Вінницький Національний Технічний Університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: Розробка системи розпізнавання позицій людини на основі нейронної мережі

Виконав: студент IV курсу, групи 2ПІ-20б

спеціальності

121 – Інженерія програмного забезпечення

(шифр і назва напряму підготовки, спеціальності)

Яцик Дмитро Ігорович

(прізвище та ініціали)

Керівник: к.т.н., доцент каф. ПЗ Ткаченко О.М.

(прізвище та ініціали)

Рецензент: к.т.н., доцент каф. ОТ Богомолов С.В

(прізвище та ініціали)

Допущено до захисту

Зав. кафедри _____

«10» червня 2024 р.

ВНТУ – 2024

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 – «Інформаційні технології»
Спеціальність 121 – «Інженерія програмного забезпечення»
Освітньо-професійна програма – «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
д.т.н., професор
О. Н. Романюк _____
«12» березня 2024 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Яцику Дмитру Ігоровичу

1. Тема роботи – «Розробка системи розпізнавання позицій людини на основі нейронної мережі». Керівник роботи: Ткаченко Олександр Миколайович, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від 11 березня 2024 р. № 80.
2. Строк подання студентом роботи 3 червня 2024 року.
3. Вихідні дані до роботи: середовище програмної розробки – Sublime Text 3; мова програмування – Python; фреймворки – Flask, Bootstrap; допоміжні технології – HTML, CSS, JS; персональний ПК з ОС Windows 10; відео з позиціями людини в йозі.
4. Зміст текстової частини: вступ; обґрунтування вибору методу розробки та постановка задач дослідження; розробка алгоритмів та структури програмного продукту; розробка модулів програми; тестування програми, висновки; список використаної літератури; додатки.
5. Перелік ілюстративного матеріалу: титульний слайд: автор, науковий керівник; мета, предмет та об'єкт дослідження; актуальність розробки; аналіз аналогів; наукова новизна; розробка структури інтерфейсу; uml-діаграма класів моделі нейронної мережі; діаграма діяльності застосунку; блок-схема обробки відео; використані технології; тестування системи; апробація та публікації; результати роботи, фінальний слайд.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-4	Ткаченко О.М., к.т.н., доцент каф. ПЗ	13.03.24	03.06.24

7. Дата видачі завдання 13 березня 2024 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз проблеми, обгрунтування актуальності розробки системи та постановка задачі	14.03.24 – 22.03.24	вик.
2	Розробка структури та алгоритмів програмного продукту	13.03.24 – 22.03.24	вик.
3	Аналіз та вибір середовища програмування	25.03.24 – 19.04.24	вик.
4	Розробка модулів програмного продукту	22.04.24 – 10.05.24	вик.
5	Тестування програми	13.05.24 – 24.05.24	вик.
6	Оформлення матеріалів до захисту БДР	25.05.24 – 30.05.24	вик.

Студент _____ **Д. І. Яцик**
(підпис) (ініціали та прізвище)

Керівник бакалаврської дипломної роботи _____ **О. М. Ткаченко**
(підпис) (ініціали та прізвище)

АНОТАЦІЯ

УДК 00492

Яцик Д.І Розробка системи розпізнавання позиції людини на основі нейромережі: бакалаврська дипломна робота зі спеціальності 121 – інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2024. 77 с.

На укр. мові. Бібліогр. : 31 назв ; рис. : 38; табл. 3.

У бакалаврській дипломній роботі було розглянуто та проаналізовано застосунки у сфері розпізнавання людини. Робота включала аналіз об'єкту, предмету, завдань та методів дослідження. Основною метою роботи стала – розробка системи розпізнавання позиції людини на основі нейронної мережі Програмний засіб має стати надійним інструментом для розпізнавання різних позицій та забезпечити швидке та точне реагування на рухи користувачів.

В межах роботи було створено алгоритми та блок-схеми для основних операцій, також було створено макет інтерфейсу користувача.

Систему було розроблено за допомогою мови програмування Python. Для серверної та клієнтської частини було використано середовище розробки Sublime Text 3, бібліотека машинного навчання TensorFlow, Keras, Mediapipe, бібліотеки для роботи з даними Pandas, NumPy, допоміжний фреймворк Flask. Сумарний програмний продукт відповідає поставленим завданням та меті.

Ключові слова: нейронні мережі, розпізнавання позиції людини, глибоке навчання.

ABSTRACT

UDC 00492

Yatsyk D.I. Development of a system for recognizing human position based on a neural network: bachelor's thesis on the specialty 121 – Software Engineering, educational program – Software Engineering. Vinnytsia: VNTU, 2024. 77 p.

In Ukrainian. Bibliography: 31 titles; Figures: 38; Table 3.

Applications in the field of human recognition were considered and analyzed in the bachelor thesis. The work included an analysis of the object, subject, tasks and research methods. The main goal of the work was the development of a human position recognition system based on a neural network. The software should become a reliable tool for recognizing various positions and provide a quick and accurate response to user movements.

As part of the work, algorithms and flowcharts for basic operations were created, and a user interface layout was also created.

The system was developed using the Python programming language. The Sublime Text 3 development environment, the TensorFlow machine learning library, Keras, Mediapipe, libraries for working with Pandas data, NumPy, and the Flask auxiliary framework were used for the server and client parts. The total software product corresponds to the set tasks and purpose.

Keywords: neural networks, human position recognition, deep learning.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. За останні роки зросла популярність технологій пов’язаних із нейронними мережами. Ця технологія надає можливість відстеження рухів людини за допомогою відео, використовуючи штучний інтелект. Зі зростанням популярності онлайн-комунікацій та віддалених тренувань, відображення позицій людини стає дедалі важливішою функцією для застосунків.

Розробка програмних складових для застосунків з розпізнавання позицій людини на основі нейронної мережі є складним завданням, що вимагає поєднання декількох етапів розробки. Застосунок повинен ефективно опрацьовувати дані з відео, аналізувати позиції та взаємодіяти з користувачем.

З розвитком онлайн–комунікацій та зростаючою потребою у віддалених тренуваннях стала популярною тенденція розробки застосунків для розпізнавання позицій людини на основі нейронної мережі. Це породило попит на програмні системи, які можуть допомогти в реалізації таких веб-застосунків..

Існуючі веб-застосунки для розпізнавання позицій людини на основі нейронної мережі можуть бути не достатньо потужними або не задовольняти усіх потреб користувачів. Тому актуальною є розробка власних застосунків з використанням цієї технології.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета і завдання дослідження. *Метою даного дослідження* є підвищення ймовірності правильного розпізнавання позицій людини на основі нейронної мережі та забезпечення зручної взаємодії з користувачем через застосунок. Робота передбачає розробку різних частин застосунку, таких модуль обробки

даних відео, модуль розпізнавання позицій людини та частина користувацького інтерфейсу, а також інші.

У дослідженні основними задачами є:

- обґрунтування вибору методу розробки та постановка задачі дослідження;
- розробка структури та алгоритмів програмного продукту;
- розробка програмних модулів реалізації системи розпізнавання позицій людини на основі нейронної мережі;
- тестування системи.

Об'єкт дослідження – процес розпізнавання позиції людини по даних з відеопотоку.

Предмет дослідження – методи та засоби розпізнавання позицій людини на основі нейронної мережі.

Методи дослідження. Під час роботи було використано: методи штучного інтелекту для розробки нейронної мережі, методи теорії алгоритмів для розробки алгоритмів розпізнавання, методи проектування програмного забезпечення для проектування реалізації застосунку мовою Python з використанням мікро веб-фреймворку Flask.

Новизна отриманих результатів.

1) Удосконалено модель нейронної мережі, в якій, на відміну від існуючих, введено додаткову операцію налаштування розміру вхідних даних, що дозволило підвищити швидкодію розробленої нейронної мережі.
2) Удосконалено метод обробки відео, який відрізняється від існуючих тим, що в ньому використано фреймворк комп’ютерного зору Mediapipe, що дозволило спростити роботу системи та розробку.

Практична цінність одержаних результатів. Розроблено алгоритми та програму, що реалізує розпізнавання позицій людини. Алгоритми та окремі

програмні модулі можуть бути використані розробниками для створення інших програмних систем, зокрема для розпізнавання зображень.

Особистий внесок. Усі наукові результати отримано здобувачем самостійно.

Апробація роботи. Матеріали бакалаврської дипломної роботи доповідались на міжнародній науково-практичній інтернет-конференції студентів, аспірантів, молодих науковців “МОЛОДЬ В НАУЦІ: ДОСЛІДЖЕННЯ, ПРОБЛЕМИ, ПРЕСПЕКТИВИ (МН-2024)”.

Публікації. За тематикою дослідження опубліковано 1 наукову працю у збірнику матеріалів конференції “МОЛОДЬ В НАУЦІ: ДОСЛІДЖЕННЯ, ПРОБЛЕМИ, ПРЕСПЕКТИВИ (МН-2024)”. [1].

1 ОБҐРУНТУВАННЯ ВИБОРУ МЕТОДУ РОЗРОБКИ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Аналіз стану технологій для визначення позицій людини на основі нейронної мережі

В сучасному світі, де технології проникають у всі сфери життя, визначення позицій людини стає ключовою задачею для багатьох застосувань. Це відбувається в контексті зростаючої популярності віртуальної та доповненої реальності, розвитку мобільних додатків, технологій безпеки, медичного та фітнес-моніторингу, а також в інших сферах. Розглянемо детальніше основні технології, що використовуються для визначення позицій людини, та їх застосування.

Почнемо з комп'ютерного зору, одного з найпоширеніших методів визначення позицій людини. Ця технологія базується на аналізі відео або зображень, що отримуються з веб-камер або камер мобільних пристроїв. Алгоритми комп'ютерного зору використовуються для виявлення та відстеження позиці тіла людини на зображенні. Цей метод є досить ефективним у відстеженні рухів, а також у визначенні позицій об'єктів у просторі.

Другий метод використовує сенсори, такі як акселерометри, гіроскопи, магнітометри та датчики тиску. Ці сенсори вбудовані у багато сучасних мобільних пристроїв та інші гаджети які ми носимо. Вони здатні виявляти рухи тіла або окремих частин тіла і надсилати відповідні дані для аналізу та обробки.

Третій метод використовує нейронні мережі, що є одними з найбільш потужних та ефективних інструментів для визначення позиції людини. Нейронні мережі можуть аналізувати вхідні дані, отримані від веб-камер або сенсорів, для точного визначення позиції тіла людини. Вони здатні відокремлювати тіло від фону та точно визначати його позицію, навіть у складних умовах.

Нарешті, використання алгоритмів машинного навчання дозволяє нам тренувати моделі на великих наборах даних для поліпшення точності визначення позиції та відстеження рухів. Ці алгоритми можуть розпізнавати закономірності в даних та вдосконалювати свої можливості з кожним новим набором даних.

Застосування цих технологій широке і має значний потенціал. Вони використовуються у віртуальних та доповнених реальностях для іммерсивного ігрового досвіду, в медичних пристроях для моніторингу стану пацієнтів, в безпеці для відстеження та розпізнавання осіб, а також в багатьох інших сферах.

Отже, у висновку можна сказати, що розвиток технологій для визначення позиції людини відкриває нові можливості для впровадження інноваційних продуктів та сервісів у різних галузях. Завдяки поєднанню комп'ютерного зору, сенсорів, нейронних мереж і алгоритмів машинного навчання ми можемо створювати досконалі технології, які допомагатимуть нам у повсякденному житті та сприятимуть нашому технологічному прогресу.

1.2 Порівняльний аналіз аналогів

Враховуючи те що технології на основі нейронних мереж стають все більш популярні, та застосовуються у різних сферах, важко зробити конкретний порівняльний аналіз аналогів таких систем, але враховуючи такі аспекти як точність, швидкість обробки, ефективність в умовах обмежень ресурсів та використання в різних сферах застосування можна дійти до певних висновків. До найбільш відомих рішень відносять:

- TikTok;
- Freeletics;
- Snapchat;
- Fitbod;
- Fitbit.

Прикладом використання нейроних мереж є популярний додаток TikTok (китайський соцмедійний застосунок для створення та поширення відеофайлів та онлайн-трансляцій. Застосунок дозволяє користувачам створювати музичні, танцювальні, комедійні та інші відео тривалістю до 10 хвилин, але зазвичай до 15 секунд) (рис. 1.1). TikTok [2] використовує технологію відстеження позиції тіла для створення різноманітних ефектів та фільтрів, які можна застосовувати під час зйомки відео. Наприклад, він може додати анімаційні елементи або фільтри до користувачів під час відеозйомки.

Рисунок 1.1 – Приклад роботи додатку TikTok

Інший приклад – додаток Fitbod (рис. 1.2), який використовує штучний інтелект, щоб показати користувачам, які м’язи вони тренують. Fitbod [3] – це додаток для фітнесу, який надає персоналізовані тренувальні програми. Він використовує технологію PoseNet для відстеження позицій тіла.

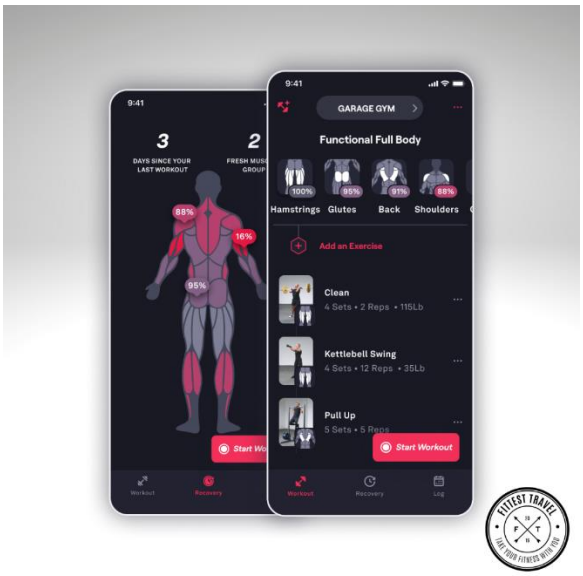

Рисунок 1.2 – Приклад роботи додатку Fitbod

Крім того, інші соціальні медіа-платформи, такі Snapchat (рис. 1.3), включили AR-фільтри та ефекти у свої додатки, на основі розпізнавання позицій з використанням нейронних мереж.

Ці фільтри та ефекти дозволяють користувачам додавати віртуальні елементи до своїх фотографій та відео, забезпечуючи цікавий та унікальний користувацький досвід.

Snapchat [4] є популярним мобільним додатком для обміну миттєвими повідомленнями, який також має функцію реалізації різноманітних фільтрів та ефектів розширеної реальності, що використовують технологію відстеження позицій обличчя та тіла. Ці фільтри можуть додавати анімаційні елементи, маски чи інші ефекти до зображень або відео, використовуючи технології відстеження позицій.

Рисунок 1.3 – Приклад роботи додатку Snapchat

Фітнес застосунок Freeletics (рис. 1.4) Freeletics [5] – це додаток для фітнесу, який пропонує тренувальні програми та тренування в режимі реального часу. Він використовує технологію OpenPose для визначення позицій тіла.

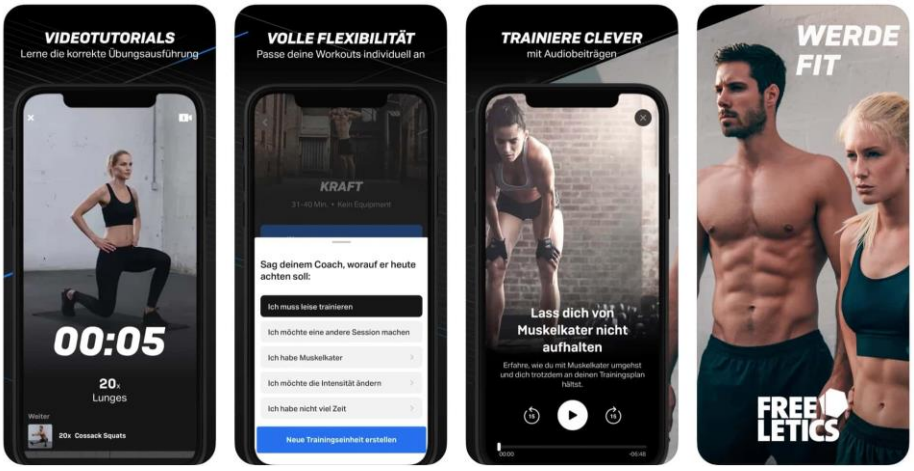

Рисунок 1.4 – Приклад роботи додатку Freeletics

Як ще один із прикладів на ринку фітнес додатків які використовують штучний інтелект є Fitbit [6].

Fitbit (рис. 1.5) – це відомий фітнес-трекер, який використовує технологію HRNet для відстеження позицій тіла та визначення виконаних вправ. Він надає користувачам зворотний зв'язок щодо їхньої активності та прогресу у фітнесі.

Рисунок 1.4 – Приклад додатку Fitbit

Результати порівняння аналогів зведено в табл. 1.1.

Таблиця 1.1 – Порівняльні характеристика онлайн платформ

Критерії	TikTok	Fitbod	Snapchat	Freeletics	Fitbit	Власний додаток
Кросплатформеність без втрати функціоналу	+	-	+	-	-	+
Автентифікація	+	+	+	+	+	+
Платформа спільноти	+	-	+	-	-	+
Відображення результатів	+	+	+	+	+	+
Розпізнавання конкретної позиції	-	-	-	-	-	+
Загальна оцінка	80%	40%	80%	40%	40%	100%

Відповідно до таблиці порівняльних характеристик розробка власної системи розпізнавання позиції на основі нейронної мережі є доцільною. Отриманий продукт зможе покрити недоліки наявних рішень та забезпечити для користувача нові враження із взаємодією з такого роду додатками.

Враховуючи складнощі та обмеження, з якими стикаються користувачі при використанні наявних рішень, було зроблено вибір розробити власний програмний продукт – веб-сервіс, спрямований на ефективне розпізнавання позиції людини. Цей веб-сервіс створений з метою допомогти користувачам вирішувати проблеми, пов'язані з точністю та надійністю розпізнавання позиції.

1.3 Аналіз методів розв’язання задачі

Існує кілька проблем, пов'язаних з розпізнаванням позицій людини, особливо в умовах складних та динамічних змін. Деякі дії та позиції людини можуть бути складними для розпізнавання, особливо коли вони відбуваються в реальному часі або при швидкому русі. Наприклад, акробатичні трюки, танці або швидкі спортивні рухи можуть становити виклик для точного розпізнавання.

Виклики, пов'язані з зовнішніми умовами, включають зміну освітлення, яка може впливати на точність розпізнавання. Наприклад, зміна освітлення в спортивному залі або на вулиці може ускладнити процес розпізнавання позицій.

Для широкого впровадження систем розпізнавання позицій у різних сферах важливо мати системи, які доступні, зручні у використанні та мають низьку складність налаштування [7]. Покращення доступності можуть розширити можливості використання цих систем.

У сучасному світі на ринку інформаційних технологій доступно декілька готових рішень, які можуть бути використані для розпізнавання та подальшої обробки поз людини. Ці рішення базуються на передових технологіях штучного

інтелекту та комп'ютерного зору. Серед наявних варіантів можна виділити чотири основні рішення, які широко використовуються в цій сфері.

Одним з цих рішень є OpenPose [8]. OpenPose – це бібліотека з відкритим кодом, розроблена для розпізнавання людських поз в реальному часі з використанням нейронних мереж. Вона була розроблена дослідницькою групою в Університеті Карнегі-Меллон і стала першою системою, яка дозволяє одночасно відстежувати кілька людей у відео чи зображеннях. Працює шляхом аналізування геометричних зв'язків між різними частинами тіла і створює точне скелетне представлення позиції людини (рис. 1.5).

OpenPose може визначати положення ключових точок на тілі людини, таких як плечі, лікті, зап'ястя, стегна, коліна та щиколотки. Всього система може розпізнати до 135 ключових точок. Окрім ключових точок тіла, OpenPose також може визначати ключові точки обличчя та рук, що дозволяє використовувати його для додаткових завдань, таких як розпізнавання жестів. Система здатна розпізнавати та відстежувати позиції кількох людей одночасно, навіть якщо вони перекриваються або знаходяться в складних позиціях.

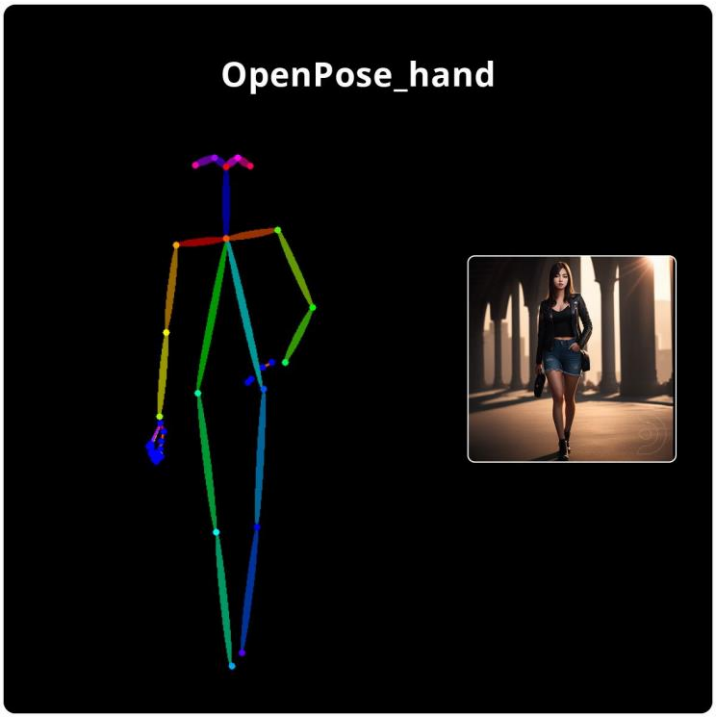

Рисунок 1.5 – Скелетне представлення людини за допомогою OpenPose

Завдяки використанню сучасних архітектур нейронних мереж та великих датасетів для навчання, OpenPose досягає високої точності у визначенні позицій.

Хоча OpenPose є потужним інструментом для розпізнавання людських поз, він має деякі недоліки та обмеження, які варто враховувати. Для досягнення реального часу обробки зображень чи відео потрібні потужні графічні процесори (GPU). Це може бути дорого для індивідуальних користувачів або малих підприємств. OpenPose може мати проблеми з точністю у випадках, коли люди знаходяться у складних позах або частково перекриті іншими об'єктами чи людьми. Освітлення та фон також можуть впливати на точність розпізнавання.

Також вагомим недоліком OpenPose – точність та ефективність OpenPose, значною мірою залежать від якості та різноманітності даних, на яких вона була навчена. У випадках специфічних завдань або нетипових даних може знадобитися додаткове донавчання моделі.

Також варто розглянути PoseNet (рис. 1.6), як ще одне готове рішення для розпізнавання позицій людини. PoseNet [9] – це нейронна мережа для оцінки поз людини, яка може ідентифікувати позиції тіла в реальному часі з використанням звичайних камер. Вона була розроблена Google і підтримується в бібліотеках TensorFlow.js та TensorFlow Lite, що дозволяє використовувати її як у веб-додатках, так і на мобільних пристроях.

Рисунок 1.6 – Приклад роботи PoseNet

Серед плюсів, завдяки простим API в TensorFlow.js та TensorFlow Lite, інтеграція PoseNet в системи є досить простою. PoseNet здатний працювати в реальному часі на звичайних камерах, що робить його ідеальним для інтерактивних додатків. Підтримує як веб, так і мобільних платформи. Код є відкритим, що дозволяє користувачам модифікувати його відповідно до своїх потреб.

Однак, як і у будь-якого рішення, у PoseNet також є свої недоліки. Хоча PoseNet добре працює для загальних застосувань, він може бути недостатньо

точним для деяких специфічних завдань, таких як медичні або спортивні аналізи. PoseNet визначає лише 17 ключових точок на тілі, що може бути недостатньо для деяких застосувань. Для роботи в реальному часі може знадобитися потужне апаратне забезпечення, особливо на мобільних пристроях.

Іншим значущим інструментом, який не можна оминути є MediaPipe [10]. MediaPipe – це крос-платформна бібліотека для побудови мультимедійних та машинного навчання пайплайнів, розроблена Google. MediaPipe містить кілька моделей, включаючи Face Mesh, Hands, Pose, Holistic та інші, які можуть працювати на різних пристроях і платформах.

Перевагою MediaPipe є його вбудована підтримка обробки в реальному часі (рис 1.7). Також підтримка Android, iOS, Python, і JavaScript, що дозволяє створювати додатки для різних платформ. MediaPipe пропонує різноманітні моделі для різних завдань, включаючи розпізнавання облич, рук, поз та інших. Завдяки оптимізації бібліотеки, MediaPipe може працювати в реальному часі навіть на пристроях із обмеженими ресурсами. Має добре документоване API та приклади, що полегшують інтеграцію в існуючі програмні засоби.

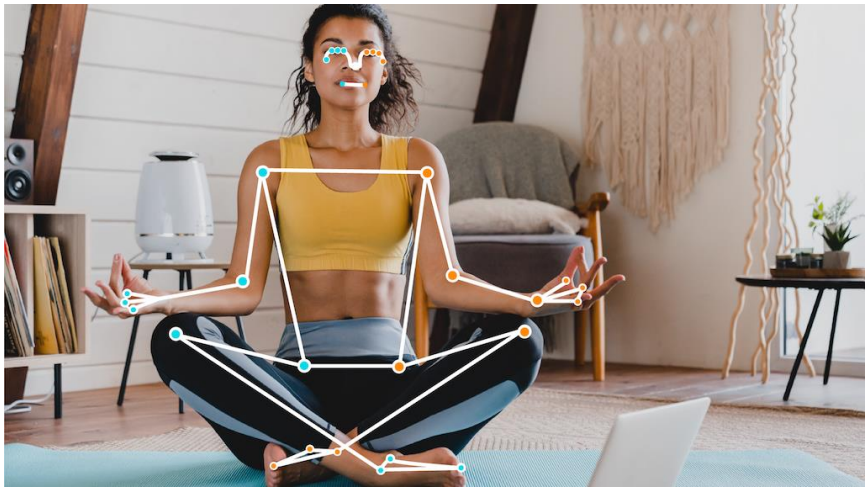

Рисунок 1.7 – Приклад зображення людини за допомогою MediaPipe

Проте, як і інші рішення, має недоліки. Порівняно з іншими бібліотеками, MediaPipe може вимагати більше зусиль для початкового налаштування та конфігурації. Для досягнення високої продуктивності може знадобитися специфічне апаратне забезпечення.

Обробка складних пайплайнів може вимагати багато ресурсів та великого обсягу даних, що може бути проблематичним для мобільних пристроїв

Як уже зрозуміло існує багато готових рішень по обробці зображень, які використовують нейронні мережі, але більш гнучким буде підхід створення моделі навченої на безпосередньо нейронній мережі, для створення такої моделі є декілька варіантів розв’язання задачі.

Convolutional Neural Networks. Це один з найбільш популярних підходів до розпізнавання образів. Зазвичай використовуються для аналізу зображень та виконання завдань класифікації, детекції об'єктів та сегментації. Для розпізнавання позиції людини CNN може використовуватися для виявлення ключових точок (наприклад, суглобів) на тілі людини та визначення їх позиції на зображенні. CNN – це клас нейронних мереж, які спеціалізуються на обробці зображень. Вони використовують конволюційні шари для ефективного виявлення локальних функцій на зображеннях.

Серед переваг CNN, це те що вона добре підходять для роботи з зображеннями завдяки використанню конволюційних шарів, має здатність автоматично вивчати корисні ознаки зображень, може бути ефективно навчена на великих об'ємах даних.

Розглядаючи недоліки, можна виділити те що може вимагати значної кількості обчислень та ресурсів, вразлива до перенавчання на обмежених або недостатньо репрезентативних даних, не завжди здатна ефективно моделювати динамічні зміни, такі як рух.

Pose Estimation Models. Існують спеціалізовані моделі, спроектовані конкретно для завдань оцінки позиції людини. Такі моделі можуть

використовувати як CNN для виявлення ключових точок, так і різноманітні архітектури, такі як Hourglass, OpenPose тощо. Вони зазвичай навчаються на великих наборах даних, які містять зображення людей з маркованими ключовими точками. Ці моделі спеціалізуються на розпізнаванні позиції людини на зображеннях. Вони зазвичай використовують ключові точки або суглоби, щоб визначити положення тіла.

Переваги даного методу в тому що він спеціалізований для завдання розпізнавання позиції людини, тому можуть бути дуже ефективними, може враховувати геометричні залежності між ключовими точками, часто досягає високої точності на відповідних даних.

Поміж недоліків можна виділити те що вимагає великих об'ємів даних для навчання, щоб досягти достатньої точності, може бути вразливим до змін у позиції, освітленні та обставинах зйомки.

Deep Learning-based Feature Extraction. Деякі підходи використовують глибокі нейронні мережі для екстракції ознак зображення, які потім використовуються для оцінки позиції. Наприклад, можна використовувати попередньо навчені моделі, такі як ResNet або VGG, для отримання векторних представлень зображення людини, які потім можуть бути подані на класифікатор для прогнозування позиції.

Цей підхід використовує попередньо навчені глибокі моделі для екстракції ознак зображення, тому серед переваг можна зазначити - готові моделі, що може значно зменшити час навчання, може досягти високої точності за умови наявності достатньої кількості даних для доналаштування, здатний до передбачення на нових зображеннях.

По недоліках – потребує великої кількості даних для доналаштування, може бути відносно витратним за обчислювальними ресурсами під час навчання.

Recurrent Neural Networks (RNNs). Іноді для розпізнавання позицій людини можна використовувати RNN або їх варіанти, такі як Long Short-Term Memory

(LSTM) або Gated Recurrent Unit (GRU). Вони можуть враховувати послідовності зображень або кадрів відео для врахування динамічної інформації про рух тіла людини. RNN – це клас нейронних мереж, які здатні моделювати послідовності даних. Вони часто використовуються для аналізу послідовностей, таких як часові ряди або послідовності кадрів відео.

Переваги – здатні моделювати динамічні зміни в часі, що робить їх ефективними для аналізу руху людини, можуть враховувати контекст з попередніх кадрів для кращого розуміння поточного стану.

Недоліки – можуть бути відносно складними для навчання та налаштування, часто вимагають значних обчислювальних ресурсів.

Combination of Multiple Models. Іноді краще використовувати комбінацію різних моделей або підходів для досягнення кращих результатів. Наприклад, можна об'єднати CNN для виявлення образів та RNN для моделювання динаміки руху людини. Цей підхід полягає в тому, що різні моделі або підходи комбінуються для досягнення кращих результатів. Може поєднувати переваги різних моделей для збільшення точності та стабільності. Дозволяє використовувати різні аспекти даного завдання, які краще моделюються різними методами. Але при тому, вимагає більше обчислювальних ресурсів і часу для розробки та навчання, а також потребує ретельного налаштування та управління для ефективної інтеграції різних моделей

Проаналізувавши плюси та мінуси кожної нейронної мережі, було вирішено що додаток буде створюватись методами з використанням архітектури CNN, яка добре підходить для обробки зображень, а також допоміжним елементом у роботі із зображеннями бути готове рішення - бібліотека MediaPipe. Це дозволить нам достатньо просто інтегруватися у розробку з перших кроків, та забезпечити цілком гнучке та точне розпізнавання позицій людини у відео, так як реакція на рухи буде максимально швидкою. Що дасть змогу створити

надійний та доволі простий по архітектурі застосунок, який буде виконувати поставлену мету.

1.4 Постановка задач розробки системи

Задача полягає у розробці системи, яка здатна автоматично аналізувати зображення або відео, визначати та відстежувати положення та орієнтацію різних частин тіла людини.

Важливим аспектом є створення інтуїтивно зрозумілого і легкого у використанні інтерфейсу для користувача.

Система повинна бути доступною навіть для непрофесійних користувачів і не повинна включати надлишкових технічних деталей, які можуть ускладнити процес використання.

Для досягнення визначеної мети було сформульовано конкретний перелік завдань, які необхідно виконати:

- провести аналіз існуючих методів та комп’ютерних технологій для розпізнавання позицій людини;
- вибрати програмні інструменти, які найкраще підходять для реалізації поставленої мети;
- провести аналіз принципів системи;
- розробити архітектуру нейронної мережі;
- навчити модель розпізнавати позицій людини;
- розробити та протестувати систему;
- розробити інтуїтивно зрозумілий інтерфейс, який дозволяє користувачу взаємодіяти з системою.

1.5 Висновки

У першому розділі було обгрунтовано вибір методу розробки та постанови задачі в розробці застосунку розпізнавання позицій. Було проведено порівняння відомих платформ які використовують нейронні мережі для визначення позицій, таких як: TikTok; Snapchat; Google maps; Fitbod; Fitbit.

У результаті порівняння було проаналізовано, що досліджувані платформи мають значну популярність серед ринку такого роду систем. Проаналізувавши переваги та недоліки відомих аналогів, було визначено, що багато із них не надають весь функціонал при мультиплатформеності. Таким чином було доведено доцільність розробки власного програмного рішення. На основі отриманої інформації було сформовано перелік задач, які необхідно виконати для розробки застосунку з розпізнавання позицій на основі нейронної мережі.

2 РОЗРОБКА СТРУКТУРИ ТА АЛГОРИТМІВ ПРОГРАМНОГО ПРОДУКТУ

2.1 Аналіз принципів системи

Для реалізації програмного застосунку з розпізнавання позицій людини на основі нейронної мережі потрібне розуміння принципів роботи застосунку, воно є ключовим елементом для успішного розроблення, підтримки та вдосконалення програмного забезпечення. Це дозволяє розробникам ефективно працювати з кодом, розуміти його поведінку та досягати високої якості продукту.

Було прийнято рішення розробки веб-системи, так як веб-застосунки можуть бути доступні з будь-якого пристрою, що має підключення до Інтернету та веб-браузеру. Це забезпечує більшу гнучкість і зручність для користувачів, оскільки вони можуть використовувати застосунок на будь-якому пристрої. Також така система може бути оновлена централізовано на сервері, що дозволяє розробникам внести зміни або виправити помилки без необхідності оновлення на кожному користувальницькому пристрої. Веб-застосунки працюють на будь-якому пристрої та операційній системі, що підтримують веб-браузер. Це дозволяє покрити більш широку аудиторію користувачів.

Також таку на такій платформі зазвичай легше розгортатись, оскільки використовуються веб-сервери та веб-хостинг. Також підтримка користувачів може бути спрощена, оскільки проблеми можна вирішувати централізовано. Є широкі можливості інтеграції - веб-застосунки легше інтегруються з іншими веб-сервісами та API, що робить їх відмінним вибором для багатьох бізнес-застосунків. А також користувачі можуть запускати веб-застосунки на будь-яких пристроях з доступом до Інтернету та веб-браузеру, незалежно від їх потужності або операційної системи.

Клієнт-серверна [11] архітектура (рис. 2.1) є однією з основних моделей розподіленої обчислювальної системи, де функціональність системи поділяється на дві основні частини: клієнти та сервери. Клієнт, який може бути користувачем або пристроєм, взаємодіє з сервером для отримання послуги або інформації. У цій архітектурі клієнти та сервери взаємодіють через мережу, таку як Інтернет або локальну мережу. Клієнти надсилають запити на сервери, а сервери відповідають на ці запити, надаючи необхідні дані або послуги. Цей процес відбувається за допомогою комунікаційного протоколу, такого як HTTP (Hypertext Transfer Protocol) для веб-систем.

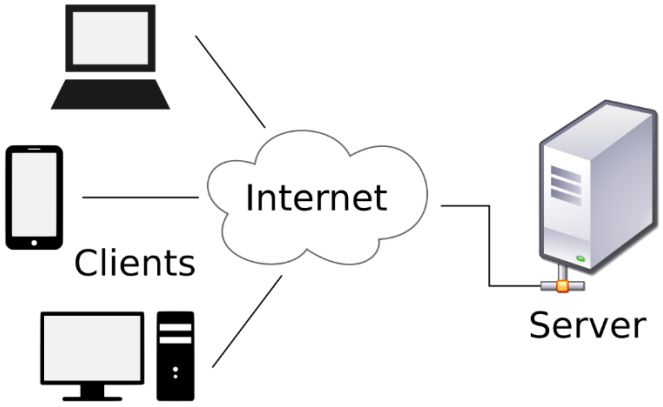

Рисунок 2.1 – Клієнт-серверна архітектура

Розробка даного застосунку з розпізнавання позицій людини на основі нейронної мережі може бути розділена на кілька основних етапів:

По-перше, на початку програми імпортуються всі необхідні бібліотеки та модулі, такі як Flask для створення веб-застосунку, pandas для роботи з даними, та інші. Також завантажується модель нейронної мережі для розпізнавання позицій за допомогою моделі, навченої глибоким машинним навчання, з технологією TensorFlow.

По-друге, це етап користувацької аутентифікації. Для аутентифікації користувача створені роути. В роутах перевіряється введений логін та пароль з

базою даних користувачів, яка зберігається у файлі. Якщо дані вірні, користувач автентифікується, інакше виводиться повідомлення про помилку. Аналогічно працює реєстрація у роуті для одноіменної дії.

Далі етап обробки завантажених відео, після завантаження відеофайлу на сервер буде відбуватися обробка відео для розпізнавання позицій людини. Ця обробка буде включати в себе використання бібліотеки mediapipe для виявлення ключових точок на людині у кожному кадрі відео. Потім буде використовуватися нейронна мережа для аналізу цих точок і визначення позицій.

Наступним етапом роботи застосунку, буде модуль візуалізації результатів. Результати роботи алгоритму відобразяться на веб-сторінці. Користувач зможе переглядати зображення базових позицій, а також порівнювати зображення з відео для кожного виявленого моменту.

На завершення у дію ввійде завантаження та зберігання результатів. Після обробки відео та отримання результатів, вони будуть зберігатися у формі ZIP-архіву. Користувач може завантажити цей архів для подальшого використання. Крім того, результати обробки будуть зберігатися у файлі, де зберігаються часові мітки та інформація про розпізнані позиції.

Отже, розробка застосунку з розпізнавання позицій людини на основі нейронної мережі включає у себе достатньо об'ємну кількість етапів, як по кількості самих модулів, так і по їх наповненню. Для роботи з таким проектом потрібна ретельна підготовка та проектування, що по закінченню, в сенергії з пристойним програмним кодом дозволить створити корисний застосунок, який користувач зможе використовувати за цільовим призначенням.

2.2 Розробка структури інтерфейсу програмного застосунку

Створити правильну структуру інтерфейсу застосунку дуже важливо, тому що це основна річ з якою взаємодіє користувач, що також в пливає на зручність

безпосередньої роботи. Компоненти програми інтерфейсу користувача (на стороні клієнта) складається з компонентів інтерфейсу користувача, також відомих як «інтерфейс», які мають відношення до кінцевих користувачів програми.

Сюди входять усі елементи, що відображаються на пристрої клієнта, включаючи інформаційну панель, зображення, тексти, меню, журнали та інші частини інтерфейсу користувача.

Це користувацько-орієнтовані компоненти, орієнтовані на кожну дію користувача під час навігації веб-програмою. Функція на стороні клієнта включає:

- Інтерфейс користувача (UI) – видима частина програми, з якою взаємодіють користувачі. Він включає HTML для структури вмісту, CSS для стилізації та JavaScript для інтерактивності.
- Веб-браузер – програмне забезпечення користувачів, яке використовується для доступу та взаємодії з веб-додатками.

При розробці структури інтерфейсу також потрібно враховувати всі потреби користувачів при майбутньому використанні. Створити ефективний роутинг між сторінками, що дозволитиь майбутньому користувачеву швидко та просто переходити по логіці веб-застосунку, між такими елементами як сторінки, меню, кнопки, посилання, кабінет, тощо.

Було визначено всі необхідні елементи структури та реалізовано, для головної сторінки системи розпізнавання позицій людини на основі нейронної мережі (рис. 2.2).

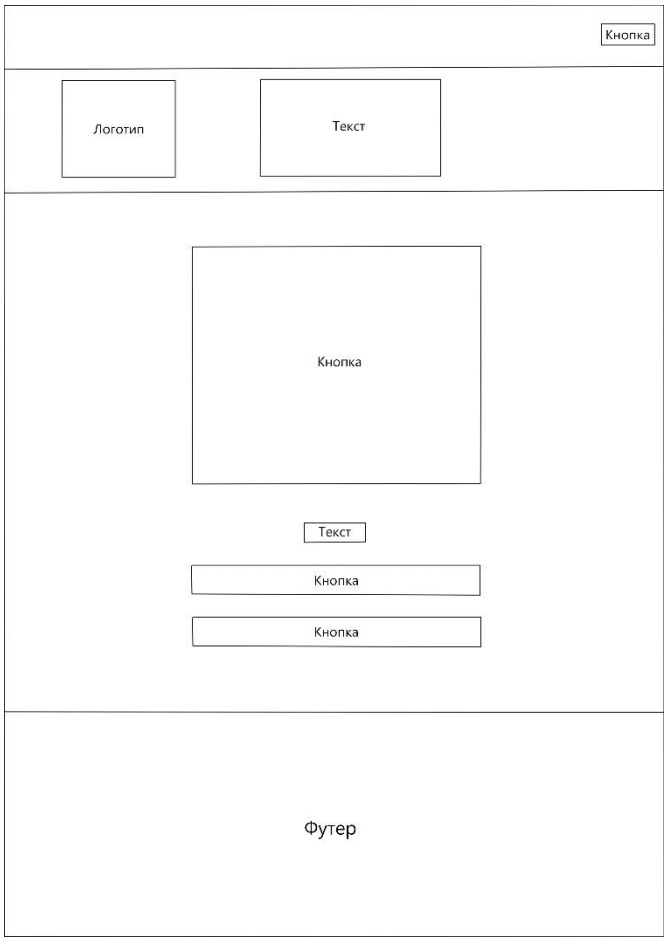

Рисунок 2.2 – Головна сторінка

Розроблена структура інтерфейсу системи з розпізнавання дає змогу користувачеві розпочати користування ресурсом без всіляких перешкод у інтуїтивній взаємодії, що робить занурення у застосунок простим та швидким.

Візуальне оформлення веб-ресурсу відіграє ключову роль у створенні привабливого та користувачам зручного інтерфейсу, це перше що кидається в очі відвідувачу. Щоб зацікавити користувача потрібно обирати відповідну кольорову палітру, яка відображатиме бренд та створить відповідний настрій. Вибравши чіткий і зручний для читання шрифт для текстового контенту, використовуючи різні розміри шрифту та стилі – це допоможе у наголошуваннях важливої інформації. А використання високоякісних зображень дозволить

привернути до себе увагу та розкрити контент системи, при тому потрібно слідкувати за розмірами файлів, для швидкого завантаження сторінок.

Отже, таким чином, було розроблено структуру для системи розпізнавання позиції людини на основі нейромережі.

2.3 Розробка моделі нейронної мережі

Для розпізнавання конкретних позицій можна скористатися навчанням моделі на зображеннях [12], що відображають ці позиції.

Оскільки розпізнавання позицій людини на зображеннях відноситься до задачі детекції об'єктів, можемо скористатися згортковими нейронними мережами (CNN), які добре підходять для обробки зображень.

Для створення моделі нейронної мережі було вирішено обрати практичний спосіб – взяти за основу архітектуру нейромережі EfficientNet. EfficientNet була розроблена дослідниками з Google Brain, використовує методику під назвою "compound scaling", що дозволяє збалансовано збільшувати глибину, ширину та роздільну здатність мережі для досягнення кращих результатів з меншою кількістю параметрів і меншою обчислювальною потужністю. Це складна архітектура на основі згорткових нейронних мереж (Convolutional Neural Networks, CNN).

Коротко розглянемо ключові шари нейронної мережі.

Згортковий шар (Convolutional Layer) – цей шар візьме вхідне зображення та виконає згортку з фільтрами для виявлення різних ознак на зображенні. Можна використовувати декілька таких шарів для покращення здатності моделі розпізнавати позиції. Кожен фільтр відповідає за виділення специфічних ознак, таких як краї, текстури або кольорові патерни.

Нормалізаційний шар використовується для стабілізації та прискорення тренування нейронної мережі шляхом нормалізації вхідних даних для кожного

шару. Це досягається шляхом центрування та масштабування даних, що допомагає зменшити вплив зміни параметрів у мережі та забезпечує більш стабільний та швидкий процес навчання.

Глибинно-згортковий шар (DepthwiseConv2D) є варіантом стандартного згорткового шару, де кожен фільтр застосовується до кожного каналу вхідного зображення окремо. Це допомагає зменшити кількість параметрів та обчислювальних витрат, зберігаючи при цьому високу якість виділення ознак.

Вихідний шар (Output Layer) – вихідний шар буде виходити з класифікаційними результатами для різних позицій. Зазвичай є останнім шаром у нейронній мережі. Його завданням є перетворення вихідних даних з попереднього шару на кінцевий результат, який може бути класом у задачі класифікації або числовим значенням у задачі регресії. Тип активації, використовуваної в цьому шарі, залежить від конкретної задачі.

Ці шари є основними [13] компонентами багатьох нейронних мереж і відіграють важливу роль у виділенні ознак, нормалізації даних, тощо.

Основні компоненти, власної моделі нейронної мережі, у даній системі включають:

1) InputLayer – початковий шар, який приймає вхідні дані розміром (300, 300, 3), що означає зображення розміром 300x300 пікселів з 3 каналами (кольорове зображення).
2) Rescaling – шар для масштабування вхідних даних. Це корисно для нормалізації пікселів зображення до діапазону [0, 1].
3) Normalization – шар для нормалізації даних, який дозволяє змінити розподіл даних до стандартного нормального розподілу.
4) Conv2D – шари згортки, які витягують ознаки зображення. Вони визначають кількість фільтрів, розміри ядер та кроки згортки.
5) BatchNormalization – шар для нормалізації виходу попереднього шару для прискорення навчання та стабілізації моделі.

6) Activation – шар активації, який застосовує нелінійні функції активації, такі як "swish".
7) DepthwiseConv2D - шари глибинної згортки, які використовуються для ефективної обробки зображень.
8) GlobalAveragePooling2D та Reshape – шари для глобального усереднення та перетворення розмірності даних.
9) Multiply – шар для множення даних з метою реалізації механізмів уваги або інших функцій.
10) Add – шар додавання, який використовується для сумування результатів різних шляхів у моделі, що корисно в резидуальних мережах.
11) Dropout – шар для регуляризації, який випадково відключає певні нейрони під час навчання для запобігання перенавчанню.

Ця конфігурація відповідає сучасним архітектурам глибинного навчання, що використовують механізми уваги та нормалізації для покращення точності та стабільності моделі.

Нейронна мережа має такий алгоритм роботи:

- Попередня обробка даних – вхідні зображення нормалізуються та масштабуються, що дозволяє мережі ефективніше працювати з даними.
- Згорткові шари. Застосовуються різні фільтри для витягування локальних характеристик зображення.
- Пакетна нормалізація. Нормалізуються виходи згорткових шарів, що прискорює процес навчання та зменшує вплив початкових значень ваг.
- Активуючі функції. Застосовується функція активації Swish, яка допомагає моделі краще узгоджуватися з складними нелінійними даними.

– Стиснення та збудження(Squeeze and Excitation). Компонент стиснення зменшує кількість каналів, а збудження масштабує важливі канали.
– Глобальне середнє згортання - узагальнює інформацію з усіх просторів на зображенні, перетворюючи її в один вектор.
– Класифікація – вектор передається на вихідний шар для остаточної класифікації позиції.

Розглянемо uml-діаграму класів моделі нейронної мережі (рис. 2.3).

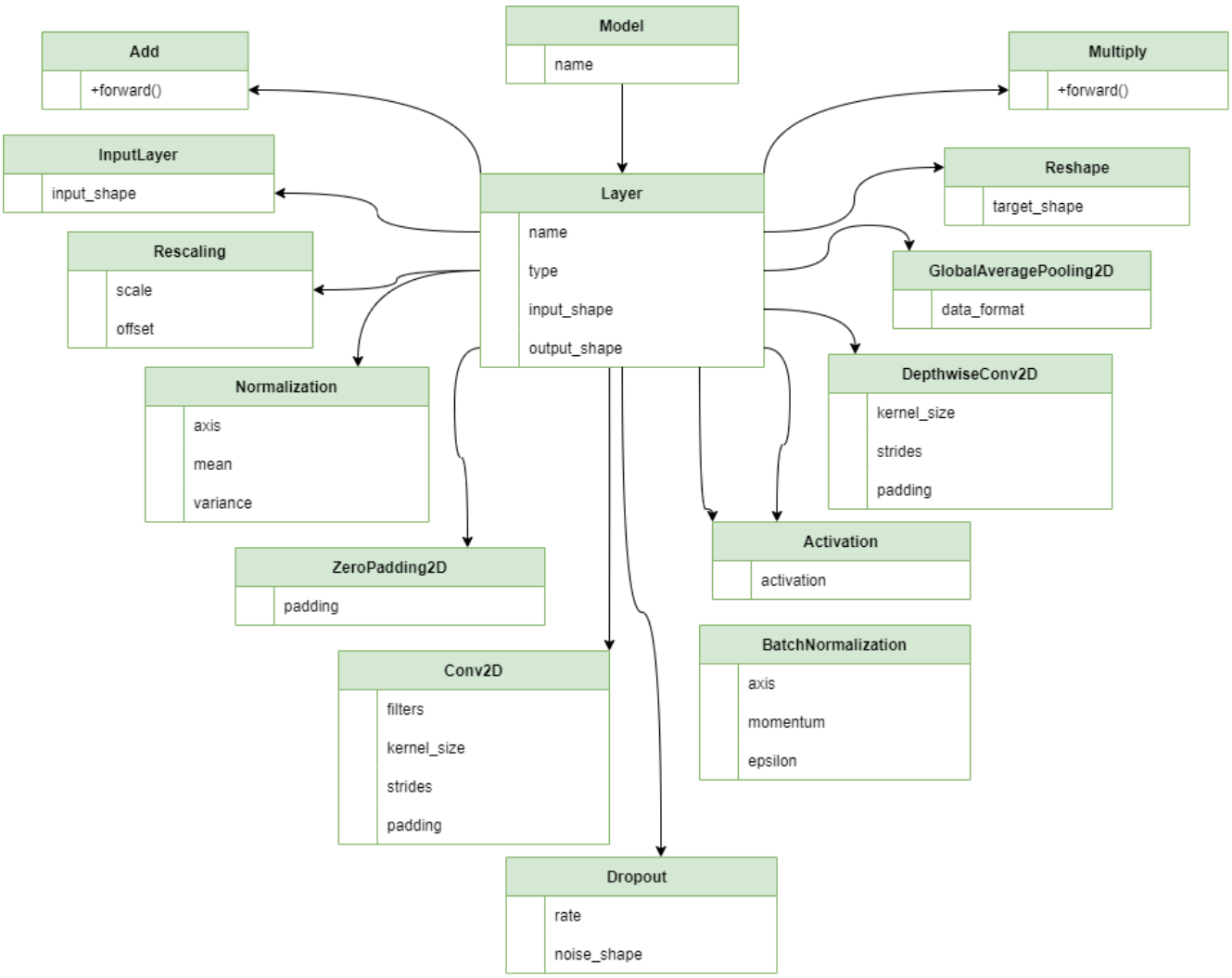

Рисунок 2.3 – UML-діаграма класів

Ця архітектура ефективно витягує і аналізує характеристики зображень, що дозволяє їй точно класифікувати різні позиції людини в йозі.

2.4 Розробка системи

Для кращого розуміння роботи застосунку із розпізнавання позицій на основі нейронної мережі було вирішено розробити модель роботи системи на прикладі UML діаграми діяльності (рис. 2.4). Згідно вказаної діаграми першим кроком користувач повинен авторизуватися в системі, якщо користувач не авторизований, або не зареєстрований, це потрібно зробити для подальшого входу у систему і роботи з нею, якщо дані введено вірно - користувач потрапить у вікно системи.

Наступним етапом потрібно завантажити відео для обробки застосунком, якщо формат відео буде некоректним, або файл буде іншого формату, тоді ми отримаємо повідомлення про відповідну помилку, якщо файл буде коректним, почнеться обробка відео системою.

При обробці відео система проаналізує його, та опрацює кожен його фрейм для розпізнавання позицій на ньому.

Коли робота буде завершеною користувач зможе завантажити отримані результати обробки відео у форматі архіву zip, або переглянути дані обробки на окремій сторінці у застосунку.

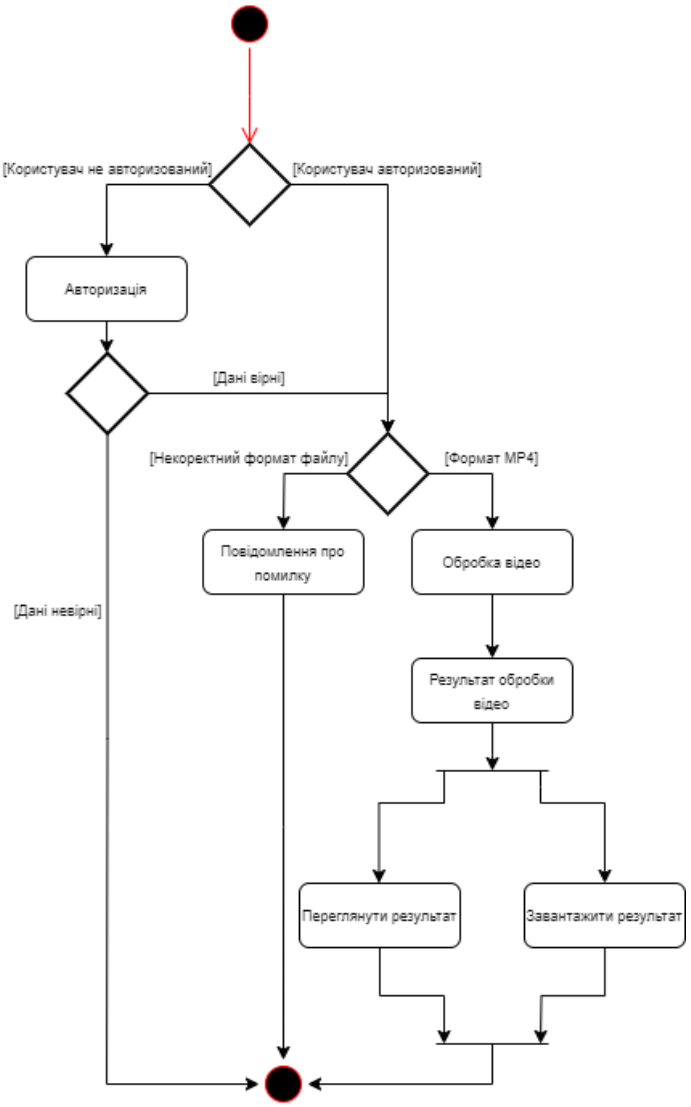

Рисунок 2.4 – Діаграма діяльності застосунку

Розробка діаграми діяльності була виконана у програмному забезпеченні по роботі із діаграмами – draw.io

2.5 Розробка алгоритму роботи системи

Для розробки застосунку, для розпізнавання позицій людини, необхідно розробити загальний алгоритм, згідно якого буде працювати система. Згідно даного алгоритму, спочатку завантажується сторінка із системою, далі

користувачу потрібно виконати обов’язкову авторизацію для забезпечення безпечної роботи. Користувач виконує авторизацію, але якщо даного користувача немає в системі, потрібно перейти до реєстрації, після чого уже авторизуватися з відповідними даними.

Після реєстрації користувач потрапляє у робоче вікно застосунку, де знаходиться основний цільовий функціонал системи. Далі користувач завантажує відео-файл у відповідне вікно системи, після чого виконується обробка даного елементу, якщо є проблеми із форматом завантаженого відео, потрібно завантажити відео з коректним форматом.

Після завантаження система обробляє відео на визначення кадрів та довжини. Далі до роботи підключаються бібліотеки із обробки позицій людини – MeadiaPipe та OpenCV. Основні кроки роботи MeadiaPipe та OpenCV включають наступні етапи: відкриття відео для обробки, обробка кадрів відео, витягування позиції з маски.

Також підвантажується модель глибокого машинного навчання на основі технології Keras, що забезпечує передбачення та визначення позиції по класифікації, а також відбувається розбиття на тайм коди.

Отримуються параметри відео: кадрова швидкість (fps) та загальна кількість кадрів. Для кожного кадру відео обробляється кадр за допомогою Mediapipe, щоб отримати результати розпізнавання позиції (pose_results) та витягується маска сегментації позиції (mask).

З маски сегментації отримуються індекси позитивних пікселів, які використовуються для вибору відповідної частини кадру.

Цей вибраний фрагмент зображення потім масштабується та підготовлюється для передачі до моделі машинного навчання. Вибраний фрагмент зображення передається до навчальної моделі для класифікації позиції. Модель повертає передбачення (prediction), яке перетворюється на назву позиції та ймовірність (probability) цього передбачення.

Якщо ймовірність передбачення вище 0.9, то зображення візуалізується, ізольоване на основі маски сегментації.

Маска сегментації – це зображення, що відображає області на зображенні, які належать до певних класів або категорій. У задачах сегментації, основна мета полягає у визначенні та виділенні окремих об'єктів або регіонів на зображенні.

Маска сегментації зазвичай представлена у вигляді зображення, де кожний піксель може мати певне значення або мітку, яке відображає клас чи категорію, до якої належить цей піксель. Вона використовується в різних доменах, таких як комп'ютерне зору, медичне зображення, робототехніка та інші.

Наступним етапом, визначені та класифіковані позиції, із розбитими на відрізки відповідно до них тайм кодами зберігаються.

Після чого користувач може переглянути результати обробки відео, або є можливість завантажити ці результати на свою робочу машину в форматі zip-архіву, який можна потім розпакувати, та також переглянути відповідну інформацію зібрану файлами.

Розглянемо модуль обробки відео більш детально, з візуальним алгоритмом його роботи (рис. 2.5).

Рисунок 2.5 – Модуль обробки відео

2.6 Висновки

У другому розділі було проведено аналіз вхідних та вихідних даних продукту розробки. Також було розглянуто покращення розпізнавання позиції за допомогою моделі машинного навчання. Було розроблено основні алгоритми роботи застосунку та алгоритм обробки відео. Також було розроблену модель роботи програмного продукту за допомогою UML діаграм, та структуру інтерфейсу.

3 РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ РЕАЛІЗАЦІЇ СИСТЕМИ РОЗПІЗНАВАННЯ ПОЗИЦІЙ ЛЮДИНИ НА ОСНОВІ НЕЙРОННОЇ МЕРЕЖІ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного забезпечення

При розробці програмного забезпечення для розпізнавання позицій людини потрібно обрати підходящі інструменти реалізації, які будуть і потужними, зможуть в повній мірі виконати реалізацію всіх задач, розробку всіх модулів застосунку та графічного інтерфейсу. Для розробки було обрано python та фреймворк flask.

Python – це високорівнева мова програмування загального призначення, яка була створена і вперше випущена ще у минулому столітті. З моменту свого випуску Python здобув величезну популярність [14] серед програмістів завдяки своїй простоті, універсальності та широким можливостям застосування.

Однією з ключових особливостей Python є його простий і зрозумілий синтаксис, який нагадує англійську мову. Це робить мову ідеальною для початківців у програмуванні, оскільки [15] вони можуть швидко освоїти основи і почати писати свої перші програми.

Python має велику кількість бібліотек та фреймворків, які охоплюють різні аспекти програмування. Для роботи з даними часто використовують такі бібліотеки, як Pandas, NumPy та Matplotlib [16]. Наприклад, Pandas дозволяє легко маніпулювати табличними даними, а NumPy забезпечує потужні засоби для роботи з багатовимірними масивами. Для розробки веб-додатків широко використовуються фреймворки Django та Flask, які забезпечують зручні інструменти для створення динамічних веб-сайтів та API.

Python активно використовується в галузі науки про дані та машинного навчання. Бібліотеки, такі як TensorFlow, Keras та PyTorch, дозволяють будувати складні моделі машинного навчання та нейронних мереж. Наприклад, TensorFlow – це потужна бібліотека, яка використовується для розробки та тренування моделей глибокого навчання.

Python є крос-платформною мовою, що означає, що програми, написані на Python, можуть виконуватись на різних операційних системах, таких як Windows, macOS та Linux. Це робить його ідеальним вибором для розробки програмного забезпечення, яке повинно працювати на різних платформах.

Python також відомий своєю великою та активною спільнотою. Це означає, що завжди можна знайти допомогу або готові рішення для різних проблем. Спільнота активно розробляє та підтримує численні бібліотеки, що дозволяє розробникам зосередитись на вирішенні конкретних завдань, а не на розробці базових інструментів.

Одним з важливих аспектів Python є його застосування в автоматизації та скриптуванні. Багато системних адміністраторів та DevOps інженерів використовують Python для автоматизації рутинних завдань, таких як управління конфігураціями, моніторинг систем та розгортання програмного забезпечення.

Python також знаходить застосування у сфері розробки ігор. Бібліотеки, такі як Pygame, забезпечують зручні засоби для створення 2D-ігор. Хоча Python не є основною мовою для розробки ігор, його простота і велика кількість бібліотек роблять його популярним вибором для прототипування та створення інді-ігор.

Завдяки своїй універсальності, простоті та великій кількості інструментів, Python залишається однією з найбільш популярних мов програмування у світі. Він дозволяє розробникам швидко створювати прототипи, тестувати і впроваджувати свої ідеї, що є важливим фактором для інновацій та розвитку в

різних галузях.Ці особливості роблять Python універсальним і потужним інструментом для розробників у різних сферах.

Flask – це мікрофреймворк для веб-розробки на Python. Незважаючи на свою легкість і мінімалістичний дизайн, Flask став [17] одним з найпопулярніших інструментів для створення веб-додатків завдяки своїй гнучкості та розширюваності. Це робить його гарним вибором для розробки системи розпізнавання позицій людини. У таблиці 3.1 візуально показано перевагу фреймворка Flask над іншими варіантами.

Таблиця 3.1 – Порівняння фреймворків для веб-розробки

Фреймворк	Flask	Django	Pyramid	Bottle	CherryPy	Falcon	Tornado
Рівень складності	Низький	Високий	Середній	Низький	Середній	Низький	Високий
Швидкодія	Висока	Середня	Середня	Висока	Висока	Висока	Висока
Спільнота	Велика	Велика	Середня	Мала	Мала	Мала	Мала
Розширюваність	Висока	Висока	Висока	Висока	Висока	Висока	Висока
Документація	Добра	Відмінна	Добра	Середня	Середня	Добра	Середня

Flask має мінімалістичний [18] підхід до розробки веб-додатків. Він не нав'язує жорсткої структури проекту чи великої кількості компонентів, що дозволяє розробникам створювати додатки за власними правилами. Основний функціонал Flask включає лише найнеобхідніші компоненти, такі як маршрутизація URL та обробка запитів.

Хоча Flask сам по собі є мінімалістичним, його можна легко розширити за допомогою численних розширень. Існує багато бібліотек, які додають функціонал для роботи з базами даних, автентифікацією, управлінням формами, обробкою запитів та багато іншого. Наприклад, розширення Flask-SQLAlchemy дозволяє інтегрувати ORM SQLAlchemy для роботи з базами даних.

Flask дозволяє розробникам використовувати будь-які бібліотеки чи інструменти на їх вибір. Це робить його ідеальним для створення прототипів та невеликих додатків, де важлива швидкість розробки та можливість швидко вносити зміни. Водночас, Flask достатньо потужний для створення складних та масштабованих додатків.

Flask часто використовують для створення [19] невеликих та середніх веб-додатків, API сервісів, а також для прототипування і тестування ідей. Він добре підходить для стартапів і невеликих команд розробників, які хочуть швидко вивести свій продукт на ринок.

Отже, Flask – це потужний і гнучкий інструмент для веб-розробки на Python, який підходить як для новачків, так і для досвідчених розробників. Його легкість, розширюваність та гнучкість роблять його ідеальним вибором для створення широкого спектру веб-додатків – від простих сайтів до складних веб-сервісів та API. Якщо ви шукаєте інструмент для швидкої та ефективної розробки веб-додатків, Flask може стати відмінним вибором.

Keras – це високорівневий API для розробки та тренування нейронних мереж, який працює на основі різних бібліотек глибокого навчання, таких як TensorFlow, Theano, та Microsoft Cognitive Toolkit (CNTK). Keras був створений та вперше випущений у 2015 році, тобто це достатньо новий інструмент. Основною метою Keras є забезпечення простого і зрозумілого інтерфейсу для швидкої розробки моделей глибокого навчання.

Keras має інтуїтивний та зрозумілий API, який дозволяє швидко створювати і тренувати нейронні мережі. Завдяки цьому Keras є відмінним вибором для новачків, які тільки починають працювати з глибоким навчанням.

Keras побудований на [20] принципах модульності. Це означає, що моделі, шари, функції активації, функції втрат та інші компоненти є незалежними модулями, які можна легко комбінувати для створення нових архітектур нейронних мереж.

Keras може працювати на основі різних бекендів, таких як TensorFlow, Theano та CNTK. Це дає розробникам можливість вибирати найбільш підходящий бекенд для їх потреб, а також легко переходити між ними.

Keras використовується в багатьох галузях, де необхідно застосовувати глибоке навчання. Розпізнавання об'єктів, класифікація зображень, сегментація зображень; розпізнавання мови, машинний переклад, аналіз тональності; прогнозування, кластеризація, виявлення аномалій; аналіз геномних даних, моделювання білків, медична діагностика.

Тому, Keras є потужним інструментом для розробки моделей глибокого навчання, який поєднує простоту, гнучкість і модульність. Він підходить як для новачків, так і для досвідчених розробників, які бажають швидко створювати та тренувати нейронні мережі для різних завдань. Завдяки підтримці кількох бекендів і великій кількості бібліотек, Keras залишається одним з найпопулярніших інструментів у галузі глибокого навчання.

Якщо мова йде про створення веб-інтерфейсу, найчастіше обирають HTML, CSS і Bootstrap. HTML (HyperText Markup Language) та CSS (Cascading Style Sheets) є базовими мовами для створення структури та стилізації веб-сторінок. Вони відіграють ключову роль у розробці [21] веб-додатків, забезпечуючи їхній візуальний та структурний вигляд.

HTML відповідає за структуру веб-сторінки. Це мова розмітки, яка використовує теги для визначення різних елементів на сторінці, таких як заголовки, абзаци, списки, таблиці, зображення та посилання. HTML дозволяє створювати ієрархію елементів та визначати їх взаємозв'язки, але не займається оформленням.

CSS відповідає за стилізацію веб-сторінок. Це мова стилів, яка визначає зовнішній вигляд елементів, їх розташування, кольори, розміри, шрифти та інші аспекти візуального оформлення. CSS дозволяє створювати стилі для конкретних елементів, використовувати класи та ідентифікатори для точного

визначення стилів, а також застосовувати стилі до груп елементів. Розділення структури та стилізації робить розробку більш модульною та легко підтримуваною.

Використовуючи HTML та CSS, можна створювати веб-інтерфейси для додатків. HTML визначає структуру сторінки, включаючи розміщення елементів, заголовки, кнопки, форми та інші інтерактивні елементи. CSS забезпечує зовнішній вигляд цих елементів, встановлюючи кольори, фони, шрифти, розміри та інші властивості стилізації.

Поєднання HTML та CSS дозволяє створювати потужні та інтерактивні веб-інтерфейси, що забезпечують ефективну взаємодію користувачів з додатками та надають приємний користувацький досвід.

Bootstrap – це популярний фреймворк для розробки веб-інтерфейсів. Він надає набір готових компонентів, шаблонів та стилів, які допомагають розробникам швидко створювати сучасні та респонсивні веб-сторінки.

Однією з головних переваг Bootstrap є його гнучкість та простота використання. Фреймворк [22] використовує систему сітки, що дозволяє організовано розташовувати елементи на сторінці. Завдяки класам Bootstrap, можна легко визначати розташування елементів на різних розмірах екрану, що забезпечує адаптивний дизайн.

Крім того, Bootstrap включає велику кількість готових компонентів, таких як навігаційні панелі, кнопки, форми, модальні вікна, картки тощо. Ці компоненти легко впроваджуються в проект, просто додаючи необхідні класи до HTML-елементів. Вони мають вже визначені стилі в Bootstrap, що дозволяє швидко створювати красивий та однорідний вигляд веб-інтерфейсу.

Bootstrap також підтримує JavaScript-компоненти, такі як модальні вікна, випадаючі списки, каруселі та інші інтерактивні елементи. Можна легко додавати інтерактивність до додатків, використовуючи готові JavaScript-компоненти Bootstrap або розширюючи їх власним кодом.

Bootstrap є популярним серед розробників завдяки прискоренню процесу створення веб-інтерфейсів, забезпеченню консистентного та привабливого вигляду сторінок, а також сприянню адаптивному дизайну. Активна спільнота розробників забезпечує підтримку, документацію та розширення для цього фреймворку.

3.2 Налаштування моделі нейронної мережі

Основні етапи розробки включають: підготовку даних, вибір та налаштування архітектури моделі, тренування, валідацію та оцінку. Модель базується на архітектурі [23]. EfficientNet – це сучасна архітектура згорткових нейронних мереж, відома своєю ефективністю та високою продуктивністю при класифікації зображень.

Підготовка даних. Було підібрано великий набір зображень, що містять різні позицій йоги, за допомогою сервіса Kaggle, дані розподілені на тренувальні та тестові. Зображення були змінені до розміру, рекомендованого для EfficientNet (300×300), дані були нормалізовані та масштабовані для покращення якості навчання. Вибір та налаштування архітектури. EfficientNet обрана за її збалансованість між точністю та ефективністю обчислень, базова модель EfficientNet була ініціалізована з попередньо навченими вагами на ImageNet, до базової моделі було додано кілька повнозв'язних шарів для класифікації позицій йоги. Налаштування гіперпараметрів – визначено гіперпараметри, такі як кількість епох, розмір міні-пакета, швидкість навчання, використовувалися техніки регуляризації, такі як Dropout, для запобігання перенавчанню.

Розподіл даних та тренування. Як уже було зазначено дані розділені на тренувальну, валідаційну та тестову вибірки, використано метод перехресної валідації для більш точної оцінки продуктивності моделі [24]. Модель було

треновано на тренувальному наборі даних, використовуючи відповідний оптимізатор та функцію втрат.

Використання готового датасету значно спрощує процес розробки моделі та дозволяє зосередитися на покращенні її продуктивності, значна частина роботи з нейронною мережею заключається саме в грамотно підібраному наборі даних.

По шарам, налаштування нейронної мережі виглядає таким чином:

- Conv2D (Згортковий шар). Фільтри: 192 (кількість згорткових фільтрів, які будуть застосовані до вхідних даних). Розмір ядра: [1, 1] (розмір кожного згорткового фільтра). Кроки: [1, 1] (кроки, з якими згортковий фільтр пересувається по вхідних даних). Відступ: same (тип відступу, що забезпечує однаковий розмір вихідних даних). Активація: linear (тип функції активації, яка використовується після згортки). Використання зсуву: false (вказує, чи використовується зсув (bias) у шарі). Ініціалізатор ядра: VarianceScaling (метод ініціалізації ваг ядра). Ініціалізатор зсуву: Zeros (метод ініціалізації зміщення).
- BatchNormalization (Нормалізаційний шар). Вісь: [3] (вісь, по якій здійснюється нормалізація). Момент: 0.99 (момент для рухомого середнього значення). Епсилон: 0.001 (маленьке значення, щоб уникнути ділення на нуль). Центр: true (якщо true, додає від'ємне зміщення β). Масштаб: true (якщо true, масштабує γ). Ініціалізатор бета: Zeros (метод ініціалізації зміщення β). Ініціалізатор гамма: Ones (метод ініціалізації масштабу γ).
- DepthwiseConv2D (Глибинно-згортковий шар). Розмір ядра: [3, 3] (розмір кожного глибинного згорткового фільтра). Кроки: [1, 1] (кроки, з якими глибинний згортковий фільтр пересувається по вхідних даних). Відступ: same (тип відступу, що забезпечує

однаковий розмір вихідних даних). Активація: linear (тип функції активації, яка використовується після згортки). Використання зсуву: false (вказує, чи використовується зсув (bias) у шарі). Ініціалізатор ядра: VarianceScaling (метод ініціалізації ваг ядра). Ініціалізатор зсуву: Zeros (метод ініціалізації зміщення).

- GlobalAveragePooling2D (Шар глобального середнього пулінгу). Формат даних: channels_last (формат даних, де канали йдуть останніми, наприклад, [height, width, channels]).
- Reshape (Шар перетворення форми). Цільова форма: [1, 1, 816] (нова форма тензора).
- Multiply (Шар множення) та Add (Шар додавання) немають специфічних налаштувань.
- Dropout (Шар відсікання). Частка відключених нейронів: 0.02307692307692308 (відсоток нейронів, які будуть випадково відключені). Форма шуму: [null, 1, 1, 1] (форма шуму для відключення нейронів).

Ці налаштування забезпечують ефективну роботу моделі, покращуючи стабільність навчання та здатність моделі до генералізації.

3.3 Розробка компоненту обробки відео

Для реалізації розробки компонента обробки відео потрібно продумати метод та підходи в роботі, а також пункти по яким буде проходити розробка.

При розробці було вирішено скористатись наступним методом:

1) Завантаження відео
   - Відео завантажується користувачем через веб-інтерфейс і зберігається в каталозі user_data/{current_user.id}/.
2) Обробка відео

- Відео відкривається за допомогою cv2.VideoCapture.
- Знімаються основні характеристики відео, такі як FPS та загальна кількість кадрів.
- Відео розбивається на кадри з певним інтервалом, визначеним FPS.

3) Аналіз
   - Кожен кадр обробляється за допомогою mediapipe для визначення людини.
   - Якщо в кадрі знаходиться людина, створюється маска сегментації.
   - Кадр, обмежений маскою сегментації, масштабовується до розміру 300x300 і передається в модель нейронної мережі для передбачення йога позицій.
   - Якщо впевненість передбачення перевищує 90%, кадр і його передбачення зберігаються для подальшої обробки.
4) Фільтрація результатів
   - Створюється датафрейм з усіма передбаченнями і часовими позначками.
   - Виконується фільтрація результатів, щоб зберегти тільки ті, де модель має високий рівень впевненості і передбачення стабільні.
5) Створення результатів
   - Результати зберігаються у вигляді архіву, який містить зображення передбачених позицій і відповідні базові зображення з еталонного датасету.
   - В архів також включається файл Timecodes.txt з часовими позначками і назвами передбачених позицій.

Розглянемо роботу з відео (рис. 3.1), а саме функцію def process.

```
def process(username):
...
cap = cv2.VideoCapture(f'user_data/{username}/video.mp4')
...
for tc,img in df[['timecode','frame']].values:
    ...
    pose_results = pose.process(img)
    mask=pose_results.segmentation_mask
    ...
    prediction = model.predict(img_to_predict, verbose=0)
    class_name = poses_list[np.argmax(prediction)]
    probability = prediction.max()
    ...
    arr.append((tc,class_name, probability))
...
```

Рисунок 3.1 – Частини коду аналізу відео

Після аналізу всіх кадрів відео, результати, які зберігаються в DataFrame (рис. 3.2) можемо переглянути далі. У цьому DataFrame кожен рядок представляє кадр відео з відповідною часовою міткою, класом позиції та її ймовірністю.

```
df2=\
pd.DataFrame(arr, columns=['tc','c','p'])\
```

Рисунок 3.2 – Збереження в df2

Отже, було розроблено метод обробки відео для розпізнавання позицій людини за допомогою нейронної мережі.

3.4 Розробка компоненту авторизації користувача в системі

Застосунок має простий компонент авторизації, який базується на використанні файлу passwords.txt для зберігання хешів паролів користувачів.

Давайте розглянемо деталі авторизації. Коли користувач намагається увійти в систему, він переходить на сторінку /login (рис. 3.4), де може ввести своє ім'я користувача та пароль.

```
@app.route('/login', methods=['GET', 'POST'])
def login():
```

Рисунок 3.4 – Роут для логіювання

При GET-запиті сторінка авторизації просто відображається. При POST-запиті сервер отримує введені користувачем дані, зчитує файл passwords.txt, перевіряє введений логін, і якщо він існує, перевіряє хеш-пароль за допомогою функції check_password_hash з бібліотеки flask_bcrypt (рис. 3.5).

```
if passwords.query(f'login.isin(["{username}"])').__len__()==1:
    if check(passwords.query(f'login.isin(["{username}"])')['hash'].iloc[0], request.form['password']):
        global user
        user = username
        return redirect(url_for('index'))
```

Рисунок 3.5 – Обробка пост запиту

Користувач може створити новий обліковий запис, введенням імені користувача та пароля на сторінці /register.

При POST-запиті сервер перевіряє, чи існує введений користувач в файлі passwords.txt. Якщо ні, пароль хешується та додається до файлу (рис. 3.6).

```
if passwords.query(f'login.isin(["{username}"])').__len__()==0:
    pd.DataFrame([username, gen(request.form['password']).decode('utf-8')], index=passwords.columns).T\
    .to_csv('passwords.txt', sep='\t',index=False,header=False, mode='a')
    global user
    user=username
    return redirect(url_for('index'))
```

Рисунок 3.6 – Логіка реєстрації

Якщо користувач вже існує, він отримує повідомлення про те, що логін вже зареєстровано.

Обробка даних користувачів під час реєстрації є важливим аспектом забезпечення безпеки в системі. Щоб підвищити безпеку зберігання даних

користувачів, особливо паролів, необхідно застосовувати надійні методи захисту. Зокрема, паролі користувачів, введені під час реєстрації, повинні зберігатися у вигляді хешів.

Це означає, що оригінальні паролі не зберігаються в базі даних. Замість цього, паролі обробляються хеш-функцією, яка перетворює їх на унікальні хеш-коди. Коли користувач вводить свій пароль під час входу до системи, цей пароль знову хешується, і отриманий хеш порівнюється зі збереженим у базі даних. Якщо хеші збігаються, користувач отримує доступ до системи.

Цей підхід додає додатковий рівень захисту, оскільки навіть адміністратори бази даних не можуть побачити реальні паролі користувачів. Якщо хеш-код пароля буде зламано або викрадено, відновити оригінальний пароль з нього неможливо.

Для підвищення безпеки також можна використовувати такі методи, як соління паролів (додавання випадкових даних до пароля перед хешуванням) та застосування більш складних алгоритмів хешування, таких як bcrypt або Argon2. Це робить процес зворотного відновлення паролів ще більш складним і забезпечує додатковий захист від атак.

Під час авторизації користувачів введений пароль обробляється хеш-функцією, і отриманий хеш порівнюється зі збереженим у базі даних. Якщо вони збігаються, користувач успішно авторизується.

Такі методи забезпечують високий рівень конфіденційності та безпеки, захищаючи паролі користувачів від несанкціонованого доступу та використання. Використання сучасних підходів до хешування та додаткових методів безпеки робить систему більш надійною і стійкою до атак.

Щоб забезпечити ще вищий рівень безпеки для паролів користувачів, рекомендується використовувати додаткові техніки: Сіль (Salt), Ітерація, Перевірка паролів, Аудит безпеки, Використання сучасних алгоритмів хешування.

Сіль. Додавання випадкового рядка, відомого як "сіль", до пароля перед хешуванням забезпечує унікальність хешу, навіть якщо два користувачі використовують однакові паролі. Сіль додає додатковий рівень складності, ускладнюючи атаки перебором.

Ітерація. Багаторазове застосування хеш-функції (ітерація) може ускладнити спроби брутфорсу, роблячи процес перевірки пароля більш тривалим. Наприклад, можна використовувати алгоритми, які підтримують високу кількість ітерацій, такі як bcrypt, scrypt або Argon2.

Використання сучасних алгоритмів хешування. Важливо використовувати сучасні і надійні алгоритми хешування, такі як SHA-256 або SHA-512, які вважаються безпечними на сьогоднішній день.

Перевірка паролів. Крім зберігання хешів паролів, важливо реалізувати ефективний механізм перевірки паролів. При введенні пароля користувачем, система повинна хешувати введений пароль з використанням того ж самого алгоритму і солі, які були використані при реєстрації. Після цього отриманий хеш порівнюється зі збереженим хешем в базі даних.

Аудит безпеки. Регулярний аудит безпеки є ключовим для виявлення та усунення потенційних вразливостей в системі. Це включає перевірку правильності реалізації хешування паролів, а також використання сучасних стандартів безпеки.

3.5 Розробка інтерфейсу програмного застосунку

Робота користувача з програмною системою включає в себе взаємодію з інтерфейсом. Користувач може почати використовувати програмну систему, відкривши веб-додаток на своєму пристрої. При запуску програми, користувач отримує доступ до інтерфейсу, який надає зручну та інтуїтивно зрозумілу навігацію.

У розробці інтерфейсу для веб-додатку системи, було приділено достатньо уваги цій справі, та вирішено зробити візуал застосунку максимально простим та зрозумілим для використання користувачем.

Основним кольором інтерфейсу обрано білий та сірий, оскільки дані кольори є нейтральними. Допоміжними кольорами обрано червоний, синій та зелений, оскільки вони добре грають на контрасті та виділяються для користувача.

При запуску застосунку, користувача зустрічає екран із обов'язковою формою автентифікації (рис. 3.7), з якою в користувача буде асоціюватися безпека при подальшій роботі із застосунком.

Система автентифікації відіграє ключову роль у будь-якому веб-додатку або веб-системі. Її простота та зрозумілість є критичними для забезпечення зручного та безпечного доступу користувачів до системи. Легкість використання системи автентифікації дозволяє користувачам швидко зрозуміти процес входу, виконати необхідні дії та без зайвих труднощів отримати доступ до системи.

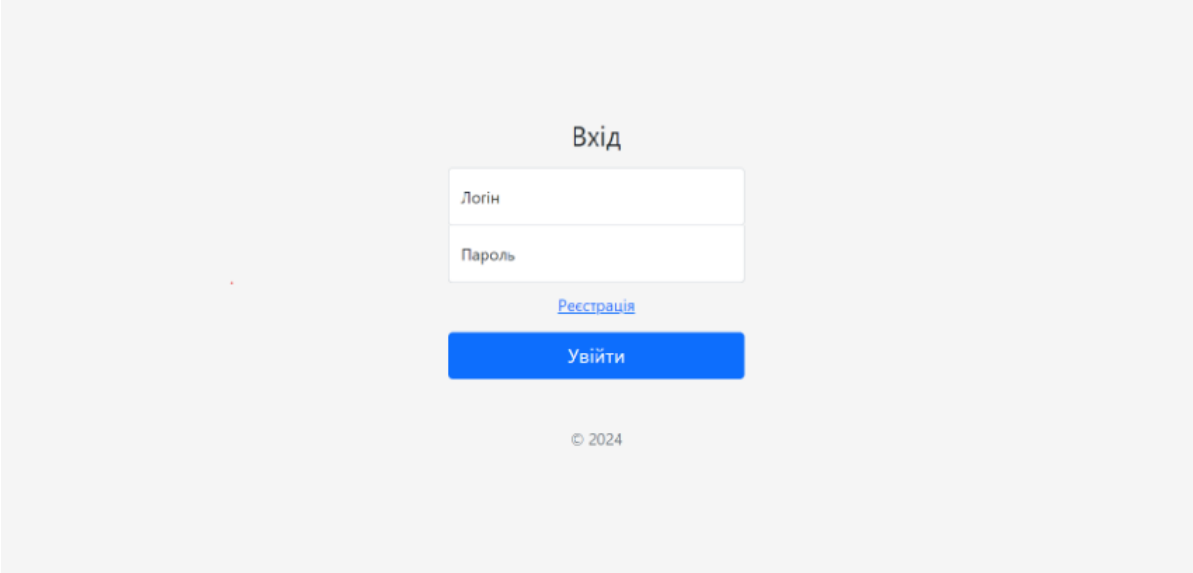

Рисунок 3.7 – Вікно автентифікації

Ця сторінка використовується для підтвердження ідентичності користувача та надання доступу до внутрішніх функцій системи. На сторінці автентифікації користувач повинен ввести свої облікові дані, такі як логін та пароль. Після введення цих даних користувач натискає кнопку «Увійти», і система перевіряє коректність введеної інформації.

Якщо дані правильні і користувач має необхідні права доступу, він буде автентифікований та перенаправлений на головну сторінку з доступом до основних функцій системи.

Якщо користувач ще не зареєстрований, він може перейти на сторінку реєстрації, де зможе заповнити необхідну інформацію для створення облікового запису. Під час реєстрації користувач має вказати логін та пароль.

Введені під час реєстрації дані перевіряються на валідність та унікальність, щоб уникнути створення дублікатів облікових записів. Таким чином, користувачі, які бажають отримати доступ до системи, можуть легко створити обліковий запис, пройшовши процедуру реєстрації.

Також для кожного користувача передбачена окрема сесія з унікальним ідентифікатором входу, що забезпечить безпеку та багатокористувацький режим.

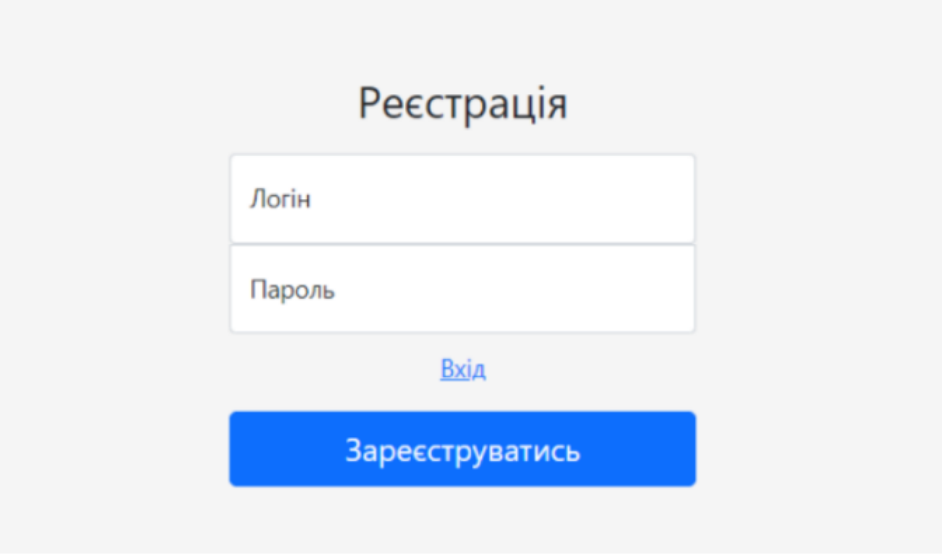

Рисунок 3.8 – Екран реєстрації користувача

На основній сторінці, (рис. 3.9), розміщена інтерактивна панель, яка надає можливість користувачеві взаємодіяти з додатком для обробки відео.

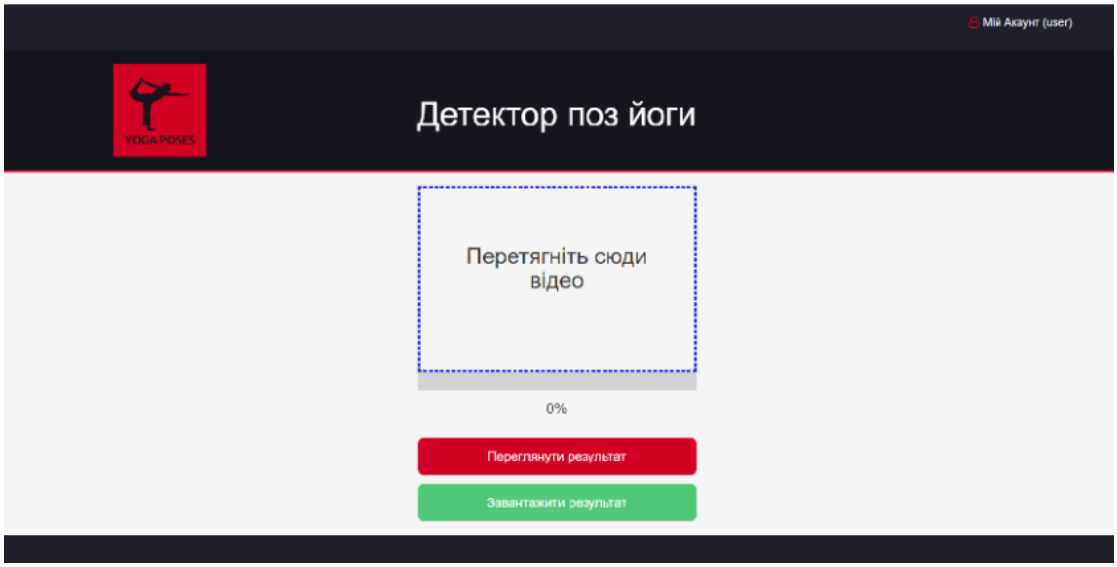

Рисунок 3.9 – Головна сторінка

На цій сторінці користувач може перетягнути відео файл у відповідну область, яка приймає завантажені файли. Після завантаження відео, система розпочинає його обробку (рис. 3.10) за допомогою методів розпізнавання позицій людини.

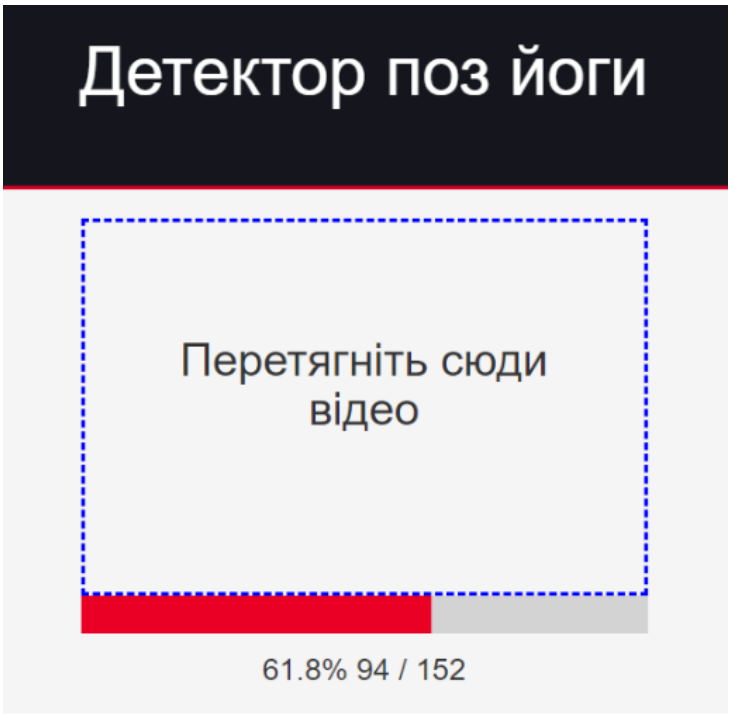

Рисунок 3.10 – Прогрес обробки відео

Прогрес обробки відео відображається на шкалі, яка показує відсотки завершення процесу. Користувач може спостерігати за зміною значень на шкалі та відстежувати прогрес обробки. Після обробки відео та завершення, користувач може переглянути результати за допомогою відповідної кнопки (рис 3.11).

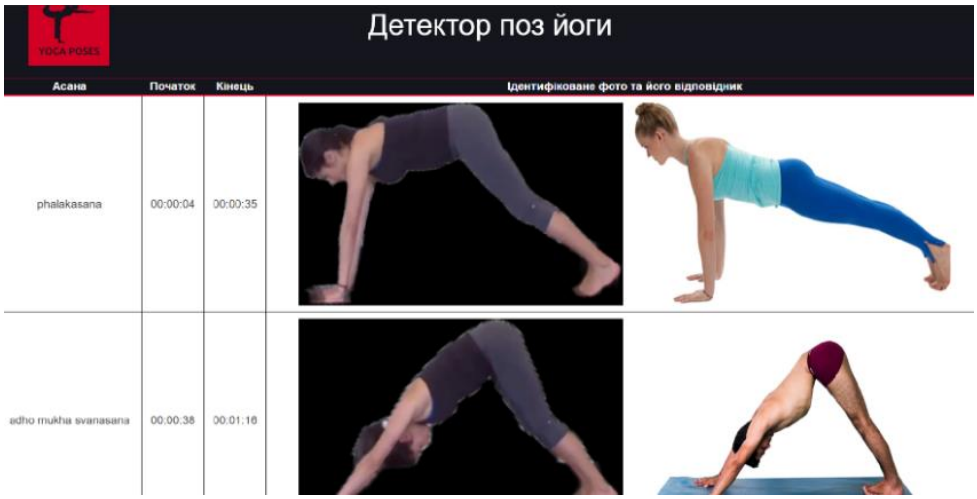

Рисунок 3.11 – Результат обробки відео

При натисканні та кнопку «Переглянути результат», нас перенесе на сторінку із результатами обробки де ми зможемо побачити список із назвами розпізнаних позицій, тайм коди цих позицій, та саму візуалізацію позиції з відео, та її відповідник з датасету.

Також користувач може завантажити ці результати у форматі zip-архіву, натиснувши на відповідну червону кнопку «Завантажити результат» (рис. 3.12)

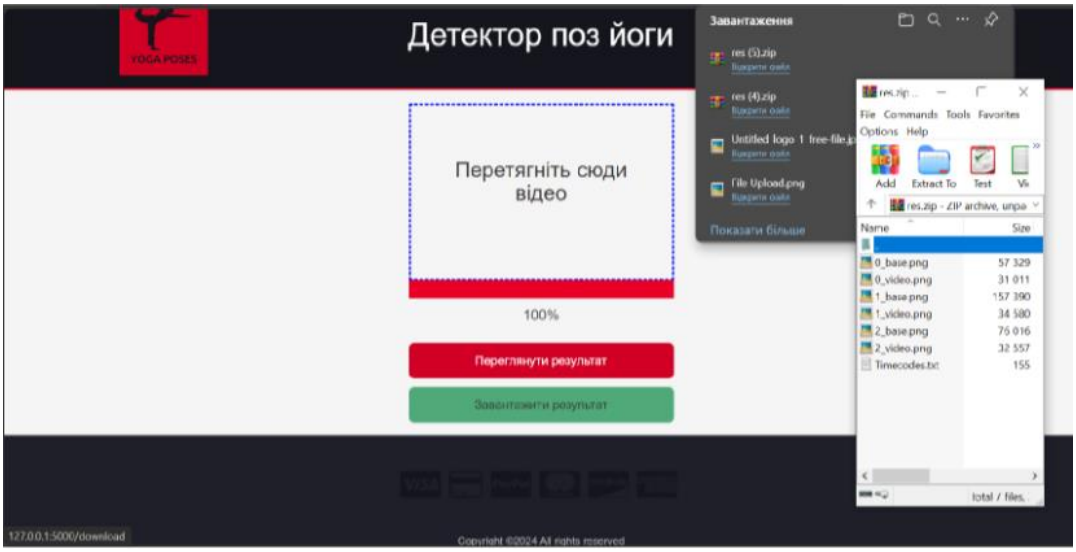

Рисунок 3.12 – Завантаження результатів

Отже, було описано клієнтську частину даного застосунку.

3.6 Висновки

У третьому розділі було обґрунтовано вибір мови програмування та технологій для розробки системи. Проаналізувавши потреби програмного продукту було обрано мову програмування Python. Для зручності розробки графічного інтерфейсу було обрано HTML, CSS, Js, Bootstrap через можливість написання графічного інтерфейсу з використанням фреймворку Flask

4 ТЕСТУВАННЯ СИСТЕМИ

4.1 Впровадження та тестування моделі нейронної мережі

Для впровадження нейронної мережі в код системи потрібно отримати фінальний файл для роботи – model.h5, для цього зазвичай використовується наступний процес в рамках роботи з бібліотекою Keras (частина TensorFlow) [25].

Перший етап, це те що уже пройдено – створення моделі (архітектура моделі, яка включає визначення шарів та їх налаштувань). Другий етап – компілювання моделі, тут в функції компіляції визначається оптимізатор, функція втрат і метрики для навчання [26]. Після чого навчена модель зберігається у файл формату «.h5», який можна використовувати для подальшого передбачення або завантаження для подальшого навчання.

В рамках впровадження модель було проведено тестування швидкості роботи на підготовленому відео формату «.mp4». Для тестування було написано код, який виконує розпізнавання за допомогою моделі, на рандомній вибірці в 100 кадрів з відео. Час розпізнавання для 20 зразків: 20.6035 секунд, час розпізнавання для 50 зразків: 26.4226 секунд, час розпізнавання для 100 зразків: 46.0296 секунд (рис. 4.1).

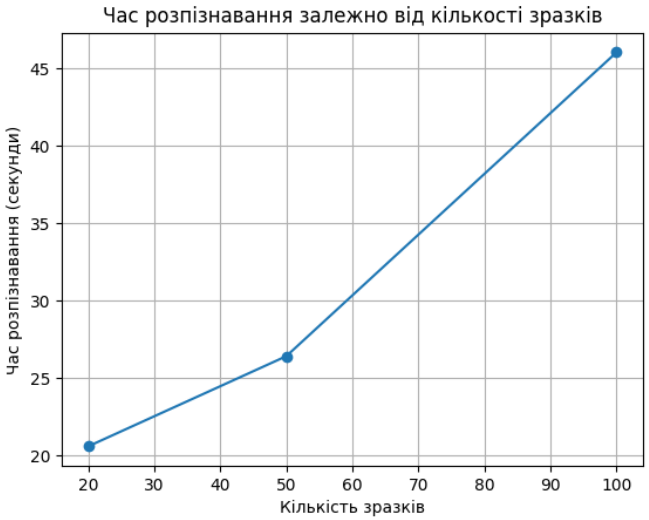

Рисунок 4.1 – Час розпізнавання

Після чого було опимізовано модель, щоб збільшити її швидкість. Щоб пришвидшити модель без втрати функціональності, використано способ зменшення розміру вхідних даних, для архітектури моделі, найбільш ефективний варіант – це зменшення розміру зображень. Минуле значення для зображень було встановлене параметрами – 300x300, після оптимізації, значення становить 280x280, що не суттєво впливає на якість розпізнавання, та значно пришвидшує час. Також для оптимізації було виконано pruning моделі, прюнінг полягає у видаленні маловажливих ваг з моделі, що зменшує її розмір і може прискорити її [27]. Прюнінг був виконаний вбудованим функціоналом бібліотеки tensorflow.

Оптимізована модель, на тому ж наборі даних показала такий результат: час розпізнавання для 20 зразків (280x280): 16.1892 секунд; час розпізнавання для 50 зразків (280x280): 17.6744 секунд; час розпізнавання для 100 зразків (280x280): 35.2110 секунд. Порівняльний графік до та після оптимізації можна переглянути (рис. 4.2).

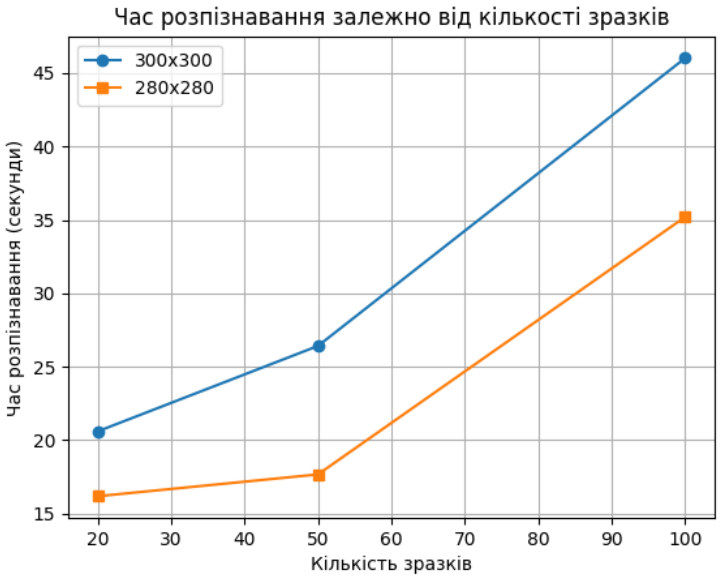

Рисунок 4.2 – Час розпізнавання оптимізованої моделі

Таким чином, середня різниця у відсотках між часом розпізнавання для розмірів зразків 300x300 і 280x280 становить приблизно 26%.

Якщо оцінювати достовірність розпізнавання моделі, то для цього було завантажено обидві моделі – власну та базову (EfficientNetB0) та проведено оцінку моделі на тестовому наборі даних, для цього був використаний датасет з позиціями йоги, з веб-ресурсу kaggle – «Yoga Pose Image Classification». Формула оцінки достовірності розпізнавання зазвичай базується на точності (accuracy). Точність визначається, як відсоток правильно передбачених класів (позицій), серед усіх передбачень. Формула для точності виглядає, як кількість правильних передбачень розділена на загальну кількість передбачень, в рамках машинного навчання це можна записати у формулі (4.1)

$$\text{Accuracy} = \frac{\sum_{i=1}^{n} 1(y_i = \hat{y}_i)}{n}, \tag{4.1}$$

де: $\boldsymbol{n}$ – загальна кількість прикладів у тестовому наборі; $\boldsymbol{y_i}$ – справжній клас для

i-го прикладу; $\widehat{\boldsymbol{y}}_{\boldsymbol{i}}$ – передбачений клас для i-го прикладу; $\mathbf{1}$ – індикаторна функція, яка дорівнює 1, якщо $\boldsymbol{y}_{\boldsymbol{i}} = \widehat{\boldsymbol{y}}_{\boldsymbol{i}}$, і 0 в іншому випадку.

У бібліотеці TensorFlow/Keras, ця метрика обчислюється автоматично під час виклику відповідного методу для моделі.

Результати проведеного тестування на підготовленому наборі даних відображений в таблиці 4.1.

Таблиця 4.1 – Оцінка точності розпізнавання моделей

Модель	**Точність (%)**
Власна модель	88,22
EfficientNetB0	77,1

Отже, завдяки збільшеній глибині моделі, та оптимізованим налаштуванням, вдалося підняти точність розпізнавання власної моделі на 11,12%.

4.2 Впровадження та тестування обробки відео

На основі розробленого методу, було вирішено скористатись готовим рішенням для аналізу відео, в контексті розробки системи, а саме фреймворком MediaPipe. Використання MediaPipe у аналізі відео, для розпізнавання позицій, має декілька переваг у порівнянні з розробкою власного інструменту з аналогічним функціоналом.

MediaPipe є «дорослим» та стабільним інструментом, він розроблений і підтримується компанією Google, що забезпечує високий рівень якості та стабільності. Він широко використовується у різних додатках і має доведений трек-рекорд успіху.

Трек-рекорд успіху, зібраний на основі даних з відкритих джерел (рис. 4.3): PyPI статистика завантажень [28]; GitHub статистика [29]; наукові публікації LearnOpenCV [30].

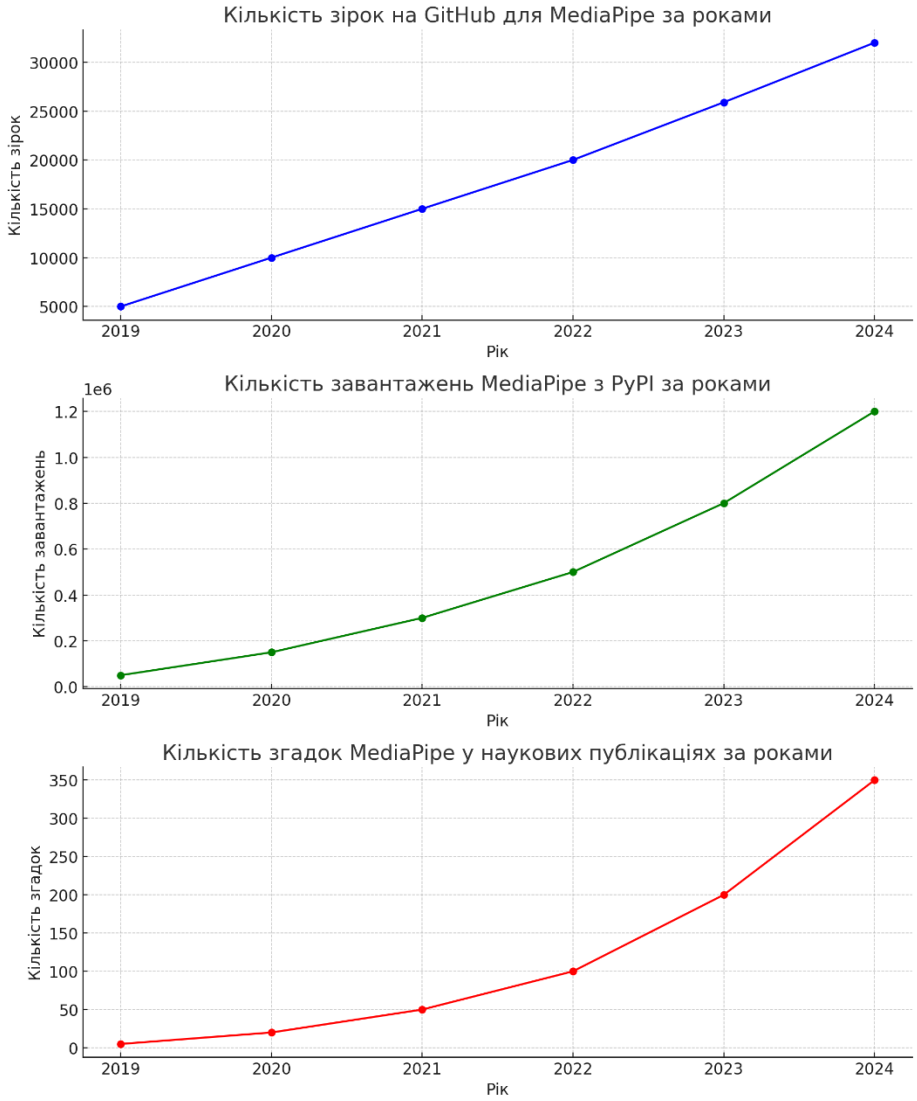

Рисунок 4.3 – Трек-рекорд успіху MediaPipe

Інший аспект впровадження даної бібліотеки – використання готового рішення дозволяє значно зекономити час та ресурси на розробку і тестування. Mediapipe пропонує готові до використання модулі для розпізнавання, які можна швидко інтегрувати у проект.

Даний фреймворк є оптимізованим та ефективним, MediaPipe оптимізований для роботи на різних платформах, включаючи мобільні пристрої. Він використовує апаратне прискорення, що забезпечує високу продуктивність і знижує навантаження на процесор. Модулі MediaPipe треновані на великих наборах даних і використовують сучасні методи машинного навчання для забезпечення високої точності розпізнавання (рис. 4.4).

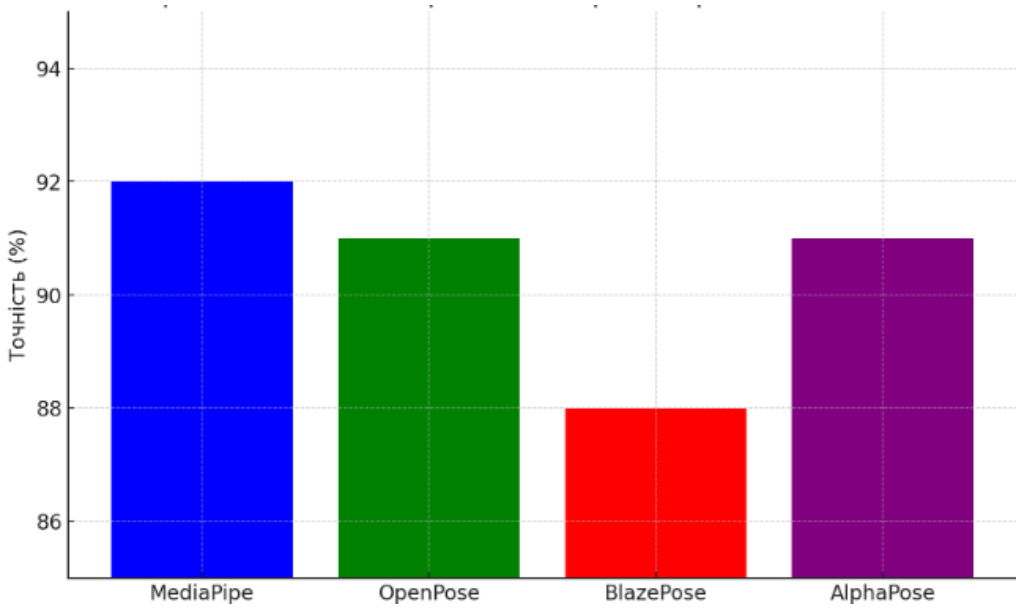

Рисунок 4.4 – Порівняння точності

При перегляді порівняння MediaPipe, та інших схожих фреймворків по інформації з відкритих джерел, MediaPipe має імовірність 92% успішного розпізнавання.

Створення власної моделі, яка досягне аналогічної точності, потребувало б значних зусиль і ресурсів. Також існує ризик якості, що власний інструмент не досягне того ж рівня точності і надійності, який забезпечує MediaPipe, особливо якщо не мати значного досвіду у цій галузі.

Враховуючи ці фактори, використання MediaPipe є більш доцільним для використання в обробці відео, забезпечуючи швидкість розробки, високу точність і ефективність при мінімальних витратах на підтримку і розвиток.

4.3 Тестування системи

Тестування веб-системи є важливим етапом розробки, що дозволяє переконатися в правильній роботі коду та відсутності помилок. Для Flask-додатку можна провести різноманітні види тестування, такі як модульне тестування, функціональне тестування, нефункціональне тестування та інтеграційне тестування.

Модульне тестування є одним з ключових підходів до автоматизованого тестування програмного забезпечення, який зосереджений на перевірці окремих модулів (частин коду) без залучення всієї системи. Метою модульного тестування є переконання в тому, що кожна частина програми працює правильно окремо від інших частин. Модульне тестування допомагає виявити і виправити помилки на ранніх етапах розробки, що робить робочий процес більш ефективним і забезпечує високу якість коду.

Функціональне тестування – це тип тестування програмного забезпечення, який зосереджений на перевірці функціональності системи відповідно до вимог та специфікацій. Це дозволяє переконатися, що програмне забезпечення виконує свої основні функції правильно та ефективно. Функціональне тестування є важливим компонентом процесу розробки програмного забезпечення, який допомагає забезпечити якість продукту та впевненість в його надійності перед випуском в продакшин.

Нефункціональне тестування – це процес перевірки атрибутів або характеристик програмного забезпечення, які не є пов'язаними з конкретними функціональними вимогами. Це тип тестування, який зосереджений на оцінці

якості програмного забезпечення, його продуктивності, надійності, безпеки та інших нефункціональних аспектах. Нефункціональне тестування є важливим елементом в процесі тестування програмного забезпечення, оскільки воно забезпечує комплексну оцінку якості та надійності продукту перед його випуском.

Інтеграційне тестування – це процес перевірки взаємодії між різними компонентами програмного забезпечення для переконання в тому, що вони працюють правильно разом. Це важливий етап у процесі розробки, який дозволяє виявити та усунути проблеми, пов'язані з інтеграцією компонентів, у розробці на раніх етапах. Інтеграційне тестування дозволяє забезпечити надійну роботу системи в цілому, перевіряючи, як окремі компоненти взаємодіють між собою. Це допомагає виявити та усунути проблеми з інтеграцією на ранніх етапах розробки, що є ключовим для успішного розгортання та експлуатації програмного забезпечення.

Протестуємо нашу систему на коректне відображення у різних браузерах та інших системах. Тестування коректного відображення є частиною нефункціонального тестування і відноситься до тестування сумісності (Compatibility Testing). Тестування сумісності визначає, як ваше програмне забезпечення працює на різних платформах, операційних системах, пристроях, браузерах та їх версіях. Основна мета — забезпечити, що програмне забезпечення працює коректно та однаково на різних середовищах. Тестування сумісності є критично важливим для забезпечення якісного користувацького досвіду, оскільки воно дозволяє виявити та усунути проблеми, пов'язані з різноманітністю платформ та конфігурацій, на яких може використовуватися ваше програмне забезпечення

Для початку, запустимо систему у браузері Opera One (рис. 4.5) (версія: 108.0.5067.29), Система:Windows 10 64-bit, Версія Chromium:122.0.6261.112, де переглянемо чи все відображається вірно та чи немає дефектів.

Opera – це веб-браузер, який розробляється компанією Opera Software. Він був запущений у 1995 році і став одним з перших альтернативних браузерів до Internet Explorer. Opera відрізняється своєю швидкістю, інноваційними функціями та безпекою. Opera є відмінним вибором для тих, хто шукає швидкий, надійний і інноваційний веб-браузер з високим рівнем безпеки.

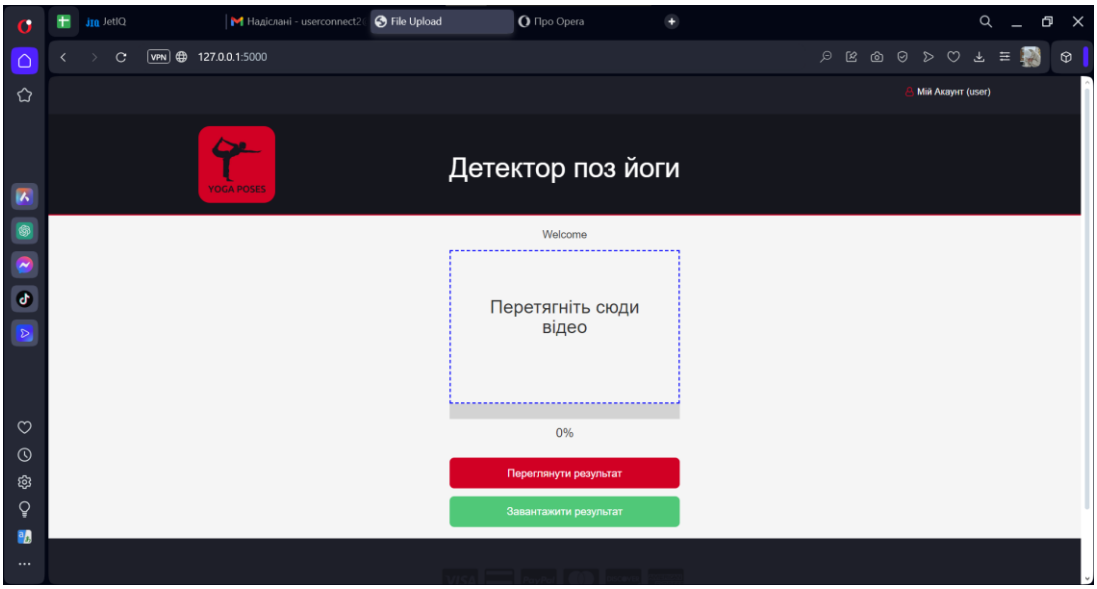

Рисунок 4.5 – Відображення в Opera One

Не знайшовши ніяких проблем перевіримо систему у іншому браузері – Microsoft Edge (рис. 4.6), Версія 123.0.2420.97 (Офіційна збірка) (64-розрядна версія).

Microsoft Edge – це веб-браузер, який розробляється компанією Microsoft і є стандартним браузером для операційних систем Windows 10 і новіших версій. Він був представлений у 2015 році як заміна для Internet Explorer і відзначився значними поліпшеннями в продуктивності, безпеці та функціональності.

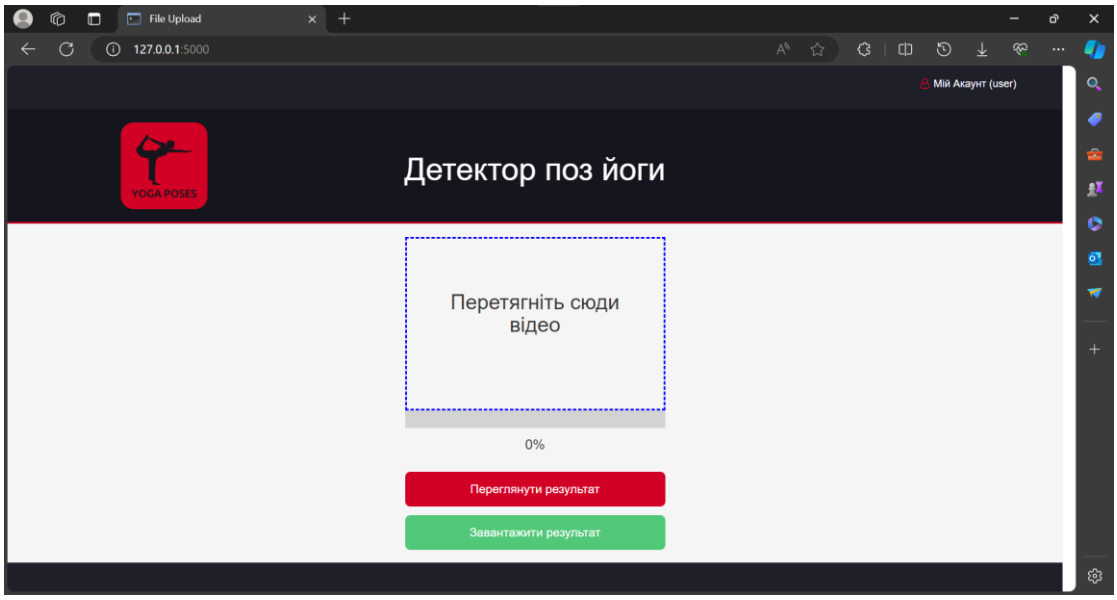

Рисунок 4.6 – Відображення в Microsoft Edge

Також потрібно перевірити застосунок на адаптивність. Адаптивний веб-дизайн – підхід до створення веб-сайтів, який забезпечує оптимальний перегляд та взаємодію з контентом на різних пристроях і екранах. Головна мета адаптивного дизайну полягає в тому, щоб веб-сайт адекватно підлаштовувався під розмір екрана, тип пристрою та його характеристики.

Переглянемо адаптивне відображення застосунку для екрану пристрою iPhone 14 Pro Max (рис. 4.7)

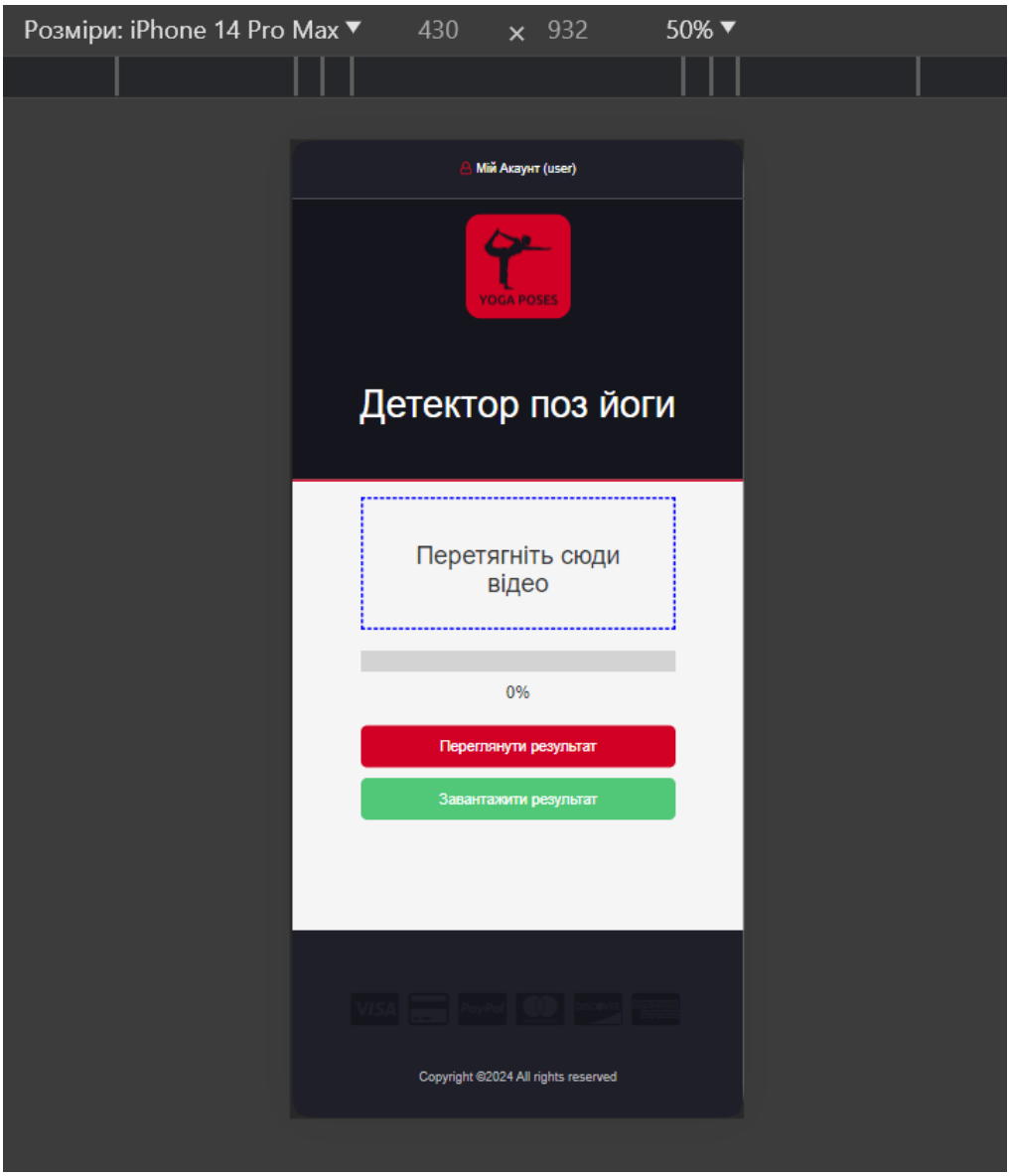

Рисунок 4.7 – Відображення для iPhone 14 Pro Max

Так як система добре відображається на всіх пристроях і має адаптивний дизайн, який підлаштовується навіть під мобільні пристрої, де екрани кардинально менші ніж на ПК чи ноутбуках.

Також було проведено тестування чорного ящика (black-box testing) – це метод тестування програмного забезпечення [31], при якому тестувальник не знає внутрішньої структури або реалізації системи. Тестування зосереджене на функціональності програми, перевіряючи, чи виконує вона свої завдання

відповідно до специфікацій. Для тестування на основі специфікацій (метод включає створення тестових випадків на основі специфікацій та вимог до програми) можемо розглянути наступні тест-кейси: (рис. 4.8) – перевірка входу до системи з валідними обліковими даними 1.1.1; (рис 4.9) – завантаження відео для аналізу.

Test case	
id	1.1.1
shot desc	Перевірка входу з валідними обліковими даними
precondition	1. Перейти на сайт
steps	1. Натиснути кнопку 'Увійти' 2. Ввести логін: 'testuser' 3. Ввести пароль: 'testpassword' 4. Натиснути кнопку 'Увійти'
expected result	Успішний вхід до системи
actual result	pass

Рисунок 4.8 – Тест-кейс 1.1.1

Test case	
id	1.1.2
shot desc	Завантаження відео для аналізу
precondition	1. Увійти до системи
steps	1. Натиснути кнопку 'Завантажити відео' 2. Обрати відеофайл у форматі MP4 3. Натиснути кнопку 'Завантажити'
expected result	Успішне завантаження файлу та початок його обр
actual result	pass

Рисунок 4.9 – Тест-кейс 1.1.2

Для тестування граничних значень, метода який перевіряє поведінку системи при введенні граничних (мінімальних та максимальних) значень, в даному випадку перевірка входу до системи з мінімальною довжиною пароля, маємо наступний тест-кейс (рис. 4.10).

Test case	
id	1.1.3
shot desc	Перевірка входу з коротким паролем
precondition	1. Перейти на сайт
steps	1. Натиснути кнопку 'Увійти' 2. Ввести логін: 'testuser' 3. Ввести пароль: 'a' 4. Натиснути кнопку 'Увійти'
expected result	Відмова у вході через недостатню довжину парол
actual result	pass

Рисунок 4.10 – Тест-кейс 1.1.3

Та тест-кейс на перевірку завантаження занадто великого файлу, відеофайлу розміром понад 500 МБ (рис. 4.11).

Test case	
id	1.1.4
shot desc	Завантаження великого відеофайлу
precondition	1. Увійти до системи
steps	1. Натиснути кнопку 'Завантажити відео' 2. Обрати відеофайл розміром понад 500 МБ 3. Натиснути кнопку 'Завантажити'
expected result	Відмова у завантаженні через перевищення ліміту
actual result	pass

Рисунок 4.11 – Тест-кейс 1.1.4

Ще було вирішено провести фукціональне тестування модуля автентифікації юніт-тестами (рис. 4.12), які охоплюють даний модуль застосунку.

```
Клас TestApp наслідує unittest.TestCase:
  Метод setUp:
    Створити клієнт тестування для додатку

  Метод test_index_redirects_to_login_if_not_logged_in:
    Надіслати GET-запит на головну сторінку
    Перевірити, чи перенаправлення на сторінку входу

  Метод test_login_with_correct_credentials:
    Виконати тестовий контекст додатку
      Надіслати POST-запит на сторінку входу з правильними обліковими даними
      Перевірити, чи статус відповіді 200
      Перевірити, чи містить відповідь повідомлення "Welcome"

  Метод test_login_with_incorrect_credentials:
    Надіслати POST-запит на сторінку входу з неправильними обліковими даними
    Перевірити, чи містить відповідь повідомлення "Невірний логін або пароль"

  Метод test_register_with_existing_username:
    Надіслати POST-запит на сторінку реєстрації з існуючим ім'ям користувача
    Перевірити, чи містить відповідь повідомлення "Логін вже зареєстровано"

  Метод test_register_with_new_username:
    Надіслати POST-запит на сторінку реєстрації з новим ім'ям користувача
    Перевірити, чи містить відповідь повідомлення "Аккаунт створено успішно"

Запустити всі тести, якщо скрипт виконується напряму
```

Рисунок 4.12 – Представлення юніт-тестів автентифікації псевдокодом

Модуль було протестовано на успішний роут по сторінці логіювання, саме логіювання, також перевірку на винятки, при введенні неправильного логіну чи паролю, перевірку на вже зареєстрований акаунт, та на успішність створення акаунту. Результати проходження юніт-тестів можна побачити (рис. 4.9).

```
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
.....
----------------------------------------------------------------------
Ran 5 tests in 0.889s

OK
[Finished in 35.5s]
```

Рисунок 4.13 – Результат юніт-тестів

4.4 Створення інструкції користувача

Інструкція користувача передбачає визначення технічних вимог для запуску програмного продукту. Оскільки застосунок розроблено на базі фреймворку Flask, він є веб-сервісом, тому його можна використовувати на будь-якому комп'ютері або пристрої, підключеному до Інтернету. Конкретних вимог до потужності апаратного забезпечення немає, адже розпізнавання позицій людини відбувається на сервері. Веб-застосунок зручно відображається на різних пристроях, включаючи комп'ютери, ноутбуки та навіть телефони. Користувачі можуть комфортно використовувати веб-застосунок, незалежно від розміру екрану.

Рекомендується використовувати оновлені версії операційних систем, таких як Windows, macOS або Linux. Для доступу до веб-сервісу необхідно мати встановлений веб-браузер. Веб-сервіс сумісний з різними популярними веб-браузерами, такими як Google Chrome, Mozilla Firefox, Safari, Microsoft Edge тощо. Для оптимальної сумісності та безпеки рекомендується використовувати останню версію веб-браузера.

Для використання веб-сервісу потрібне активне інтернет-підключення. Чим стабільніше та швидше підключення до Інтернету, тим плавнішою буде взаємодія з веб-сервісом і отримання результатів розпізнавання.

Перейшовши за посиланням користувача зустрічає сторінка входу у застосунок, після авторизації, або ж якщо користувач ще не має акаунта – реєстрації, користувача редіректить на головну сторінку, де знаходиться вікно завантаження відео, для розпізнавання позицій на основі нейронної мережі. Після завантаження відео починається його обробка, яка відображається на відсотковій шкалі, після завершення обробки користувач може переглянути результат (рис. 4.14) із позиціями, де знаходяться їх назви, тайм коди, фото розпізнаних позицій з відео та їх відповідники із датасету, для наглядності.

Також користувач може завантажити ці результати в архів та потім переглянути на своїй робочій машині.

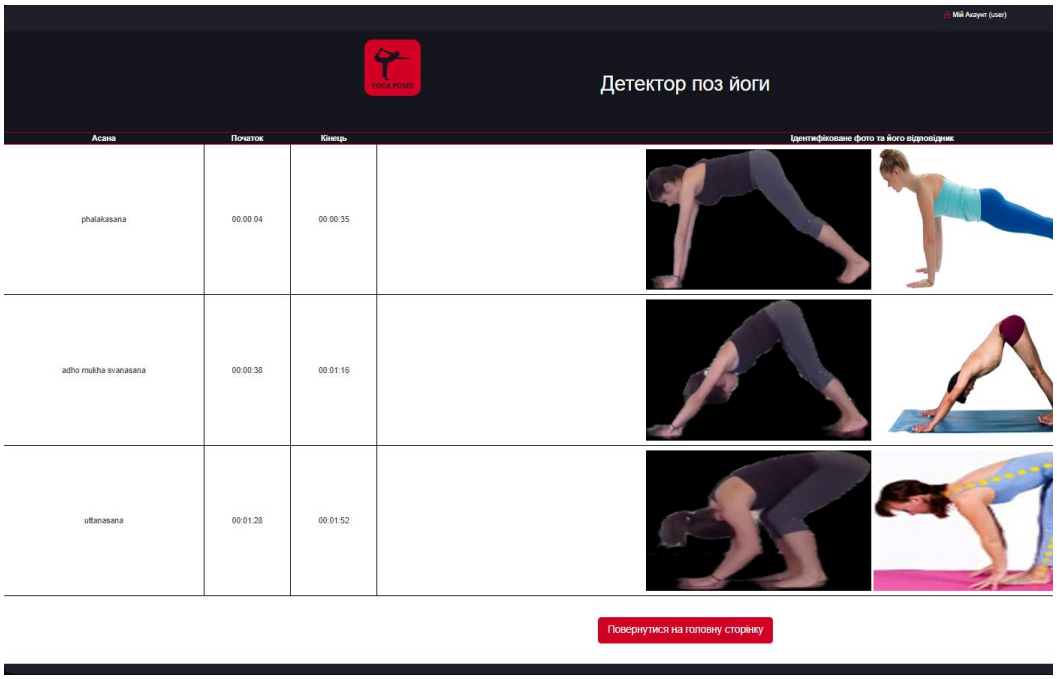

Рисунок 4.14 – Сторінка з результатами обробки відео

4.5 Висновки

У четвертому розділі було проведено тестування частин програмного застосунку з розпізнавання позицій людини на основі нейронної мережі. Було перевірено коректність відображення програмного застосунку на різних інструментах, а також проведені юніт-тести по авторизації у застосунку. Також було розроблено інструкцію користувача по-основному функціоналу експлуатації системи.

ВИСНОВКИ

У межах бакалаврської дипломної роботи було розроблено систему для розпізнавання позицій людини на основі нейронної мережі.

Для роботи було використано середовище розробки на Python – Sublime Text 3 Build 4169.

У першому розділі, було проведено аналіз сучасного стану застосунків для розпізнавання позицій людини на основі нейронних мережі, також із розробленою системою було зроблено порівняльний аналіз аналогів. Також в рамках першого розділу, було виконано аналіз методів розв’язання поставленої задачі. А також було встановлено цілі для застосунку з розпізнавання позицій людини на основі нейронної мережі.

У другому розділі роботи було проведено розробку структури та алгоритмів програмного продукту, виконано аналіз принципів системи. Було розроблено схеми загального алгоритму роботи системи, а також модель нейронної мережі для розпізнавання позицій людини. Також було представлено UML діаграми та структуру інтерфейсу.

У третьому розділі було виконано варіантний аналіз і обґрунтування вибору засобів для реалізації програмного забезпечення, було визначено, що для розробки буде використано мову програмування Python, а система буде працювати на веб-фреймворку Flask. Також було налаштовано модель нейронної мережі, розроблено компоненти обробки відео, та розроблено компонент авторизації користувача в системі, з інтерфейсом програмного застосунку.

У четвертому розділі було проведено впровадження та тестування основних модулів системи, а також проведенні різноманітні тестування самої системи способом чорного ящика, юніт-тестами, тощо. Та розроблено інструкції користувача.

Отже, в дипломній бакалаврській роботі ми домоглися виконання усіх поставлених задач, та довели доцільність створення системи для розпізнавання позицій людини.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Яцик Д.І. Про підвищення достовірності розпізнавання позиції людини на основі нейромережі : Матеріали конференції «Міжнародна науково-практична інтернет-конференція студентів, аспірантів та молодих науковців "МОЛОДЬ В НАУЦІ: ДОСЛІДЖЕННЯ, ПРОБЛЕМИ, ПЕРСПЕКТИВИ"», Вінниця, 2024. : ВНТУ, 2024. URL: https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/view/21196 (дата звернення: 05.05.2024)
2. TikTok URL: https://www.tiktok.com/uk (дата звернення: 08.05.2024)
3. Fitbod URL: https://fitbod.me (дата звернення: 08.05.2024)
4. Snapchat URL: https://www.snapchat.com/explore/info (дата звернення: 08.05.2024)
5. Freeletics URL: https://www.freeletics.com (дата звернення: 08.05.2024)
6. Fitbit URL: https://www.fitbit.com/global/us/home (дата звернення: 08.05.2024)
7. Гунченко Ю.О., Мартинович Л.Я., Волинець О. О. Дослідження нейронних мереж для класифікації зображень / за ред. І.В. Толока. Київ. : ВІКНУ, 2021. 103-104 с.
8. Blog Roboflow. What is OpenPose? A Guide for Beginners URL: https://blog.roboflow.com/what-is-openpose/ (дата звернення: 21.05.2024)
9. Analytics Vidhya. Posture Detection using PoseNet with Real-time Deep Learning project URL: https://www.analyticsvidhya.com/blog/2021/09/posture-detection-using-posenet-with-real-time-deep-learning-project/ (дата звернення: 21.05.2024)

10. Viso.ai MediaPipe: Google’s Open Source Framework (2024 Guide) URL: https://viso.ai/computer-vision/mediapipe/ (дата звернення: 21.05.2024)

11. Web-розробка на Python і Django. / за ред. В. Грабовський. Київ: Видавництво Національного університету "Львівська політехніка", 2019. 192 с.

12. Машинне навчання та нейронні мережі в практиці. / за ред. Н. Петрова, І. Сидоренко. Київ: Видавничий дім "Ін Юре", 2020. 336 с.

13. Deep Learning на Python. / за ред. С. Чернікова. Київ: Видавництво "Детектор медіа", 2019. 272 с.

14. "Learning Python" O'Reilly Media / by Mark Lutz, 2013. 1648 p.

15. Python для професіоналів: детальний огляд мови Python. / за ред. І. Стрижень. Львів: Видавництво Старого Лева, 2018. 512 с.

16. Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython. 2nd Edition. O'Reilly Media / by McKinney, Wes., 2017. 544 p.

17. Flask Web Development with Python Tutorial. O'Reilly Media / by Grinberg M., 2018. 300 p.

18. Flask Web Development: Developing Web Applications with Python. O'Reilly Media / by Miguel G., 2018. 336 p.

19. Flask Web Development: Developing Web Applications with Python. O'Reilly Media / by Miguel Grinberg., 2018. 332 p.

20. Zfort. TensorFlow, Keras, Scikit-learn і PyTorch URL: https://www.zfort.com.ua/blog/porivnyannya-naikrashikh-instrumentiv-mashinnogo-navchannya-tensorflow-keras (дата звернення: 21.05.2024)

21. Д. Дакетт, М. Хеселтін, Р. Макфарланд HTML і CSS. Дизайн та створення веб-сайтів. Київ: Видавництво "К.І.С.", 2021. 400 с.

22. W3schools. Bootstrap 3 Tutorial URL: https://www.w3schools.com/bootstrap/ (дата звернення: 21.05.2024)

23. Roboflow What is EfficientNet? The Ultimate Guide. URL: https://blog.roboflow.com/what-is-efficientnet/ (дата звернення: 21.05.2024)

24. Foxminded Deep learning: основи напряму, інструменти та технології URL: https://foxminded.ua/deep-learning/ (дата звернення: 08.05.2024)

25. TensorFlow URL: https://www.tensorflow.org/ (дата звернення: 08.05.2024)

26. Машинне навчання та нейронні мережі. / за ред. М. Горобець, О. Кравець. Київ: Видавництво "Університетська книга", 2019. 288 с.

27. Deep Learning for Computer Vision with Python. PyImageSearch / by Adrian Rosebrock., 2017. 301 p.

28. PyPI. Статистика завантажень URL: https://pypi.org/project/mediapipe (дата звернення: 08.05.2024)

29. GitHub MediaPipe URL: https://github.com/google-ai-edge/mediapipe (дата звернення: 08.05.2024)

30. LearnOpenCV URL: https://learnopencv.com/tag/mediapipe/ (дата звернення: 08.05.2024)

31. Black Box Testing – Software Engineering URL: https://www.geeksforgeeks.org/software-engineering-black-box-testing/ (дата звернення: 08.05.2024)

ДОДАТКИ

Додаток А – Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
д.т.н., професор
_____ О. Н. Романюк
«12» березня 2024 р.

Технічне завдання
на бакалаврську дипломну роботу «Розробка системи розпізнавання позицій людини на основі нейронної мережі» за спеціальністю 121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:
_____ к.т.н., доцент О. М. Ткаченко
«12» березня 2024 р.
Виконав:
___ студент гр. 2ПІ-20б Д. І. Яцик
«12» березня 2024 р.

Вінниця – 2024 року

1. Найменування та галузь застосування

Бакалаврська дипломна робота: «Розробка системи розпізнавання позицій людини на основі нейронної мережі».

Галузь застосування – спорт.

2. Підстава для розробки

Підставою для виконання бакалаврської дипломної роботи (БДР) є індивідуальне завдання на БДР та наказ №80 від «11» березня 2024 р. ректора ВНТУ про закріплення тем БДР.

3. Мета та призначення розробки

Метою дослідження є підвищення ймовірності правильного розпізнавання позицій людини на основі нейронної мережі та забезпечення зручної взаємодії з користувачем через застосунок.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись БДР.

1. Яцик Д.І. Про підвищення достовірності розпізнавання позиції людини на основі нейромережі : Матеріали конференції «Міжнародна науково-практична інтернет-конференція студентів, аспірантів та молодих науковців "МОЛОДЬ В НАУЦІ: ДОСЛІДЖЕННЯ, ПРОБЛЕМИ, ПЕРСПЕКТИВИ"», Вінниця, 2024. : ВНТУ, 2024. URL: https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/view/21196 (дата звернення: 05.05.2024)

5. Технічні вимоги

Вихідні дані до роботи: середовище програмної розробки – Sublime Text 3; мова програмування – Python; фреймворки – Flask, Bootstrap; допоміжні технології – HTML, CSS, JS; персональний ПК з ОС Windows 10; відео з позиціями людини в йозі

6. Конструктивні вимоги

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

1. Пояснювальна записка до БДР.
2. Технічне завдання.
3. Лістинги програми

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ

9. Стадії та етапи розробки:

№ п\п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи
1	Аналіз проблеми, обгрунтування актуальності розробки системи та постановка задачі	14.03.24 –22.03.24
2	Розробка структури та алгоритмів програмного продукту	13.03.24 –22.03.24
3	Аналіз та вибір середовища програмування	25.03.24 –19.04.24
4	Розробка модулів програмного продукту	22.04.24 –10.05.24
5	Тестування програми	13.05.24 – 24.05.24
6	Оформлення матеріалів до захисту БДР	27.05.24 – 30.05.24

10. Порядок контролю та прийняття

Виконання етапів бакалаврської дипломної роботи контролюється керівником згідно з графіком виконання роботи. Прийняття бакалаврської дипломної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

Додаток Б – Протокол перевірки на плагіат

ПРОТОКОЛ
ПЕРЕВІРКИ БАКАЛАВРСЬКОЇ ДИПЛОМНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка системи розпізнавання позицій людини на основі нейронної мережі

Тип роботи: БДР

Підрозділ: кафедра програмного забезпечення, ФІТКІ

Науковий керівник: Ткаченко О.М.

Оригінальність	95.8%
Схожість	4.2%

Аналіз звіту подібності

▪ **Запозичення, виявлені у роботі, оформленні коректно і не містять ознак плагіату.**
☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цілісності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.
☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховання недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Черноволик Г.О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи _____ Яцик Д.І.

Керівник роботи _____ Ткаченко О.М.

Додаток В – Лістинг програми

```
//Сервер
//
from flask import Flask, render_template, request, jsonify, send_file, redirect, url_for, flash, send_from_directory
from werkzeug.utils import secure_filename
from flask_bcrypt import generate_password_hash as gen, check_password_hash as check
from contextlib import suppress
import pandas as pd
import numpy as np
import time
import os
import io
import zipfile
import base64

import cv2
import mediapipe as mp

from tensorflow.keras.models import load_model

# Завантажуємо модель
model = load_model('model.h5')

mp_drawing = mp.solutions.drawing_utils
mp_pose = mp.solutions.pose
pose = mp_pose.Pose(min_detection_confidence=0.5, min_tracking_confidence=0.5, enable_segmentation=True)


app = Flask(__name__)
```

```
app.secret_key = 'secret_key'

progress = 0
status = '0%'
user = None
df_result=None

@app.route('/')
def index():
    if user is not None:
        return render_template('index.html', username=user)
    else:
        return redirect(url_for('login'))



@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        passwords = pd.read_csv('passwords.txt', sep='\t')
        username=request.form['username']
        if passwords.query(f'login.isin(["{username}"])').__len__()==1:
            if check(passwords.query(f'login.isin(["{username}"])')['hash'].iloc[0],
request.form['password']):
                global user
                user = username
                flash('Welcome')
                return redirect(url_for('index'))
        flash('Невірний логін або пароль')
    return render_template('login.html')

@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
```

```
            passwords = pd.read_csv('passwords.txt', sep='\t')
            username=request.form['username']
            if passwords.query(f'login.isin(["{username}"])').__len__()==0:
                pd.DataFrame([username, gen(request.form['password']).decode('utf-8')],
index=passwords.columns).T\
                .to_csv('passwords.txt', sep='\t',index=False,header=False, mode='a')
                global user
                user=username
                flash('Аккаунт створено успішно')
                return redirect(url_for('login'))
            else:
                flash('Логін вже зареєстровано')
        return render_template('register.html')

    @app.route('/results')
    def results():
        global df_result
        if df_result is not None:
            base_images = []
            video_images = []
            user_images_dir = f'static/images/{user}/'

            # Створюємо директорію для зберігання фотографій користувача, якщо вона не
існує
            os.makedirs(user_images_dir, exist_ok=True)

            # Розпаковуємо архів і зберігаємо файли у відповідну папку
            with zipfile.ZipFile(f'user_data/{user}/res.zip', 'r') as zip_ref:
                zip_ref.extractall(user_images_dir)

            # Отримуємо список шляхів до базових та відео зображень
            for filename in os.listdir(user_images_dir):
                if filename.endswith('_base.png'):
```

```
                base_images.append(f'{user_images_dir}{filename}')
            elif filename.endswith('_video.png'):
                video_images.append(f'{user_images_dir}{filename}')

        return render_template('results.html', base_images=base_images,
video_images=video_images, df_result=df_result, username=user)
    else:
        flash('Немає результатів для відображення')
        return redirect(url_for('index'))


@app.route('/results/<path:filename>')
def get_image(filename):
    return send_from_directory('static/images/{user}/', filename)


@app.route('/download')
def download():
    global user
    try:
        response = send_file(f'user_data/{user}/res.zip', as_attachment=True)
        return response
    except BaseException:
        flash('Ви ще не обробили жодне відео')
    return redirect(url_for('index'))

@app.route('/logout')
def logout():
    global user
    user=None
    return redirect(url_for('index'))

@app.route('/task_progress')
```

```
        def task_progress():
            global user
            table=[]
            try:
                table=pd.read_csv(f'user_data/{user}/Timecodes.txt',sep='\t',       index_col=0).iloc[:,:-
1].values.tolist()
            except BaseException:
                table=[]
            return jsonify({'progress': progress, 'text': status, 'table': table})

        @app.route('/upload', methods=['POST'])
        def upload():
            global user,progress,status
            if 'file' not in request.files:
                return 'No file uploaded'

            file = request.files['file']

            with suppress(BaseException):
                os.mkdir(f'user_data/{user}')
            file.save('user_data/' + user + '/' + 'video.mp4')
            try:
                global df_result
                df_result = process(user)
            except:
                flash('Сталася помилка')
                progress = 0
                status = '0%'
                return 'File uploaded successfully'
            progress = 100
            status = '100%'
            return 'File uploaded successfully'
```

```
def process(username):
    global user, mp_drawing, mp_pose, pose, model

    cap = cv2.VideoCapture(f'user_data/{username}/video.mp4')
    fps = cap.get(cv2.CAP_PROP_FPS).__int__()
    length = cap.get(cv2.CAP_PROP_FRAME_COUNT).__int__()//fps

    df=\
    pd.DataFrame(range(length), columns=['i'])\
    .assign(
    seek=lambda x: x['i']*fps,
    timecode=lambda x: pd.to_datetime(x['i'],unit='s').astype(str).str[-8:],
    frame=lambda x: x[['seek']].applymap(lambda x: cap.read()[1] if cap.set(cv2.CAP_PROP_POS_FRAMES, value=x) else None),
    shape=lambda x: x[['frame']].applymap(lambda x: None if x is None else x.shape)
    )

    poses_list = ['adho mukha svanasana', 'adho mukha vriksasana', 'agnistambhasana', 'ananda balasana', 'anantasana', 'anjaneyasana', 'ardha bhekasana', 'ardha chandrasana', 'ardha matsyendrasana', 'ardha pincha mayurasana', 'ardha uttanasana', 'ashtanga namaskara', 'astavakrasana', 'baddha konasana', 'bakasana', 'balasana', 'bhairavasana', 'bharadvajasana i', 'bhekasana', 'bhujangasana', 'bhujapidasana', 'bitilasana', 'camatkarasana', 'chakravakasana', 'chaturanga dandasana', 'dandasana', 'dhanurasana', 'durvasasana', 'dwi pada viparita dandasana', 'eka pada koundinyanasana i', 'eka pada koundinyanasana ii', 'eka pada rajakapotasana', 'eka pada rajakapotasana ii', 'ganda bherundasana', 'garbha pindasana', 'garudasana', 'gomukhasana', 'halasana', 'hanumanasana', 'janu sirsasana', 'kapotasana', 'krounchasana', 'kurmasana', 'lolasana', 'makara adho mukha svanasana', 'makarasana', 'malasana', 'marichyasana i', 'marichyasana iii', 'marjaryasana', 'matsyasana', 'mayurasana', 'natarajasana', 'padangusthasana', 'padmasana', 'parighasana', 'paripurna navasana', 'parivrtta janu sirsasana', 'parivrtta parsvakonasana', 'parivrtta trikonasana', 'parsva bakasana', 'parsvottanasana', 'pasasana', 'paschimottanasana', 'phalakasana', 'pincha mayurasana', 'prasarita padottanasana', 'purvottanasana', 'salabhasana', 'salamba bhujangasana', 'salamba sarvangasana', 'salamba sirsasana', 'savasana', 'setu bandha sarvangasana', 'simhasana', 'sukhasana',
```

'supta baddha konasana', 'supta matsyendrasana', 'supta padangusthasana', 'supta virasana', 'tadasana', 'tittibhasana', 'tolasana', 'tulasana', 'upavistha konasana', 'urdhva dhanurasana', 'urdhva hastasana', 'urdhva mukha svanasana', 'urdhva prasarita eka padasana', 'ustrasana', 'utkatasana', 'uttana shishosana', 'uttanasana', 'utthita ashwa sanchalanasana', 'utthita hasta padangustasana', 'utthita parsvakonasana', 'utthita trikonasana', 'vajrasana', 'vasisthasana', 'viparita karani', 'virabhadrasana i', 'virabhadrasana ii', 'virabhadrasana iii', 'virasana', 'vriksasana', 'vrischikasana', 'yoganidrasana']

```
arr=[]
dict_vis={}
iteration=-1
for tc,img in df[['timecode','frame']].values:
    iteration+=1
    global progress, status
    progress=100*iteration/len(df)
    status="{0:.1f}".format(progress) + f'% {iteration} / {len(df)}'
    pose_results = pose.process(img)
    mask=pose_results.segmentation_mask
    if mask is not None:
        i, j = np.where(mask>0.1)
        indices = tuple(np.meshgrid(np.arange(min(i), max(i) + 1),np.arange(min(j), max(j) + 1),indexing='ij'))

        img_to_predict = cv2.resize(img[indices], (300, 300)).reshape((-1,300,300,3))
        prediction = model.predict(img_to_predict, verbose=0)
        class_name = poses_list[np.argmax(prediction)]
        probability = prediction.max()
        if probability>0.9:
            img_to_visualize = np.einsum('ijk,ij->ijk', img, mask)[indices]
            dict_vis.update({tc: img_to_visualize})



        arr.append((tc,class_name, probability))
```

```
df2=\
pd.DataFrame(arr, columns=['tc','c','p'])\
.assign(
m95=lambda x: np.where(x['p']>0.9, x['c'], np.nan),
mf=lambda x: x['m95'].ffill(),
mb=lambda x: x['m95'].bfill(),
mg1=lambda x: np.where(x['mf']==x['mb'], x['mf'], np.nan)
)\
.query('~mg1.isna()')\
.assign(mg2=lambda x: x['mg1'].eq(x['mg1'].shift()).astype(int)
).query('mg2==1')

name,vis, vis_tc=None,None,None
arr2=[]
for i,r in df2.iterrows():
    if r['mg1']!=name:
        if name:
            arr2.append([name, first_tc, last_tc, vis, vis_tc])
        name, first_tc, last_tc=r['mg1'], r['tc'], r['tc']
    else:
        last_tc=r['tc']
    if last_tc in dict_vis.keys():
        vis=dict_vis[last_tc]
        vis_tc=last_tc

df3=\
pd.DataFrame(arr2, columns=['asana','start','end','image', 'tc'])



with suppress(BaseException):
    os.remove(f'user_data/{user}/res.zip')
```

```
        with zipfile.ZipFile(f'user_data/{user}/res.zip', "a", zipfile.ZIP_STORED, False) as fz:
            for i,r in df3.iterrows():
                fz.writestr(f'{i}_video.png', cv2.imencode('.png', r['image'])[1])
                with open(f'dataset/{r["asana"]}/' + (os.listdir(f'dataset//{r["asana"]}//')[0]), 'rb') as
fr:
                    fz.writestr(f'{i}_base.png', fr.read())
            df3\
            .drop('image', axis=1)\
            .to_csv(f'user_data/{user}/Timecodes.txt',sep='\t',index=True)
            with open(f'user_data/{user}/Timecodes.txt', 'rb') as fr:
                fz.writestr('Timecodes.txt', fr.read())

        return df3

    if __name__ == '__main__':
        app.run(debug=True)
    //
    //
    //index.html
    //
    <!DOCTYPE html>
    <html>
    <head>
        <title>File Upload</title>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <meta name="description" content="">
        <meta name="author" content="Mark Otto, Jacob Thornton, and Bootstrap contributors">
        <meta name="generator" content="Hugo 0.83.1">
        <title>Детектор поз йоги</title>

        <link rel="canonical" href="https://getbootstrap.com/docs/5.0/examples/sign-in/">
```

```
<!-- Bootstrap core CSS -->
<link href="/static/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-wEmeIV1mKuiNpC+IOBjI7aAzPcEZeedi5yW5f2yOq55WWLwNGmvvx4Um1vskeMj0" crossorigin="anonymous">

<link type="text/css" rel="stylesheet" href="static/css/style1.css"/>

<link type="text/css" rel="stylesheet" href="static/css/style.css"/>

<!-- Google font -->
<link href="https://fonts.googleapis.com/css?family=Montserrat:400,500,700" rel="stylesheet">

<!-- Bootstrap -->
<link type="text/css" rel="stylesheet" href="static/css/bootstrapResults.min.css"/>

<!-- Font Awesome Icon -->
<link rel="stylesheet" href="static/css/font-awesome.min.css">


<!-- Favicons -->



<style>
  .bd-placeholder-img {
    font-size: 1.125rem;
    text-anchor: middle;
    -webkit-user-select: none;
    -moz-user-select: none;
```

```
    user-select: none;
  }

  @media (min-width: 768px) {
    .bd-placeholder-img-lg {
      font-size: 3.5rem;
    }
  }
</style>


<!-- Custom styles for this template -->
<link href="/static/css/signin.css" rel="stylesheet">
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
<script>
    $(document).ready(function() {
        var dropArea = document.getElementById('drop-area');
        var taskProgressBar = document.getElementById('task-progress-bar');
        var taskProgressText = document.getElementById('task-progress-text');


        function updateTaskProgress(progress, text) {
            taskProgressBar.style.width = progress + '%';
            taskProgressText.textContent = text;
        }

        function fetchTaskProgress() {
            fetch('/task_progress')
                .then(response => response.json())
                .then(data => {
                    updateTaskProgress(data.progress, data.text);
                })
                .catch(error => {
```

```
            console.error('Error:', error);
        });
}

// Update task progress every second
setInterval(fetchTaskProgress, 1000);

dropArea.addEventListener('dragover', function(e) {
    e.preventDefault();
    dropArea.classList.add('highlight');
});

dropArea.addEventListener('dragleave', function(e) {
    e.preventDefault();
    dropArea.classList.remove('highlight');
});

dropArea.addEventListener('drop', function(e) {
    e.preventDefault();
    dropArea.classList.remove('highlight');
    var files = e.dataTransfer.files;
    handleFiles(files);
});

dropArea.addEventListener('click', function() {
    var fileInput = document.createElement('input');
    fileInput.type = 'file';
    fileInput.accept = 'video/*';
    fileInput.onchange = function(e) {
        var files = e.target.files;
        handleFiles(files);
    };
    fileInput.click();
```

```
    });

    function handleFiles(files) {
        var file = files[0];
        var formData = new FormData();
        formData.append('file', file);

        $.ajax({
            type: 'POST',
            url: '/upload',
            data: formData,
            processData: false,
            contentType: false,
            success: function(response) {
                console.log('File uploaded successfully');
            },
            error: function(error) {
                console.log('Error uploading file');
            }
        });
    }
});

var taskProgressBar = document.getElementById('task-progress-bar');
var taskProgressText = document.getElementById('task-progress-text');


function updateTaskProgress(progress) {
    taskProgressBar.style.width = progress + '%';
    taskProgressText.textContent = progress.toFixed(2) + '%';
}

function fetchTaskProgress() {
```

```
        fetch('/task_progress')
            .then(response => response.json())
            .then(data => {
                updateTaskProgress(data.progress);
            })
            .catch(error => {
                console.error('Error:', error);
            });
    }

    // Update task progress every second
    setInterval(fetchTaskProgress, 1000);
</script>
<style>
    #drop-area {
        width: 300px;
        height: 200px;
        border: 2px dashed skyblue;
        text-align: center;
        vertical-align: middle;
        padding: 40px;
        margin: 0 auto;
    }

    #drop-area.highlight {
        background-color: lightgray;
        vertical-align: middle;
    }

    #drop-area {
        width: 300px;
        height: 200px;
        border: 2px dashed blue;
```

```
    text-align: center;
    vertical-align: middle;
    padding: 40px;
    margin: 0 auto;
}

/* CSS */
#drop-area {
    transition: border-color 0.3s ease-in-out;

}

#drop-area:hover {
    border-color: #D10024;
    cursor: pointer;
}



#task-progress-bar-container {
    width: 100%;
    height: 20px;
    background-color: lightgray;
    margin-top: 40px;
    margin: 0 auto;
}

#task-progress-bar {
    height: 100%;
    background-color: #E90024;
}

#task-progress-text {
```

```
            margin-top: 8px;
            font-size: 16px;
        }


    </style>
</head>



<body class="text-center superconteiner mb-0">

 <header>

    <div id="top-header">
        <div class="container">
            <ul class="header-links pull-right">
                <li><a href="/logout"><i class="fa fa-user-o"></i>Мій Акаунт ({{ username }})</a></li>
            </ul>
        </div>
    </div>

    <div id="header">
        <!-- container -->
        <div class="container">
            <div class="row">
                <!-- LOGO -->
                <div class="col-md-3 d-flex align-items-center justify-content-center">
                    <div class="header-logo">
                        <a href="#">
                            <img class="logo" src="./static/images/logo.png" alt="">
                        </a>
```

```
                </div>
            </div>

            <div class="col-md-6 d-flex align-items-center justify-content-center">
                <h1 class="centered-button" >Детектор поз йоги</h1>
            </div>
        </div>
    </div>
    <!-- container -->
  </div>

</header>

<div class="topbodyredline">
  <main class="form-signin">
    <div>
      {% with messages = get_flashed_messages() %}
       {% if messages %}
        <ul class="flash-messages">
         {% for message in messages %}
          <li class="success">{{message}}</li>
         {% endfor %}
        </ul>
       {% endif %}
      {% endwith %}

      <div id="drop-area">
        <h3 class="drop-area-text">Перетягніть сюди відео</h3>
      </div>

      <div id="task-progress-bar-container">
        <div id="task-progress-bar"></div>
      </div>
```

```
                <div id="task-progress-text"></div>

                <div class="margintop">
                  <a class="w-100 btn btn-lg" href="/results"><div
class="genbutton">Переглянути результат</div></a>
                </div>
                  <a class="w-100 btn btn-lg buttonlines" href="/download"><div
class="genbutton">Завантажити результат</div></a><br>



            </div>
          </main>
        </div>

        <footer id="footer">

          <div id="bottom-footer" class="section">
            <div class="container">
              <!-- row -->
              <div class="row">
                <div class="col-md-12 text-center">
                  <ul class="footer-payments">
                    <li><a href="#"><i class="fa fa-cc-visa"></i></a></li>
                    <li><a href="#"><i class="fa fa-credit-card"></i></a></li>
                    <li><a href="#"><i class="fa fa-cc-paypal"></i></a></li>
                    <li><a href="#"><i class="fa fa-cc-mastercard"></i></a></li>
                    <li><a href="#"><i class="fa fa-cc-discover"></i></a></li>
                    <li><a href="#"><i class="fa fa-cc-amex"></i></a></li>
                  </ul>
                  <span class="copyright">
```

```
                        <!-- Link back to Colorlib can't be removed. Template is licensed under
CC BY 3.0. -->
                        Copyright ©<script>document.write(new Date().getFullYear());</script>
All rights reserved

                            </span>
                        </div>
                    </div>
                        <!-- /row -->
                </div>
                <!-- /container -->
            </div>

        </footer>

        </body>
        </html>
```

Лістинг файлу result.html:

```
        <!DOCTYPE html>
        <html>
        <head>
            <title>Результати обробки</title>

            <link type="text/css" rel="stylesheet" href="static/css/style1.css"/>

            <link type="text/css" rel="stylesheet" href="static/css/style.css"/>

            <!-- Google font -->
            <link              href="https://fonts.googleapis.com/css?family=Montserrat:400,500,700"
rel="stylesheet">

            <!-- Bootstrap -->
```

```
        <link type="text/css" rel="stylesheet" href="static/css/bootstrapResults.min.css"/>

        <!-- Font Awesome Icon -->
        <link rel="stylesheet" href="static/css/font-awesome.min.css">

    </head>
    <body>

        <header>

            <div id="top-header">
                <div class="container">
                    <ul class="header-links pull-right">
                        <li><a href="/logout"><i class="fa fa-user-o"></i>Мій Акаунт ({{ username }})</a></li>
                    </ul>
                </div>
            </div>

            <div id="header">
                <!-- container -->
                <div class="container">
                    <div class="row">
                        <!-- LOGO -->
                        <div class="col-md-3 d-flex align-items-center justify-content-center">
                            <div class="header-logo">
                                <a href="#">
                                    <img class="logo" src="./static/images/logo.png" alt="">
                                </a>
                            </div>
                        </div>

                        <div class="col-md-6 d-flex align-items-center justify-content-center">
```

```
                <h1 class="centered-button" >Детектор поз йоги</h1>
            </div>
        </div>
    </div>
    <!-- container -->
</div>

</header>



<table>
  <thead>
    <tr>
      <th class="thcenter">Асана</th>
      <th class="thcenter">Початок</th>
      <th class="thcenter">Кінець</th>
      <th class="thcenter">Ідентифіковане фото та його відповідник</th>
    </tr>
  </thead>
  <tbody>
    {% for index, row in df_result.iterrows() %}
    <tr>
      <td class="tdcenter">{{ row['asana'] }}</td>
      <td class="tdcenter">{{ row['start'] }}</td>
      <td class="tdcenter">{{ row['end'] }}</td>
      <td class="tdcenter">
        {% if base_images %}
          <img src="{{ video_images[index] }}" alt="Video Image">

        {% else %}
          <p>No base image available</p>
        {% endif %}
```

```
                    {% if video_images %}
                        <img src="{{ base_images[index] }}" alt="Base Image">
                    {% else %}
                        <p>No video image available</p>
                    {% endif %}
                </td>
            </tr>
            {% endfor %}
        </tbody>
    </table>

    <div class="centered-button">
        <a class="w-100 btn btn-lg" href="{{ url_for('index') }}">Повернутися на головну сторінку</a>
    </div>

    <footer id="footer">

        <div id="bottom-footer" class="section">
            <div class="container">
                <!-- row -->
                <div class="row">
                    <div class="col-md-12 text-center">
                        <ul class="footer-payments">
                            <li><a href="#"><i class="fa fa-cc-visa"></i></a></li>
                            <li><a href="#"><i class="fa fa-credit-card"></i></a></li>
                            <li><a href="#"><i class="fa fa-cc-paypal"></i></a></li>
                            <li><a href="#"><i class="fa fa-cc-mastercard"></i></a></li>
                            <li><a href="#"><i class="fa fa-cc-discover"></i></a></li>
                            <li><a href="#"><i class="fa fa-cc-amex"></i></a></li>
                        </ul>
                        <span class="copyright">
```

```
                        </span>
                    </div>
                </div>
                    <!-- /row -->
            </div>
            <!-- /container -->
        </div>

    </footer>
</body>
</html>
```

Додаток Г – Графічна частина

ГРАФІЧНА ЧАСТИНА

Розробка системи розпізнавання позиції людини на основі нейронної мережі

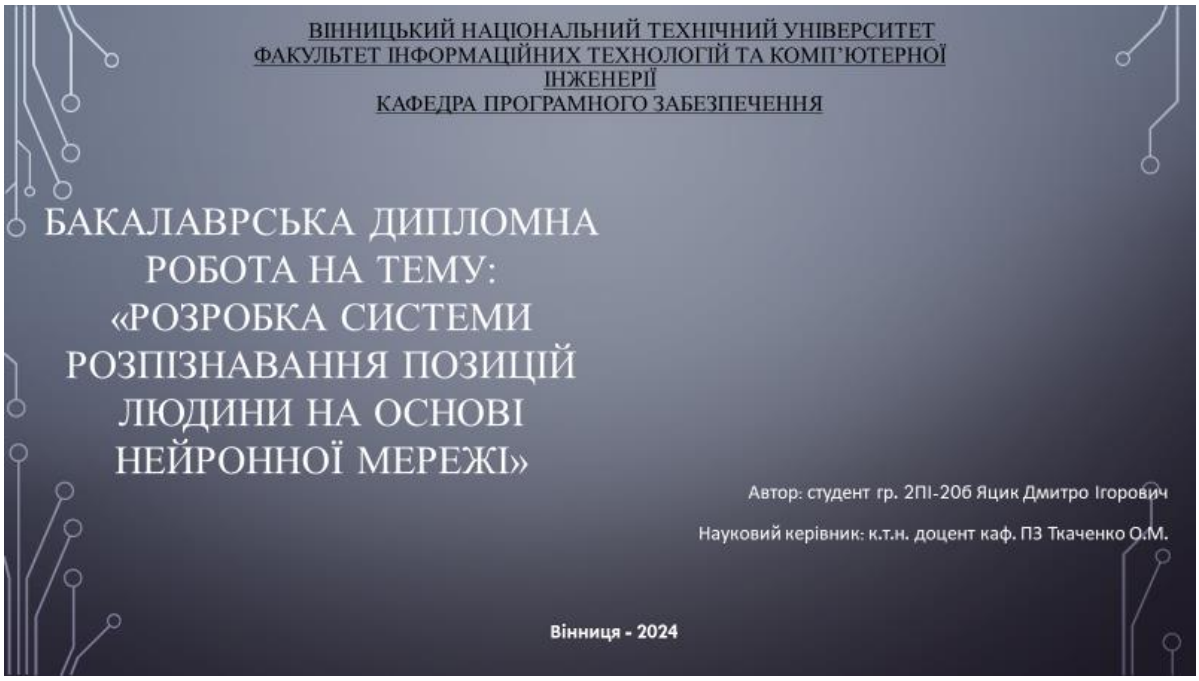

Рисунок Г.1 – Титульний слайд

Рисунок Г.2 – Мета, предмет на об’єкт дослідження

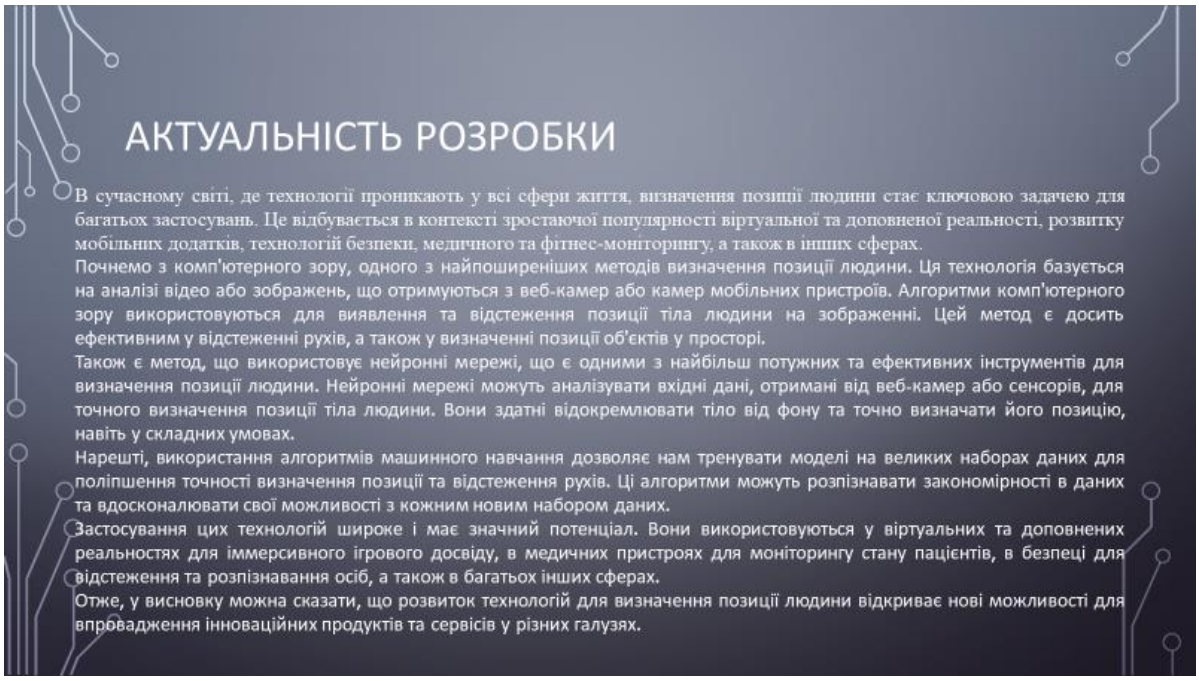

Рисунок Г.3 – Актуальність розробки

Рисунок Г.4 – Аналіз аналогів

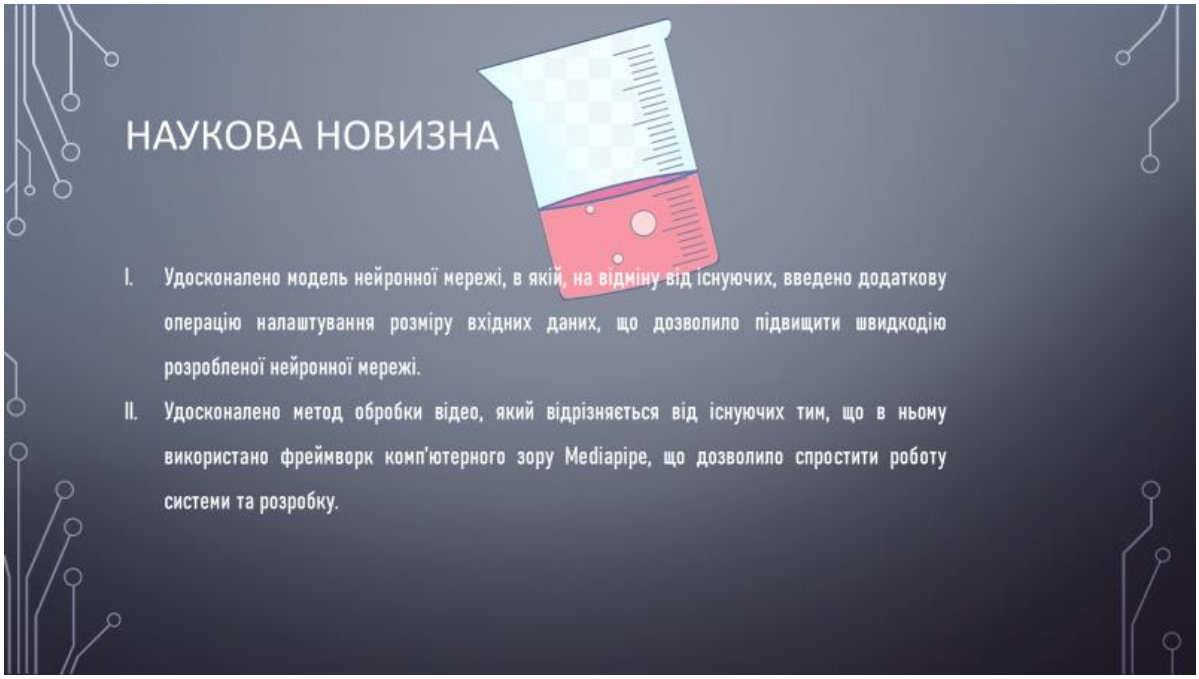

Рисунок Г. 5 – Наукова новизна

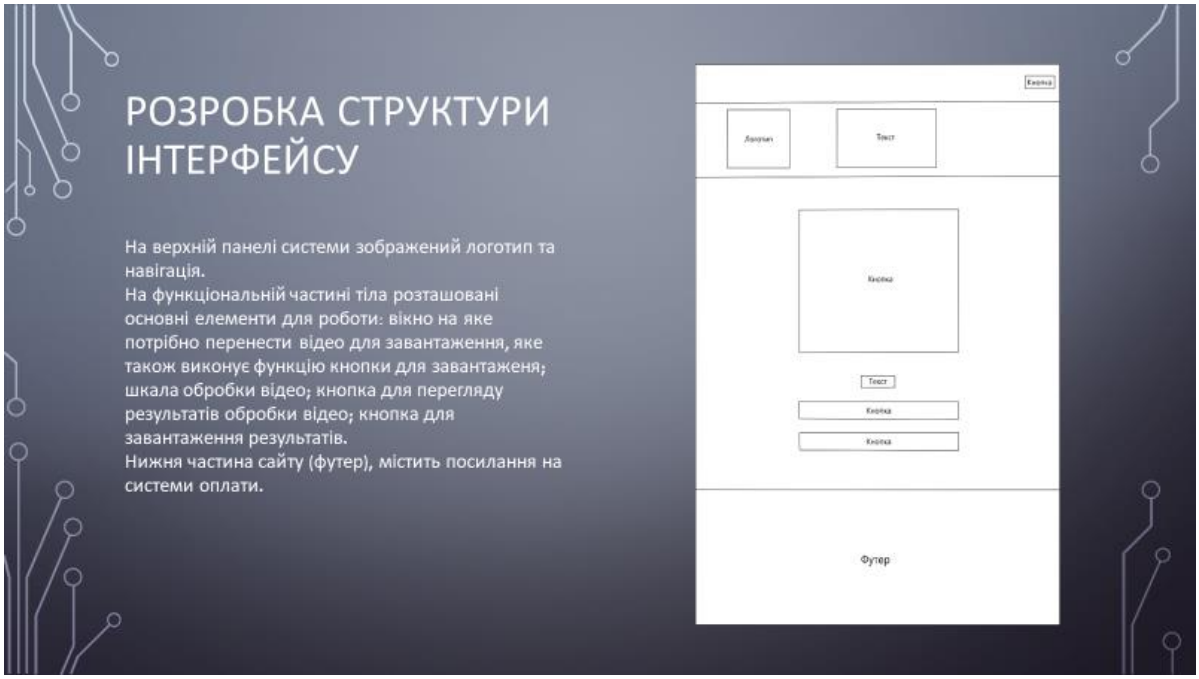

Рисунок Г.6 – Розробка структури інтерфейсу

Рисунок Г.7 – UML-діаграма класів моделі нейронної мережі

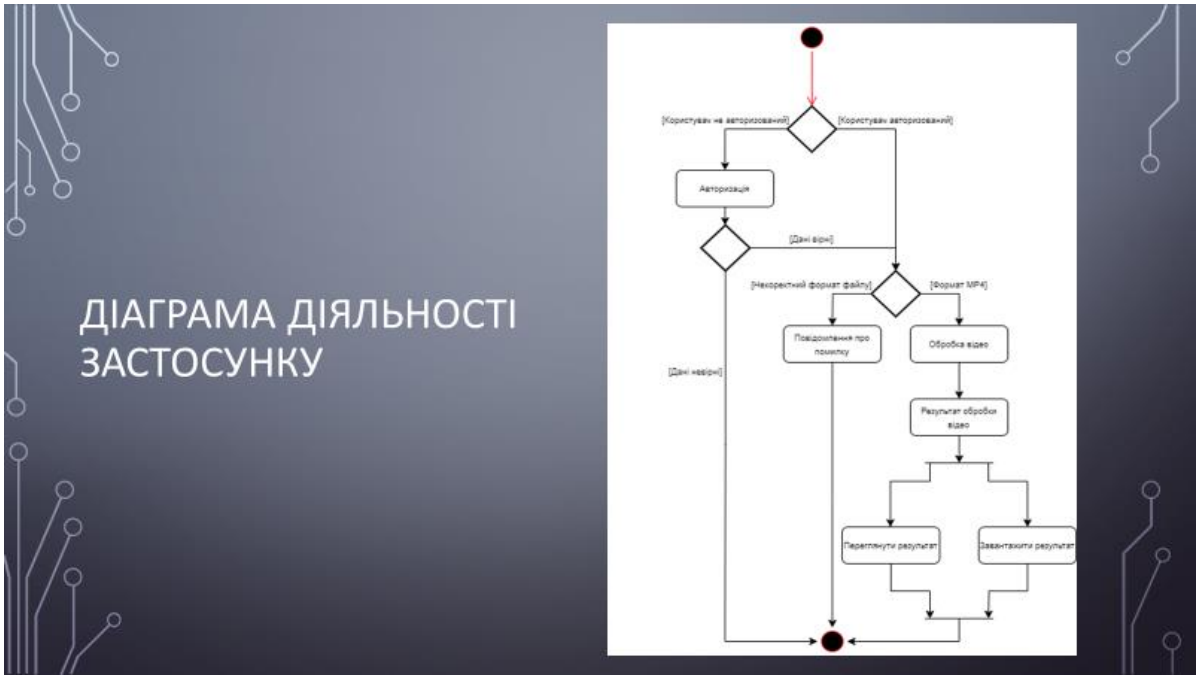

Рисунок Г.8 – Діаграма діяльності застосунку

Рисунок Г.9 – Блок-схема обробки відео

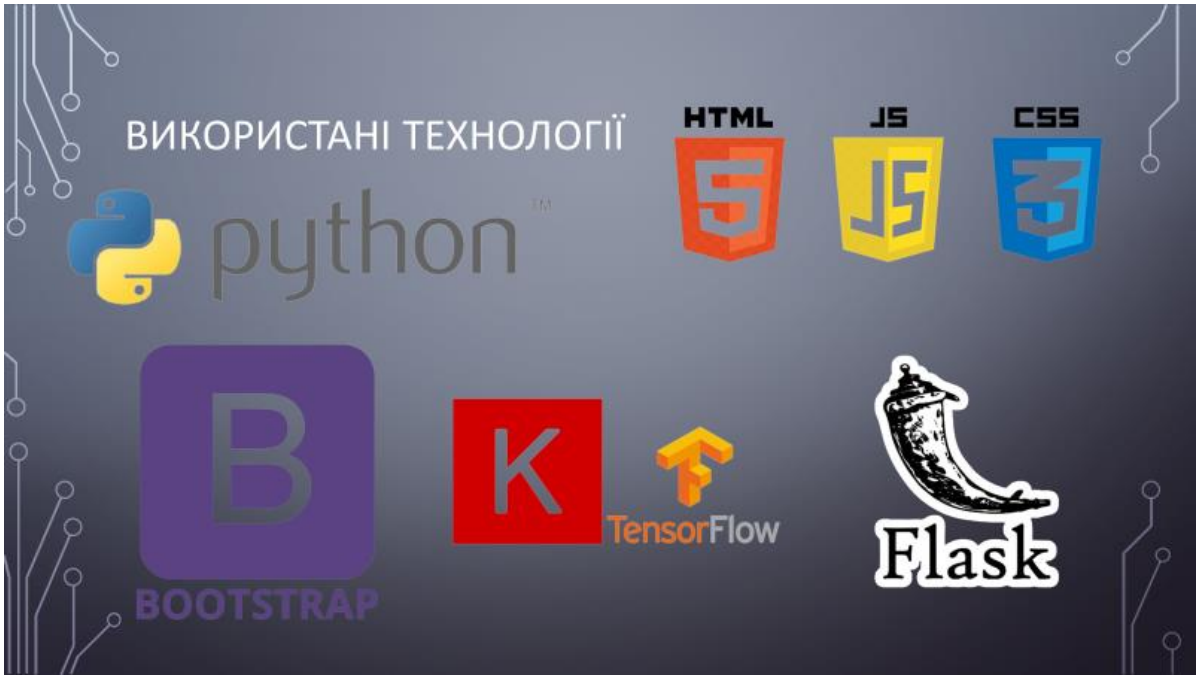

Рисунок Г.10 – Використані технології

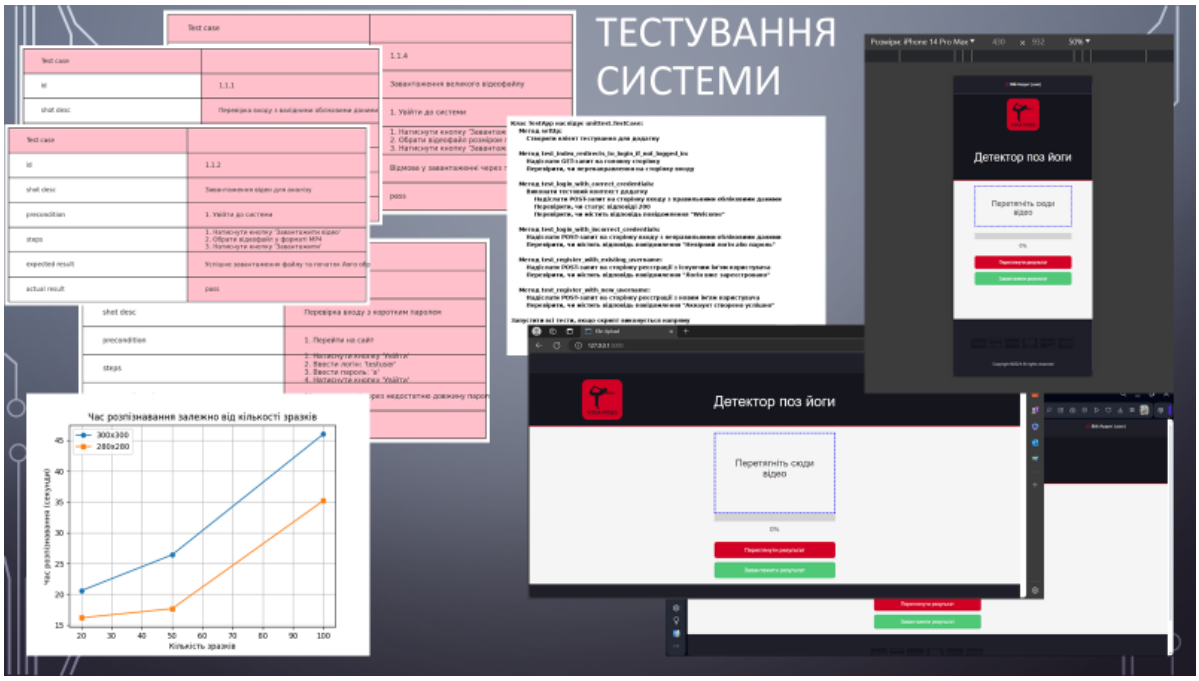

Рисунок Г.11 – Тестування системи

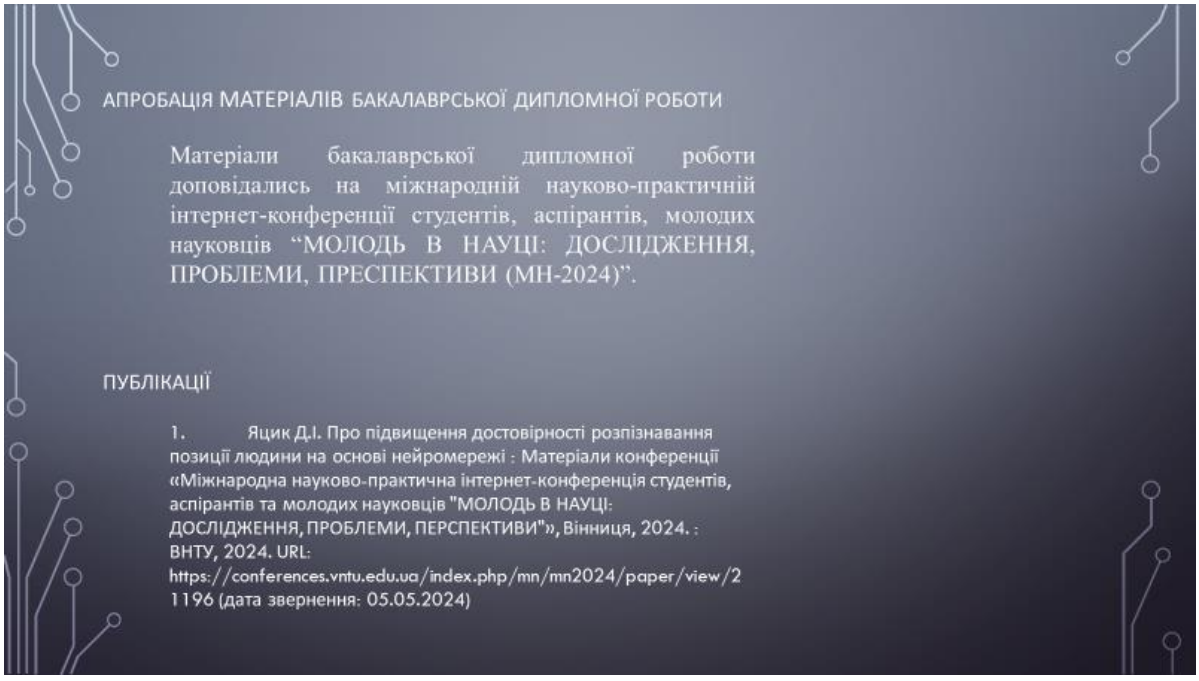

Рисунок Г.12 – Апробація та публікації

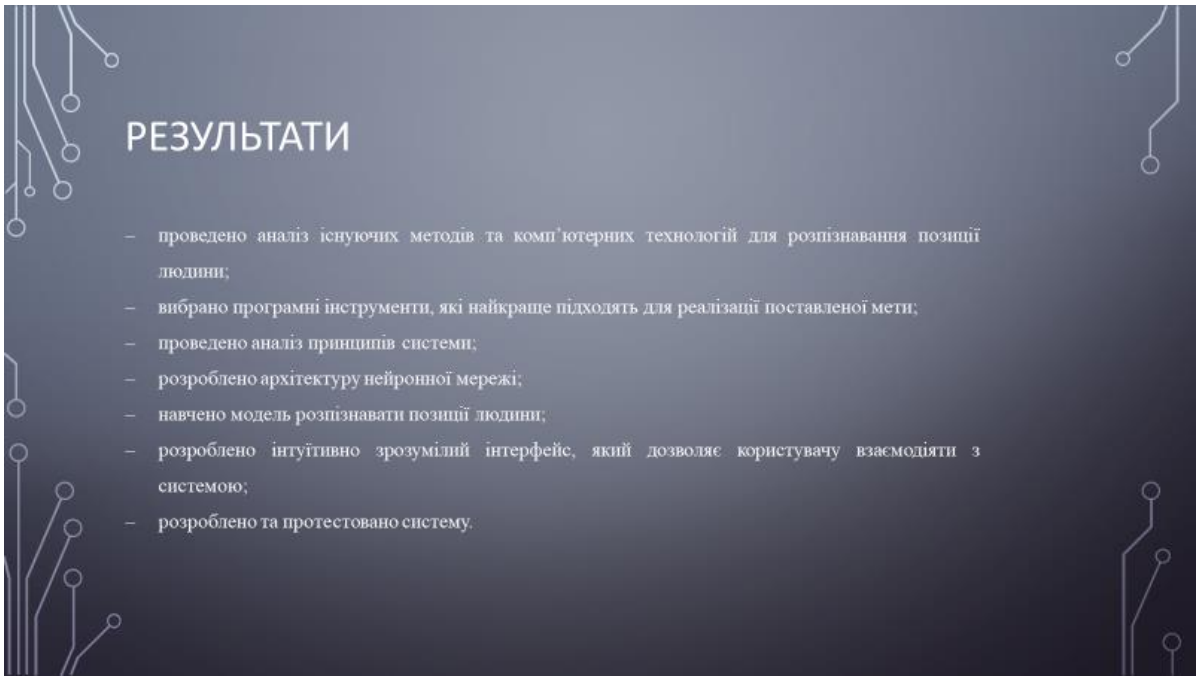

Рисунок Г.13 – Результати

Рисунок Г.14 – Кінцевий слайд