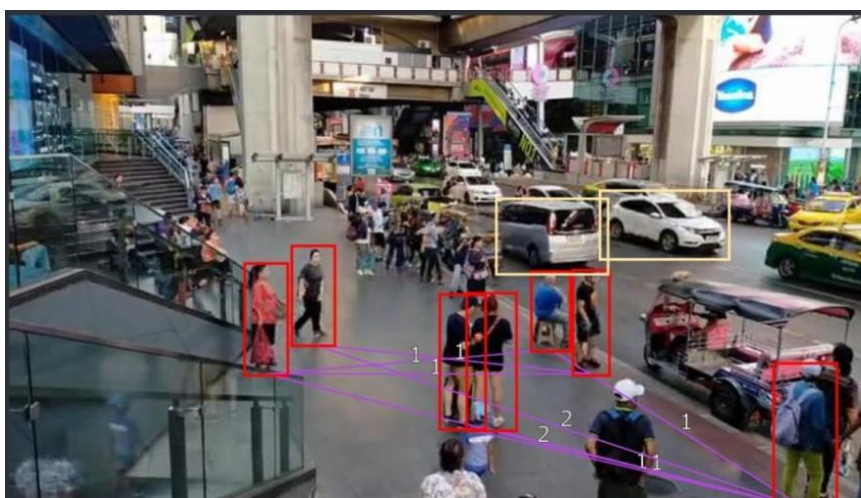


Рисунок 6 – Демонстрація роботи додатку FindFace

### **1.3 Постановка задачі**

Метою роботи є створення веб-сервісу для розпізнавання обличчя з використанням Face API JS та штучного інтелекту, що дозволить користувачам швидко і точно ідентифікувати обличчя на зображеннях, завантажених з GitHub. Ця система має на меті полегшити процес розпізнавання обличчя для використання в різноманітних сферах, таких як безпека, ідентифікація особистості чи організація баз даних для аналізу.

Основні завдання проекту:

- 1) Розробка веб-сервісу для розпізнавання обличчя з використанням JavaScript і Face API.
- 2) Можливість завантаження зображень з GitHub, що дозволяє зручно добавляти свої фотографії для аналізу системою.
- 3) Реалізація алгоритмів ідентифікації обличчя, що дозволяють зіставляти обличчя на зображеннях з наявними.
- 4) Розробка інтуїтивно зрозумілого інтерфейсу користувача для взаємодії з сервісом, включаючи можливість завантаження зображень та перегляд результатів розпізнавання.

## РОЗДІЛ II. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

### 2.1 Вибір інструментарію для реалізації ПЗ для курсової роботи

Для реалізації мого веб-сервісу для розпізнавання обличчя я використовував наступний стек технологій та інструментів:

- **JavaScript (Face API)**
- **HTML, CSS**
- **Visual Studio Code** (редактор коду)
- **GitHub** (для зберігання зображень та коду)

#### **JavaScript (Face API):**

Опис: Мова програмування, яка використовується для створення інтерактивних веб-сайтів. У моєму проєкті я використовував Face API для розпізнавання облич на зображеннях.

**Face-API.js** — це JavaScript бібліотека для розпізнавання облич, заснована на глибоких нейронних мережах. Вона дозволяє виявляти обличчя в зображеннях або відео та здійснювати ряд додаткових операцій, таких як ідентифікація та емоційний аналіз обличчя. Бібліотека використовує моделі машинного навчання, щоб здійснити розпізнавання ідентичних облич, виявляти емоції, визначати орієнтацію голови, розпізнавати ключові точки обличчя тощо.

Механізм визначення обличчя в бібліотеці **Face-API.js** заснований на **глибоких нейронних мережах**, зокрема використовується модель **SSD Mobilenet v1**, яка є ефективною для швидкого виявлення облич в реальному часі.

Використання:

- Створення функцій для завантаження зображень та обробки їх через API.
- Інтеграція Face API для порівняння обличчя з вже завантаженими зображеннями.
- Реалізація обробки подій на клієнтському боці для надання користувачу результатів розпізнавання в реальному часі.



## **HTML:**

Опис: Мова розмітки для створення структури веб-сторінок.

Використання:

- Створення основної структури веб-сторінки для завантаження зображень.
- Використання тегів для організації контенту та забезпечення правильної навігації через інтерфейс.

## **CSS:**

Опис: Мова стилів для візуального оформлення веб-сторінок.

Використання:

- Оформлення інтерфейсу користувача (UI) для створення зрозумілого і зручного для користувачів дизайну.
- Визначення стилів для кнопок, форм завантаження, результатів розпізнавання, текстів та інших елементів веб-сторінки.

## **Visual Studio Code:**

Опис: Легке і потужне середовище для розробки, яке підтримує JavaScript, HTML та CSS.

Використання:

- Написання коду для серверної і клієнтської частини проекту.
- Використовував розширення для роботи з JavaScript та налагодження коду.
- Підключення до GitHub для збереження версій та спільної роботи.

## **GitHub:**

Опис: Платформа для хостингу та управління проектами через систему контролю версій Git.

Використання:

- Зберігання коду проекту.
- Завантаження та зберігання зображень для розпізнавання.
- Спільна робота над кодом та спостереження за змінами в проекті.

## 2.2 Проектування та реалізація системи зберігання даних

Оскільки в моєму проєкті не використовується традиційна база даних для зберігання інформації про користувачів та зображень, замість цього було розроблено альтернативний підхід для зберігання та обробки даних. Основною одиницею зберігання є **репозиторій на GitHub**, де зберігаються фотографії для розпізнавання.

### Зберігання даних:


1. **Зображення користувачів** зберігаються в публічному репозиторії на GitHub. Для кожного користувача створюється відповідний набір зображень, які використовуються для тренування моделі Face API та для подальшої ідентифікації. Зображення організовуються за принципом: кожен набір фотографій має власну папку, назва якої співпадає з ідентифікацією користувача.

### Обробка даних:

1. При завантаженні зображення користувача, система звертається до репозиторію GitHub, де порівнює завантажене фото з вже наявними обличчями в репозиторії.
2. За допомогою **Face API** відбувається порівняння зображень, після чого надається результат, що підтверджує або спростовує ідентифікацію обличчя.

Цей підхід дозволяє уникнути необхідності в створенні та управлінні традиційною базою даних, що значно спрощує реалізацію проєкту, знижуючи потребу в обробці та збереженні великих обсягів даних у локальній базі.

Cursova-Face-Recognition / labeled\_images /

 **OleksandrHorban** photo2

Name	Last commit message
..	
Daniel Hodoba	Cursova
Dima Banar	Cursova
Ivan Zagreichyk	photo
Mariya Yarema	Cursova
Nastya Krupchak	photo
Nazar Bedny	Cursova
Nazar Matseykiv	photo2
Oleg Bogutskyi	Cursova
Rostik Daskaliuk	photo2
Rostyslav Dzedzinsky	Cursova
Sasha Berlyak	Cursova
Stepan Horyniuk	photo2
Viktoria Baranska	photo2
Vitaliy Bondarenko	photo
Vlad Palichyk	photo2

Рисунок 7 – Структура репозиторія з зображеннями на Github

Cursova-Face-Recognition / labeled\_images / **Daniel Hodoba** /

 **OleksandrHorban** Cursova

Name	Last commit message
..	
1.jpg	Cursova
2.jpg	Cursova
3.jpg	Cursova
4.jpg	Cursova
5.jpg	Cursova

Рисунок 8 – Зміст однієї з папок репозиторія.

Зображення пронумеровані таким чином (1.jpg, 2.jpg, 3.jpg тощо) для кожного користувача, оскільки це забезпечує зручність автоматизації процесу завантаження та обробки фотографій.

Планую використати саме цей підхід, тому що він дозволяє програмно зручно обробляти та завантажувати зображення через API, де кожне зображення може бути завантажено за номером та ім'ям файлу, що полегшує роботу з набором даних. В коді планую застосовувати цикл, що для кожного користувача завантажує 5 зображень за допомогою інтерфейсу **Face API**, де номери зображень виступають як послідовні індекси. Такий спосіб дозволяє динамічно підключати зображення до системи без необхідності вказувати імена файлів вручну, а також забезпечує стандартизоване й ефективне зберігання фото в репозиторії.

## 2.3 Реалізація ПЗ для курсової роботи

### 1. Структура проєкту

- **labeled\_images:** Папка, що містить підкаталоги для кожного однокласника, в яких зберігаються п'ять зображень для тренування моделі. Під час роботи веб-сервісу зображення завантажуються не з локальної папки, а з копії цієї папки на Github.
- **models:** Папка, що містить натреновані моделі для розпізнавання облич, зокрема:
  - face\_landmark\_68\_model-weights\_manifest.json
  - face\_recognition\_model-weights\_manifest.json
  - інші моделі для різних типів розпізнавання.
- **test\_images:** Папка для зображень, які використовуються для тестування системи.
- **face-api.min.js:** Основна бібліотека, що забезпечує функціональність розпізнавання облич.
- **index.html:** Головна HTML сторінка, на якій користувач взаємодіє з інтерфейсом.
- **script.js:** Скрипт, що містить логіку для завантаження зображень, обробки їх та порівняння з відомими зображеннями.

### 2. Покрокова реалізація

#### 2.1 Завантаження моделей

На початку завантажуються необхідні моделі з папки models. Використовуючи функцію **Promise.all()**, я асинхронно завантажую три моделі для розпізнавання обличчя, точок на обличчі та для виявлення облич за допомогою SSD MobileNet:

```
Promise.all([
  faceapi.nets.faceRecognitionNet.loadFromUri('/models'),
  faceapi.nets.faceLandmark68Net.loadFromUri('/models'),
  faceapi.nets.ssdMobilenetv1.loadFromUri('/models')
]).then(start)
```

Рисунок 9 – Функція Promise.all()

## 2.2 Завантаження зображень для тренування

Зображення для кожного користувача знаходяться в папці labeled\_images в репозиторії на Github. Для кожного імені в списку (наприклад, Daniel Hodoba, Dima Banar і т.д.) завантажуються п'ять зображень. Ці зображення обробляються для отримання дескрипторів облич, які зберігаються для подальшого порівняння:

```
function loadLabeledImages() {
  const labels = ['Daniel Hodoba', 'Dima Banar', 'Mariya Yarema', 'Nazar Bedny', 'Oleg Bogutskyi', 'Rostyslav Dzedzinsky', 'Sasha Berlyak'];
  return Promise.all(
    labels.map(async label => {
      const descriptions = [];
      for (let i = 1; i <= 5; i++) {
        console.log(`Завантажую зображення ${i} для ${label}...`);
        const img = await faceapi.fetchImage(`https://raw.githubusercontent.com/OleksandrHorban/Cursova-Face-Recognition/main/labeled_images/${label}/${i}.jpg`);
        const detections = await faceapi.detectSingleFace(img).withFaceLandmarks().withFaceDescriptor();
        descriptions.push(detections.descriptor);
        console.log(`Зображення ${i} для ${label} завантажено.`);
      }
      return new faceapi.LabeledFaceDescriptors(label, descriptions);
    })
  );
}
```

Рисунок 10 – Функція loadLabeledImages()

Ці дескриптори представляють характеристики обличчя кожної особи, які будуть використовуватися для порівняння з новими зображеннями.

## 2.3 Обробка зображень користувача

Після завантаження моделей і зображень для тренування користувач може завантажити зображення для перевірки розпізнавання. Це зображення обробляється наступним чином:

1. Користувач завантажує зображення через форму HTML.

2. Це зображення перетворюється в зображення, яке може обробити **Face API**.
3. Далі проводиться виявлення обличчя на зображенні і порівняння з дескрипторами, отриманими для тренування.

```
// Встановлюємо повідомлення про успішне завантаження моделей
statusMessage.textContent = 'Модель завантажено! Тепер завантажте зображення для розпізнавання.'

imageUpload.addEventListener('change', async () => {
  statusMessage.textContent = '' // Очищаємо повідомлення, коли починається завантаження зображення

  if (image) image.remove()
  if (canvas) canvas.remove()

  image = await faceapi.bufferToImage(imageUpload.files[0])
  image.style.maxWidth = '100%'
  container.append(image)
  canvas = faceapi.createCanvasFromMedia(image)
  container.append(canvas)

  const displaySize = { width: image.width, height: image.height }
  faceapi.matchDimensions(canvas, displaySize)

  const detections = await faceapi.detectAllFaces(image).withFaceLandmarks().withFaceDescriptors()
  const resizedDetections = faceapi.resizeResults(detections, displaySize)
  const results = resizedDetections.map(d => faceMatcher.findBestMatch(d.descriptor))

  results.forEach((result, i) => {
    const box = resizedDetections[i].detection.box
    const drawBox = new faceapi.draw.DrawBox(box, { label: result.toString() })
    drawBox.draw(canvas)
  })

  // Виводимо повідомлення про успішну обробку зображення
  statusMessage.textContent = 'Лиця успішно розпізнано!'
})
}
```

Рисунок 11 – Функції для обробки зображень користувача.

## 2.4 Виведення результатів

Розпізнане обличчя порівнюється з наявними дескрипторами, і результат виводиться на екран разом з підписом, який містить ім'я особи, до якої належить обличчя.

```
results.forEach((result, i) => {
  const box = resizedDetections[i].detection.box
  const drawBox = new faceapi.draw.DrawBox(box, { label: result.toString() })
  drawBox.draw(canvas)
})
```

Рисунок 12 – Код для виведення результату

## 2.4 Демонстрація роботи програми

### 1. Запуск програми

Для запуску програми спочатку потрібно налаштувати проект. Я запускаю сервер та веб-сторінку за допомогою HTML, JavaScript та використовуючи бібліотеку face-api.js за допомогою кнопки **Open with Live Server**.

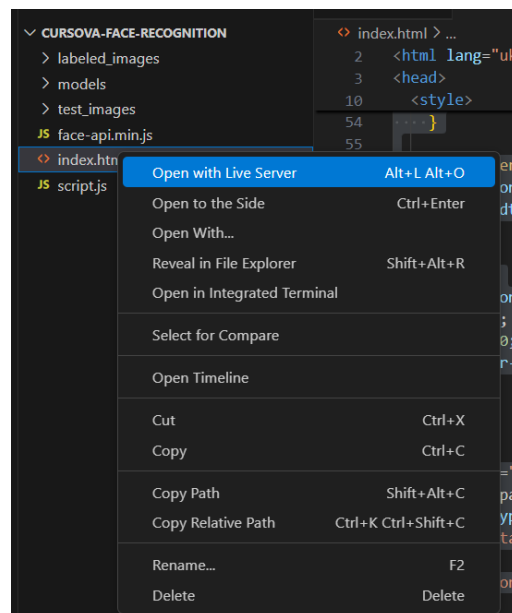


Рисунок 13 – Запуск програми

Після цього відкривається програма

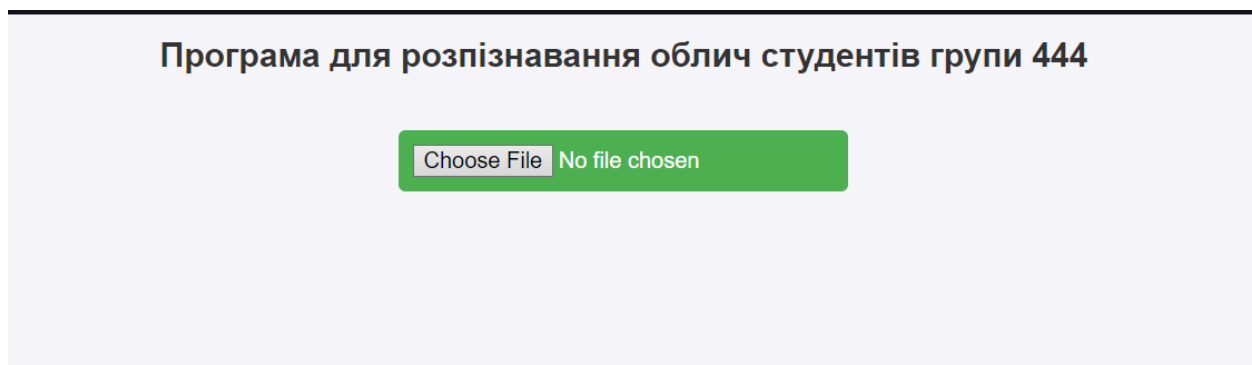


Рисунок 14 – Початковий екран запусненої програми



Почекавши хвилину-дві ми бачимо ось таке повідомлення:

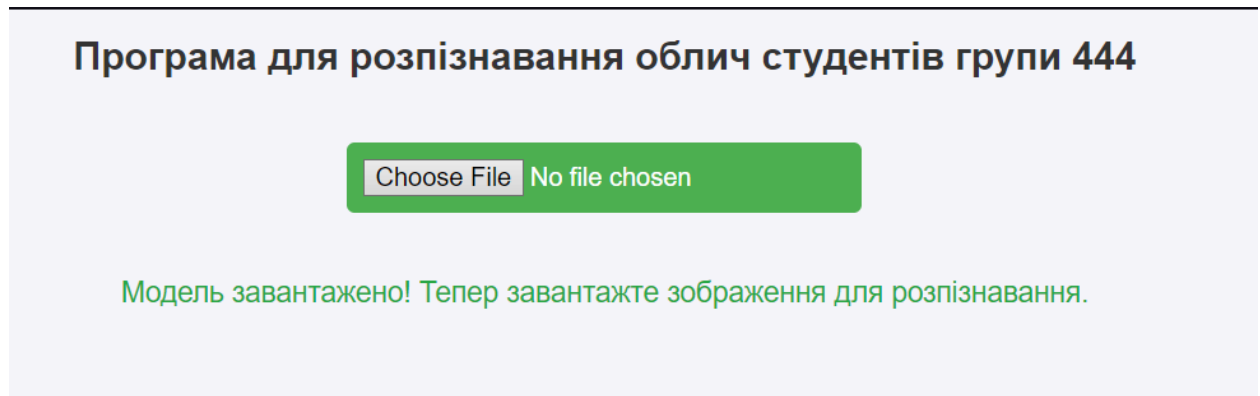


Рисунок 15 – Повідомлення про успішне завантаження моделі і зображень з Github

Давайте відкриємо консоль щоб зрозуміти що відбулось:

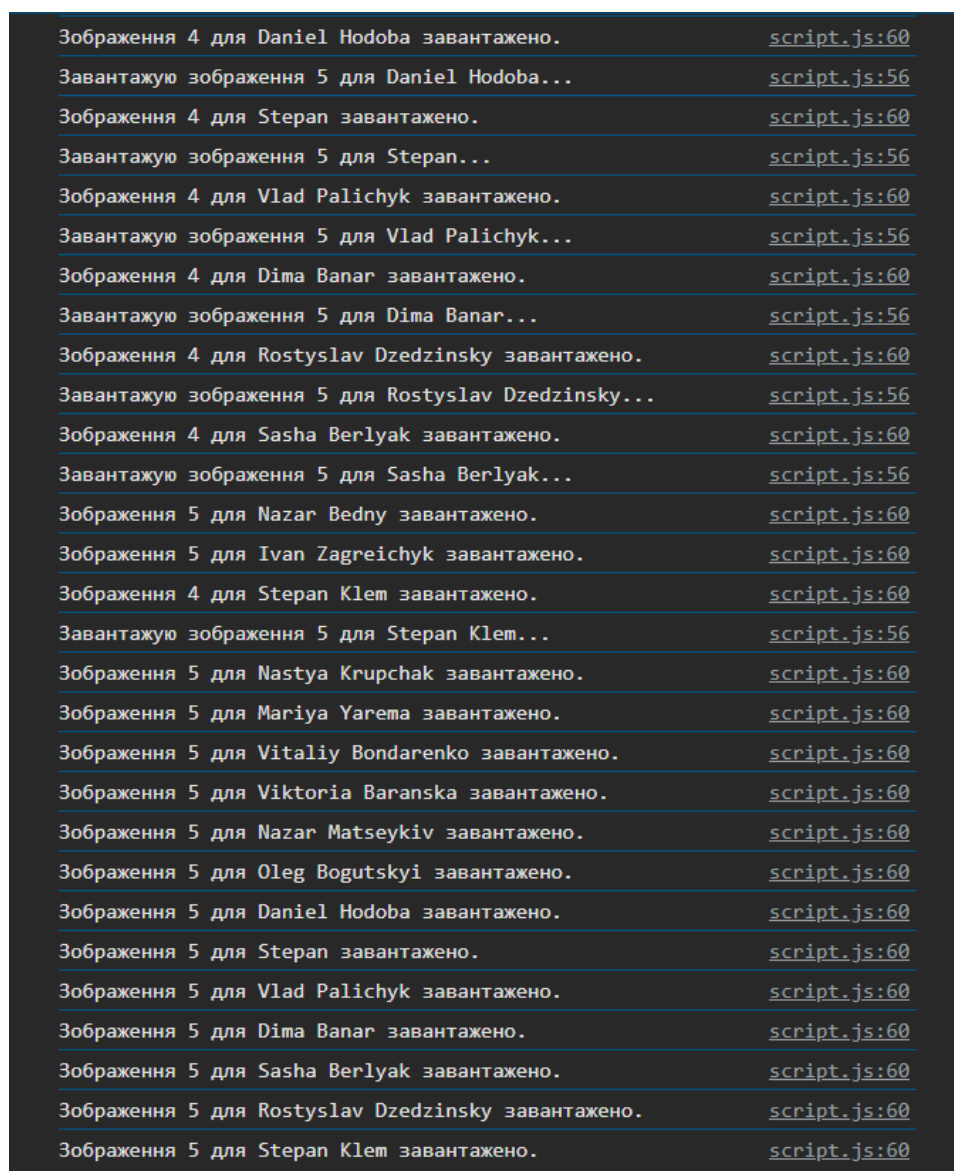


Рисунок 16 – Вивід консолі.

Цей вивід консолі свідчить про успішне завантаження зображень з репозиторію на Github. Також вивід в консоль дуже зручний для виправлення помилок, тому що я можу бачити які саме зображення не завантажились і чому.

Тепер завантажимо якесь зображення для розпізнавання:

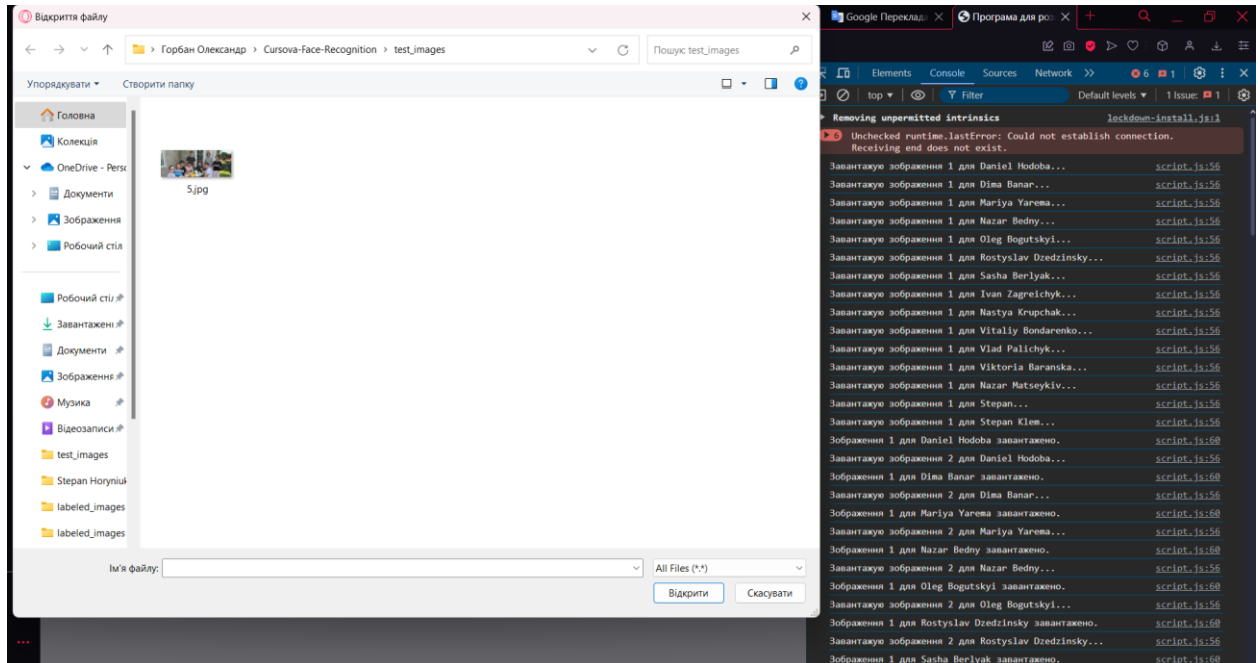


Рисунок 17 – Завантаження тестового зображення з папки test\_images

Програма аналізує зображення після чого виводить всі обличчя які знайшла, ті обличчя які не змогла ідентифікувати записує як unknown, а тих кого змогла ідентифікувати підписує їхніми лейблами, тобто назвами папок де є схожі зображення

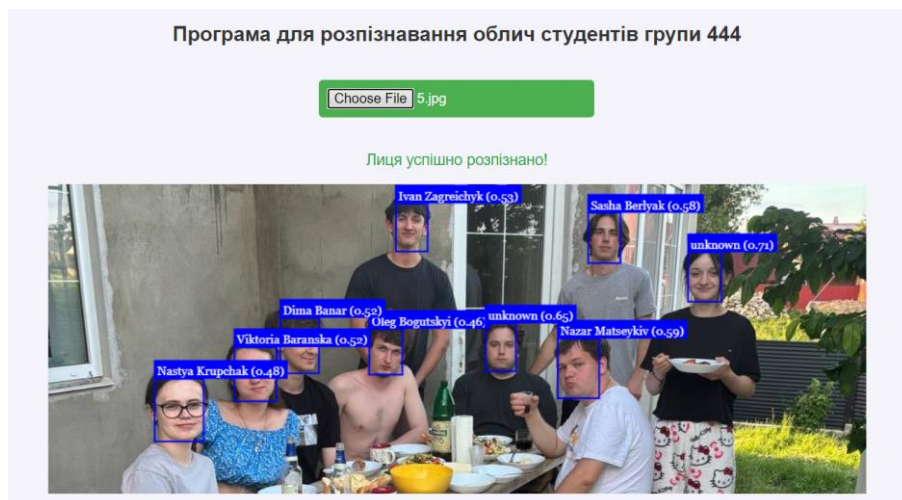


Рисунок 18 – Результат аналізу програми.

По зображенні можемо побачити що програма розпізнала Nastya Krupchak, Viktoria Baranska, Dima Banar, Oleg Bogutskyi, Ivan Zagreichyk, Sasha Berlyak. Олексія та дівчину визначило як unknown тому що їх немає в папці з зображеннями, а Сергія визначило як Nazar Matseykiv, хоча мало б визначити як unknown тому що його фотографій немає в папці.

Спробуємо інше фото, побачимо які будуть результати:

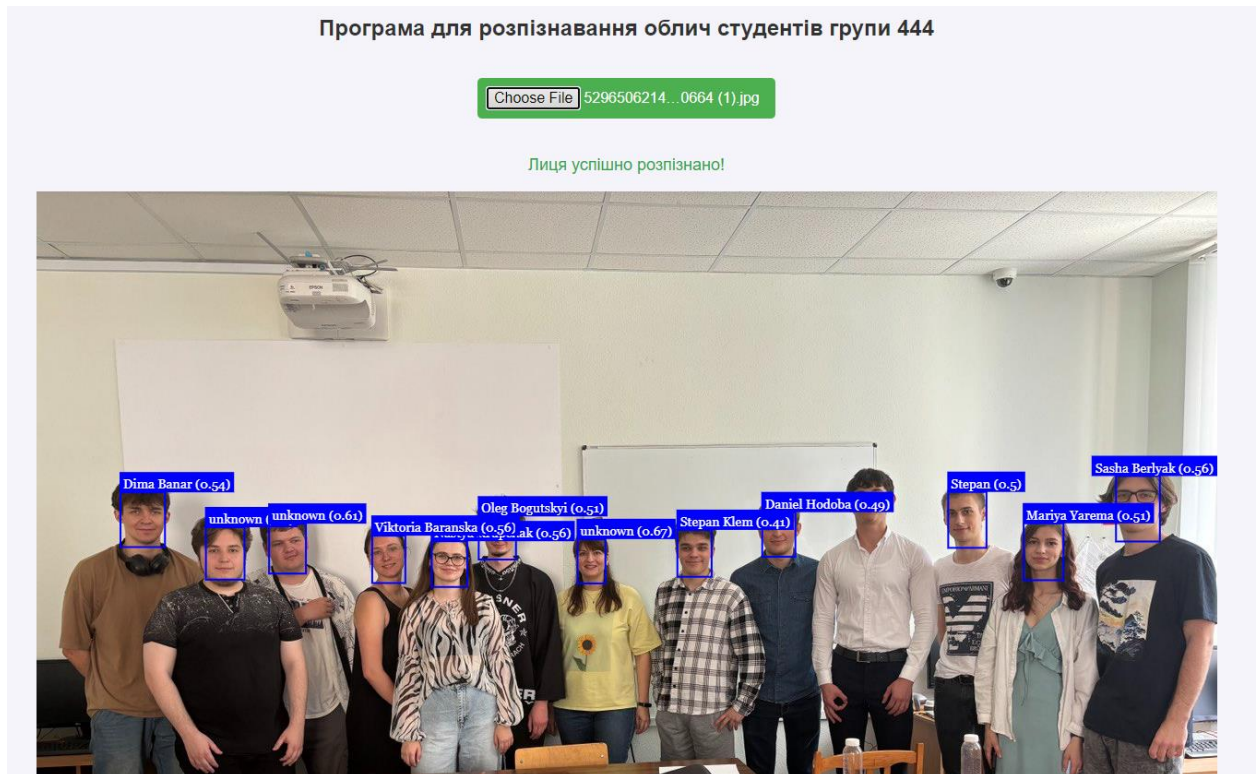


Рисунок 19 – Результат аналізу програми

З виводу бачимо що програма розпізнала всіх коректно крім Івана, тобто загальна успішність визначення **12/13**, тобто **92,3%**, роботою програми задоволений.

## ВИСНОВКИ

У цій курсовій роботі було детально описано процес створення веб-сервісу для розпізнавання облич за допомогою технологій JavaScript та бібліотеки face-api.js. Я розглянув основні теоретичні аспекти використання штучного інтелекту в розпізнаванні облич, а також пояснив, як створено систему для обробки та аналізу зображень з використанням цього підходу.

Основна мета моєї роботи полягала в реалізації веб-додатку, який здатний здійснювати розпізнавання облич студентів на завантажених зображеннях. Я використав бібліотеку face-api.js для інтеграції алгоритмів розпізнавання облич і обробки зображень, що дозволило створити інтуїтивно зрозумілий інтерфейс для користувачів.

У процесі виконання цієї роботи я здобув нові навички роботи з бібліотеками для машинного навчання на JavaScript, а також навчився налаштовувати веб-сервіс для роботи з моделюванням та обробкою зображень. Це дозволило не лише реалізувати розпізнавання облич, але й створити систему для порівняння збережених образів з новими зображеннями, забезпечивши високу точність визначення.

У першій частині роботи я розглянув теоретичні основи машинного навчання та алгоритмів розпізнавання облич, що використовуються у веб-сервісах. У другому розділі описано технології та інструменти, які я використав під час розробки веб-сервісу: face-api.js, JavaScript, HTML, а також налаштування серверної частини для обробки зображень.

Було розроблено основні функції програми, зокрема: завантаження зображень, їх обробка, порівняння з уже збереженими образами, а також виведення результатів на веб-сторінці. Важливою частиною роботи була розробка ефективного інтерфейсу, що дозволяє користувачам завантажувати зображення для аналізу і отримувати результати розпізнавання.

В результаті виконання цієї курсової роботи я розширив свої знання в області розпізнавання облич, а також здобув досвід у створенні веб-додатків з інтегрованими технологіями машинного навчання. Цей проект є прикладом

використання сучасних інструментів для створення інтелектуальних веб-сервісів.

Загалом, виконання цієї роботи дозволило мені поглибити знання в області веб-розробки та машинного навчання, а також вдосконалити навички роботи з JavaScript та бібліотеками для обробки зображень

В результаті написання курсової роботи виконані такі задачі:

1. Розробка веб-сервісу для розпізнавання облич з використанням бібліотеки face-api.js та технологій JavaScript, HTML.
2. Створення функціоналу для завантаження та обробки зображень для розпізнавання облич.
3. Реалізація порівняння облич з для визначення схожості з уже збереженими образами.
4. Відображення результатів розпізнавання облич на веб-сторінці із зазначенням ідентифікованих осіб.
5. Підготовка інтерфейсу для завантаження зображень та перегляду результатів розпізнавання.
6. Повідомлення користувача про успішне завантаження зображень та обробку, а також про невдачу при ідентифікації облич.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Сучасний підручник з JavaScript URL: <https://uk.javascript.info/> (дата звернення: 13.10.2024).
2. Основи JavaScript URL: <https://uk.javascript.info/first-steps> (дата звернення: 15.10.2024).
3. Распознавание лиц и силуэтов людей, автомобилей и номерных знаков | NtechLab. Платформа мультиобъектной видеоаналитики. URL: <https://findface.pro> (дата звернення: 16.10.2024).
4. FaceApp: Face Editor. *FaceApp: Face Editor*. URL: <https://www.faceapp.com> (дата звернення: 17.10.2024).
5. Reface – AI Face Swap App & Video Face Swaps. *Reface – AI Face Swap App & Video Face Swaps*. URL: <https://www.reface.app> (дата звернення: 18.10.2024).
6. ipynbs/face\_detection.ipynb at masterjustadudewhohacks/ipynbs. *GitHub*. URL: [https://github.com/justadudewhohacks/ipynbs/blob/master/face\\_detection.ipynb](https://github.com/justadudewhohacks/ipynbs/blob/master/face_detection.ipynb) (дата звернення: 22.10.2024)
7. GitHub - justadudewhohacks/face-api.js: JavaScript API for face detection and face recognition in the browser and nodejs with tensorflow.js. *GitHub*. URL: <https://github.com/justadudewhohacks/face-api.js> (дата звернення: 22.10.2024).
8. face-api.js. Documentatio URL: <https://justadudewhohacks.github.io/face-api.js/docs/globals.html> (дата звернення: 29.10.2024).

## ДОДАТКИ

### Додаток А

#### Index.html

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <script defer src="face-api.min.js"></script>
  <script defer src="script.js"></script>
  <title>Програма для розпізнавання облич студентів групи 444</title>
  <style>
    body {
      margin: 0;
      padding: 0;
      display: flex;
      justify-content: center;
      align-items: center;
      flex-direction: column;
      font-family: Arial, sans-serif;
      background-color: #f4f4f9;
    }

    h1 {
      margin-bottom: 20px;
      font-size: 24px;
      color: #333;
    }
```

```
#imageUpload {  
  margin: 20px;  
  padding: 10px;  
  font-size: 16px;  
  background-color: #4CAF50;  
  color: white;  
  border: none;  
  border-radius: 5px;  
  cursor: pointer;  
  transition: background-color 0.3s;  
}
```

```
#imageUpload:hover {  
  background-color: #45a049;  
}
```

```
#statusMessage {  
  margin-top: 20px;  
  font-size: 18px;  
  color: #28a745;  
}
```

```
.file-upload-container {  
  display: flex;  
  flex-direction: column;  
  align-items: center;  
}
```

```
#container {  
  position: relative;
```



```
    max-width: 100%;
}

canvas {
    position: absolute;
    top: 0;
    left: 0;
    pointer-events: none;
}
</style>
</head>
<body>
    <div class="file-upload-container">
        <h1>Програма для розпізнавання облич студентів групи 444</h1>
        <input type="file" id="imageUpload">
        <p id="statusMessage"></p>
    </div>
    <div id="container"></div>
</body>
</html>
```

## script.js

```
const imageUpload = document.getElementById('imageUpload')
const statusMessage = document.getElementById('statusMessage')
const container = document.getElementById('container')

Promise.all([
  faceapi.nets.faceRecognitionNet.loadFromUri('/models'),
  faceapi.nets.faceLandmark68Net.loadFromUri('/models'),
  faceapi.nets.ssdMobilenetv1.loadFromUri('/models')
]).then(start)

async function start() {
  const labeledFaceDescriptors = await loadLabeledImages()
  const faceMatcher = new faceapi.FaceMatcher(labeledFaceDescriptors, 0.6)

  let image
  let canvas

  // Встановлюємо повідомлення про успішне завантаження моделей
  statusMessage.textContent = 'Модель завантажено! Тепер завантажте зображення для розпізнавання.'

  imageUpload.addEventListener('change', async () => {
    statusMessage.textContent = '' // Очищаємо повідомлення, коли починається завантаження зображення

    if (image) image.remove()
    if (canvas) canvas.remove()
```

```

image = await faceapi.bufferToImage(imageUpload.files[0])
image.style.maxWidth = '100%'
container.append(image)
canvas = faceapi.createCanvasFromMedia(image)
container.append(canvas)

const displaySize = { width: image.width, height: image.height }
faceapi.matchDimensions(canvas, displaySize)

const detections = await
faceapi.detectAllFaces(image).withFaceLandmarks().withFaceDescriptors()

const resizedDetections = faceapi.resizeResults(detections, displaySize)

const results = resizedDetections.map(d =>
faceMatcher.findBestMatch(d.descriptor))

results.forEach((result, i) => {
  const box = resizedDetections[i].detection.box
  const drawBox = new faceapi.draw.DrawBox(box, { label: result.toString() })
  drawBox.draw(canvas)
})

// Виводимо повідомлення про успішну обробку зображення
statusMessage.textContent = 'Лиця успішно розпізнано!'
})
}

function loadLabeledImages() {

```

```

    const labels = ['Daniel Hodoba', 'Dima Banar', 'Mariya Yarema', 'Nazar
Bedny', 'Oleg Bogutskyi', 'Rostyslav Dzedzinsky', 'Sasha Berlyak', 'Ivan
Zagreichyk', 'Nastya Krupchak', 'Vitaliy Bondarenko', 'Vlad Palichyk',
'Viktoria Baranska', 'Nazar Matseykiv', 'Stepan', 'Stepan Klem']

    return Promise.all(

        labels.map(async label => {

            const descriptions = []

            for (let i = 1; i <= 5; i++) {

                console.log(`Завантажую зображення ${i} для ${label}...`)

                const img = await
faceapi.fetchImage(`https://raw.githubusercontent.com/OleksandrHorban/Curs
ova-Face-Recognition/main/labeled_images/${label}/${i}.jpg`)

                const detections = await
faceapi.detectSingleFace(img).withFaceLandmarks().withFaceDescriptor()

                descriptions.push(detections.descriptor)

                console.log(`Зображення ${i} для ${label} завантажено.`)

            }

            return new faceapi.LabeledFaceDescriptors(label, descriptions)

        })

    )
}

```