СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет *Інформаційних технологій та електроніки*

Кафедра *Інформаційних технологій та програмування*

ПОЯСНЮВАЛЬНА ЗАПИСКА

до магістерської дипломної роботи

Магістр
(освітньо-кваліфікаційний рівень)

на тему *Система жестового управління для презентацій*

Виконав: *студент 2 курсу, групи ІСТ-23ДМ*
126 «Інформаційні системи та технології»
(шифр і назва спеціальності)

Слюсаренко А. С.
(ініціали і прізвище)

Керівник *Лифар В. О.*
(ініціали і прізвище)

Рецензент *Меняйленко О.С.*
(ініціали і прізвище)

Київ - 2024

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет *Інформаційних технологій та електроніки*
Кафедра *Інформаційних технологій та програмування*
Освітньо-кваліфікаційний рівень *Магістр*
Спеціальність *126 Інформаційні системи та технології*
(шифр і назва спеціальності)

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТП
_____д.т.н., проф., Захожай О. І.
(підпис)
“___” _____2024 року

ЗАВДАННЯ

На магістерську дипломну роботу студента
Слюсаренко Андрій Сергійович
1. Тема роботи *Система жестового управління для презентацій*

Керівник роботи *доц. д.т.н. Лифар Володимир Олексійович*,
(прізвище, ім’я, по-батькові, науковий ступінь, вчене звання)
Затверджений наказом університету від “*6*” *12* 2024 року №*361/15.15-С*
2. Строк подання студентом роботи *14 грудня 2024 р.*
3. Вихідні дані роботи *Матеріали науково-дослідницької практики, науково-методична література, дані інтернет мережі*
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

4.1 Вступ

4.2 Аналітичний огляд питання.

4.3 Основна частина, де досліджено та проаналізовано методи для реалізації проекту, а також сформовано вирішення для розв’язку поставленої задачі.

4.4 Практична частина, в якій зроблено огляд технологій та інструментарію, які використовуються для реалізації проекту.

4.5 Висновки

4.6 Перелік використаних джерел.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслеників)_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розділ 1. Аналіз основних підходів до розпізнавання жестів рук	Доц., д.т.н. Лифар В.О.		
Розділ 2. Аналіз технологій інтеграції функцій розпізнавання жестів у веб-додатки	Доц., д.т.н. Лифар В.О.		
Розділ 3. Розробка алгоритму розпізнавання жестів у реальному часі	Доц., д.т.н. Лифар В.О.		
Розділ 4. Розробка веб-системи перегляду презентацій	Доц., д.т.н. Лифар В.О.		

7. Дата видачі завдання *5 жовтня 2024 р.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання кваліфікаційної випускної роботи	Строк виконання етапів	Примітка
1.	Одержання завдання на виконання роботи	20.10.2024	
2.	Укладання і погодження з керівником плану і етапів виконання роботи	24.10.2024	
3.	Узагальнення даних літературних джерел	28.10.2024	
4.	Аналіз шляхів виконання завдання. Вибір і погодження з керівником оптимального шляху виконання завдання	01.11.2024	
5.	Аналіз технічних засобів та існуючих систем	07.11.2024	
6.	Реалізація практичної частини завдання	24.11.2024	
7.	Укладання, оформлення та погодження пояснювальної записки з керівником	11.12.2024	
8.	Надання пояснювальної записки на кафедру	12.12.2024	
9.	Підготовка доповіді та презентації	12.12.2024	

Студент _____ *Слюсаренко А. С.*
(підпис) (ініціали і прізвище)

Керівник роботи _____ *Лифар В.О.*
(підпис) (ініціали і прізвище)

РЕФЕРАТ

Магістерська дипломна робота: 46 стор., 1 табл., 13 рис., 8 джерел.

Об’єкт дослідження – методи та технології розпізнавання жестів рук та їх застосування для інтерактивного управління.

Мета роботи – дослідження сучасних підходів до розпізнавання жестів і їх інтеграції у веб-системи, а також створення рішення для зручного управління презентаціями.

Проаналізовано підходи до розпізнавання жестів рук, визначено їх можливості. Розроблено систему, яка дозволяє здійснювати керування презентаціями за допомогою жестів у реальному часі. Особливу увагу приділено вирішенню питань, пов’язаних із точністю розпізнавання та стабільністю роботи. Також було розроблено інтерфейс для взаємодії з користувачем і забезпечено тестування системи.

Для досягнення поставленої мети дипломна робота поділена на такі завдання:

1. Дослідити основні підходи до розпізнавання жестів рук.
2. Провести порівняльний аналіз технологій, які забезпечують інтеграцію функцій розпізнавання в веб-додатки.
3. Розробити алгоритм, що забезпечує точність і швидкість розпізнавання жестів у реальному часі.
4. Розробити веб-систему перегляду презентацій та інтегрувати розроблений алгоритм керування ними

Ключові слова: РОЗПІЗНАВАННЯ ЖЕСТІВ, МЕТОДИ АНАЛІЗУ РУХІВ, ІНТЕРАКТИВНЕ УПРАВЛІННЯ, ВЕБ-ДОДАТКИ, АРХІТЕКТУРА СИСТЕМ, JAVA, CLIENT-SERVER MODEL, SPRING, SQL, TYPESCRIPT, КОМП’ЮТЕРНИЙ ЗІР.

ABSTRACT

Master's thesis: 45 pages, 1 table, 13 pictures, 8 sources.

Object of Research: Methods and technologies for hand gesture recognition and their application for interactive control.

Objective of the Study: To research modern approaches to gesture recognition and their integration into web systems, as well as to create a solution for convenient presentation control.

The study analyzed approaches to hand gesture recognition and identified their capabilities. A system was developed to enable real-time presentation control using gestures. Special attention was given to addressing issues related to recognition accuracy and system stability. Additionally, a user interface for interaction was designed, and the system was tested.

To achieve the research objective, the diploma project is divided into the following tasks:

1. Investigate the main approaches to hand gesture recognition.
2. Conduct a comparative analysis of technologies that enable the integration of recognition functions into web applications.
3. Develop an algorithm that ensures accuracy and speed of gesture recognition in real time.
4. Develop a web-based presentation system and integrate the designed gesture control algorithm.

Keywords: GESTURE RECOGNITION, MOTION ANALYSIS METHODS, INTERACTIVE CONTROL, WEB APPLICATIONS, SYSTEM ARCHITECTURE, JAVA, CLIENT-SERVER MODEL, SPRING, SQL, TYPESCRIPT, COMPUTER VISION.

ЗМІСТ

ВСТУП

Сучасний розвиток технологій сприяє впровадженню нових способів взаємодії користувачів із цифровими системами. Одним із таких інноваційних рішень є використання жестів рук для управління пристроями та програмними додатками. Безконтактне керування жестами забезпечує інтуїтивність і простоту у використанні, що відкриває перспективи для його впровадження в різних сферах, зокрема в управлінні презентаціями.

Традиційні інструменти, такі як миша або клавіатура, під час презентацій можуть обмежувати рухи виступаючого та відволікати від основного матеріалу. Використання жестів рук як засобу керування дозволяє виступаючому зосередитися на контенті, забезпечуючи при цьому швидкий і зручний доступ до функцій зміни слайдів та взаємодії з аудиторією.

У цій роботі досліджуються методи розпізнавання жестів та їх інтеграція у веб-системи. Основна мета полягає у створенні веб-додатка, що дозволяє ефективно керувати презентаціями за допомогою жестів рук. Здійснено аналіз технологій для розпізнавання рухів, а також визначено підходи, які найкраще відповідають завданням інтерактивного управління.

Результатом роботи є система, що поєднує алгоритми розпізнавання жестів із веб-функціоналом для інтерактивного керування презентаціями. Створене рішення демонструє зручність та ефективність сучасних підходів до взаємодії з цифровими системами..

РОЗДІЛ 1. АНАЛІЗ ОСНОВНИХ ПІДХОДІВ ДО РОЗПІЗНАВАННЯ ЖЕСТІВ РУК

Жести, як форма невербального спілкування, є одним із найдавніших способів передачі інформації. У контексті сучасних технологій взаємодії людини з комп'ютером жест визначається як цілеспрямований рух руки, пальців або інших частин тіла, який може бути розпізнаний та інтерпретований системою як команда. Використання жестів як інструменту управління відкриває нові можливості для створення інтуїтивних і ефективних інтерфейсів, що сприяють покращенню зручності користувацького досвіду.

На сьогодні технології розпізнавання жестів набули широкого застосування у таких галузях, як:

- Віртуальна та доповнена реальність: жести забезпечують природний спосіб маніпулювання цифровими об'єктами, що робить взаємодію більш органічною та ефективною.
- Презентаційні системи та освітні платформи: управління слайдами або іншими елементами презентації за допомогою жестів підвищує комфортність роботи та дозволяє сконцентруватися на змісті.
- Робототехніка та автоматизація: інтерпретація жестів забезпечує простий та інтуїтивний спосіб взаємодії з роботизованими системами, що є особливо актуальним у промисловості та медицині.
- Системи «розумного дому»: технології жестового управління дозволяють керувати пристроями побутового призначення, забезпечуючи безконтактний і зручний доступ до функцій.

Завдання розпізнавання жестів включає не лише визначення просторового руху руки, але й його інтерпретацію відповідно до контексту. Зважаючи на широкий спектр можливостей застосування розпізнавання жестів, ця технологія постійно вдосконалюється. Особливу увагу приділяють пошуку нових алгоритмів, що забезпечують підвищену точність та стабільність роботи, а також розробці спеціалізованих пристроїв, здатних ефективно реєструвати жести в реальному часі. Аналіз існуючих підходів та інструментів для розпізнавання жестів є важливим кроком для розробки інноваційних рішень, здатних відповідати сучасним вимогам інтерактивних систем управління.

1.1 Огляд методів розпізнавання жестів рук

Методи розпізнавання жестів рук класифікуються на два основні підходи: контактні та безконтактні, кожен із яких має свої особливості, переваги та обмеження.

Контактні методи ґрунтуються на використанні спеціалізованих пристроїв, які забезпечують прямий фізичний контакт із сенсорами або їхню близькість до тіла. Ці пристрої реєструють параметри руху, положення руки, силу натиску чи згину. Контактний підхід вважається одним із найбільш точних завдяки можливості безпосередньо фіксувати навіть найменші рухи.

Одним із найпоширеніших варіантів контактних методів є сенсорні рукавички. Вони обладнані датчиками згину, гіроскопами та акселерометрами, що дозволяють визначати положення та рух пальців і кисті в просторі. Наприклад, CyberGlove використовується для роботи в системах віртуальної реальності, анімації та робототехніці. Завдяки детальному визначенню кутових змін у суглобах, такі рукавички є незамінними в задачах, що вимагають максимальної точності рухів. Ще одним прикладом контактних пристроїв є електроміографічні браслети (EMG), які реєструють електричну активність м’язів руки під час виконання жестів. Такі пристрої, як Myo Armband, використовуються для управління комп’ютерними системами, роботами та навіть дронами. Висока чутливість до м’язових сигналів дозволяє цим пристроям інтерпретувати жести навіть до початку видимого руху, що значно розширює їхні можливості у сфері адаптивних технологій.

Контактні методи також охоплюють пристрої із тактильними сенсорами, які вимірюють силу натиску чи точку контакту. Вони застосовуються у сфері робототехніки, коли потрібно забезпечити точне маніпулювання об’єктами, або у віртуальних симуляторах, де користувачі можуть "відчувати" взаємодію з віртуальним середовищем.

Попри їхні переваги, контактні методи мають певні обмеження. Їхнє використання може бути ускладнене через високу вартість обладнання, необхідність носіння додаткових пристроїв, таких як рукавички або браслети, та можливе обмеження природності рухів користувача. Це робить їх менш зручними для тривалого застосування або масового впровадження.

Безконтактні методи розпізнавання жестів рук є одним із найперспективніших напрямів розвитку технологій взаємодії людини з комп'ютером. Ці методи базуються на застосуванні камер та алгоритмів обробки візуальної інформації для ідентифікації жестів у реальному часі. Вони відкривають нові можливості для створення природних і зручних інтерфейсів, що дозволяють управляти системами без фізичного контакту з пристроями. Основою таких методів є комп'ютерний зір, який забезпечує автоматичне сприйняття й інтерпретацію зображень для аналізу рухів руки та їхнього подальшого перекладу в команди.

1.2 Основи комп'ютерного зору в розпізнаванні жестів

Комп'ютерний зір є наукою та технологією, що вивчає методи обробки та інтерпретації зображень для автоматизації процесів розпізнавання об'єктів, їхнього положення та руху. У задачах безконтактного розпізнавання жестів комп'ютерний зір дозволяє системі визначати рухи руки, аналізувати їхні характеристики та класифікувати на основі попередньо визначених шаблонів або навчання. Процес розпізнавання жестів через комп'ютерний зір зазвичай складається з кількох ключових етапів:

- Захоплення зображень або відео. Камери фіксують рухи рук користувача. Для забезпечення високої якості розпізнавання використовуються як звичайні RGB-камери, так і камери із сенсорами глибини, які можуть захоплювати тривимірну інформацію.
- Обробка даних і сегментація. На цьому етапі виділяється область з рукою шляхом сегментації фону. Використовуються алгоритми кольорової сегментації, визначення контурів або відстеження рухів.

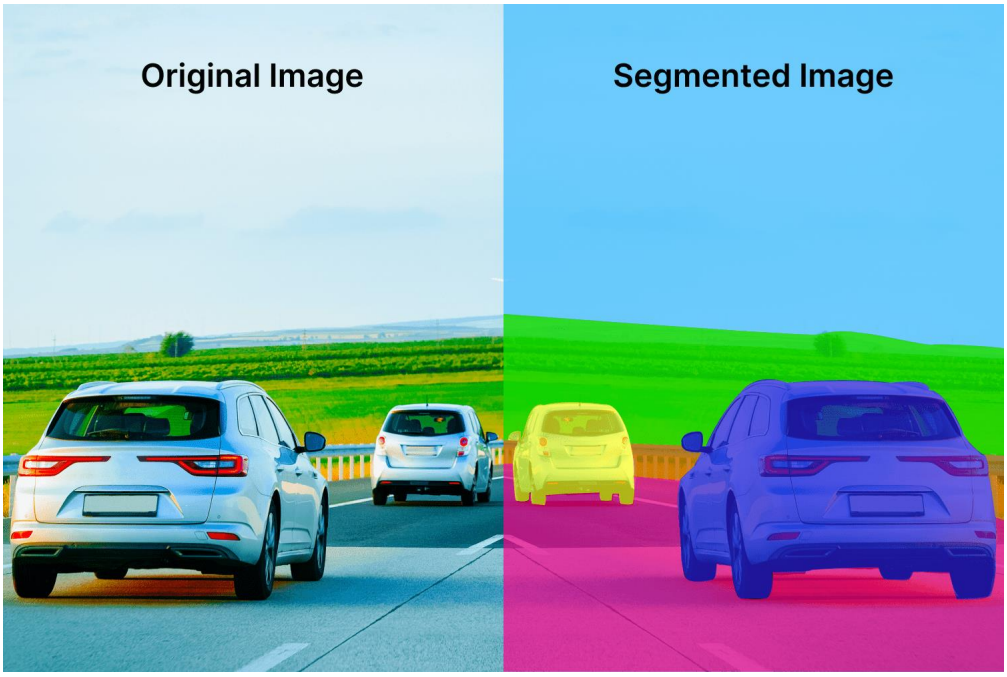

Рис 1.1 Сегментація візуального зображення

- Виділення ознак. Система визначає ключові характеристики руки, такі як форма, розташування пальців, контури долоні або положення ключових точок.
- Класифікація жестів. Виділені ознаки аналізуються для зіставлення з базою даних відомих жестів або обробляються моделями машинного навчання для розпізнавання.

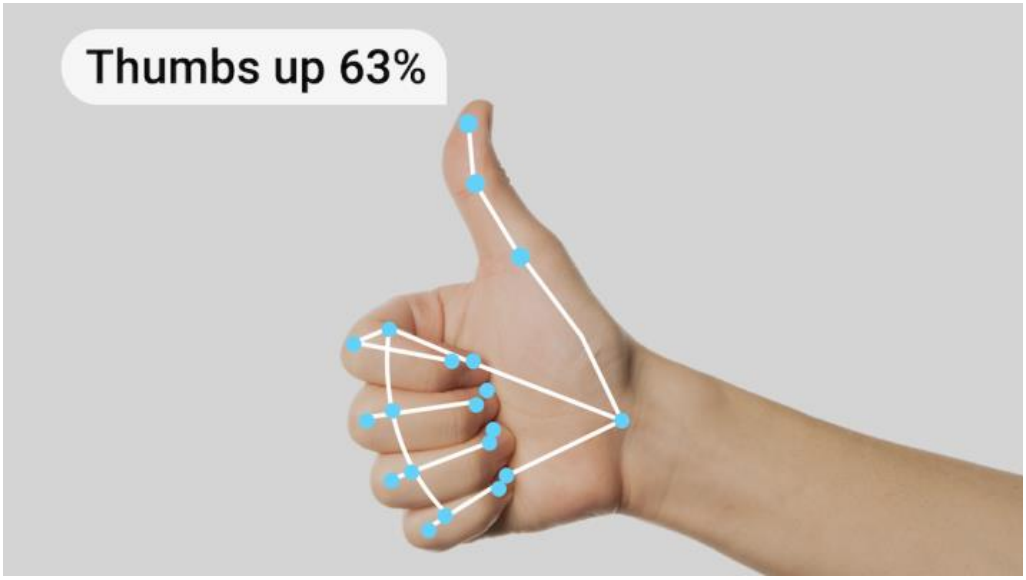

Рис 1.2 Виділення ключових точок руки та класифікація жесту

Комп’ютерний зір дозволяє автоматизувати ці процеси, використовуючи як класичні алгоритми (наприклад, сегментація з використанням кольору або текстури), так і сучасні методи глибокого навчання, які значно розширюють можливості розпізнавання жестів.

Використання RGB-камер для аналізу жестів

RGB-камери є основним інструментом для багатьох систем розпізнавання жестів. Вони дозволяють захоплювати зображення руки у двовимірному просторі, яке потім аналізується для визначення жесту. У системах із RGB-камерами часто застосовуються такі методи, як сегментація на основі кольору шкіри, виявлення контурів або використання шаблонів для визначення положення пальців. Однак робота з RGB-камерами має певні обмеження. Зокрема, ефективність алгоритмів залежить від умов освітлення, якості зображення та складності фону. Наприклад, кольори, схожі на тон шкіри, можуть спричинити помилкове розпізнавання, а низька роздільна здатність камер ускладнює ідентифікацію жестів.

Камери з сенсорами глибини

Використання камер із сенсорами глибини, таких як Microsoft Kinect, значно підвищує точність безконтактного розпізнавання жестів. Ці камери захоплюють не лише кольорові зображення, але й інформацію про глибину кожної точки сцени. Це дозволяє будувати тривимірні моделі рухів рук і значно полегшує визначення складних жестів. Завдяки тривимірному аналізу система може визначати положення руки у просторі, обчислювати кути між пальцями, ідентифікувати нахили кисті або рухи, які вимагають високої точності. Наприклад, Kinect використовується для керування роботами або у віртуальних середовищах, де жестове управління є ключовим елементом взаємодії.

Крім того, сенсори глибини забезпечують стабільність роботи навіть у складних умовах, таких як змінне освітлення або фони з великою кількістю деталей. Основним обмеженням цього підходу є висока вартість камер із глибоким сенсором, а також необхідність у потужних процесорах для обробки даних у реальному часі.

1.3 Глибоке навчання у розпізнаванні жестів

Застосування глибоких нейронних мереж у безконтактному розпізнаванні жестів відкриває нові можливості для підвищення точності та адаптивності систем. Глибоке навчання дозволяє системам автоматично виділяти релевантні ознаки з відеопотоків, що робить їх менш залежними від попередньої обробки зображень. Наприклад, згорткові нейронні мережі (CNN) використовуються для

аналізу статичних зображень, тоді як рекурентні нейронні мережі (RNN) ефективні для роботи з динамічними жестами, які потребують урахування змін у часі.

Згорткові нейронні мережі (Convolutional Neural Networks, CNN) є спеціалізованим типом глибоких нейронних мереж, розробленим для аналізу даних із просторовою структурою, таких як зображення. Вони досягли значного успіху в задачах комп'ютерного зору завдяки здатності автоматично виділяти ознаки та адаптуватися до різноманітних варіантів вхідних даних.

Архітектура CNN: основні компоненти

1. Вхідний шар

Першим компонентом CNN є вхідний шар, який приймає дані у вигляді багатовимірного тензора. Наприклад, кольорове зображення представляється тривимірним тензором розміром H×W×C, де:

H - висота зображення;

W - ширина зображення;

C - кількість каналів (зазвичай 3 для RGB-зображень).

Перед передачею до моделі зображення часто нормалізується, щоб значення пікселів лежали в діапазоні від 0 до 1 або від -1 до 1.

Нормалізація значень пікселів для вхідного зображення може бути узагальнена за допомогою формули:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

де:

x - значення пікселя у вихідному діапазоні (наприклад, від 0 до 255);

min(x) - мінімальне значення пікселя у вихідному діапазоні (зазвичай 0 для 8-бітних зображень);

max(x) - максимальне значення пікселя у вихідному діапазоні (зазвичай 255 для 8-бітних зображень);

x′ - нормалізоване значення пікселя

1. Масштабування до діапазону [0,1]

Цей метод використовується для масштабування інтенсивностей пікселів, які лежать у діапазоні [0,255] (min(x) = 0, max(x) =255), до діапазону [0,1].

Формула для нормалізації:

$$x' = \frac{x}{255}$$

де

x - значення пікселя до нормалізації,

x′ - нормалізоване значення.

Цей підхід є простим і ефективним, а також широко використовується в задачах класифікації зображень, оскільки нейронні мережі краще працюють із малими числовими значеннями.

2. Масштабування до специфічних діапазонів

Деякі архітектури нейронних мереж вимагають нормалізації вхідних даних до діапазону [−1,1]. Тоді у формулу вводяться ще два параметри newMin і newMax — нові межі діапазону, до якого необхідно привести значення

Формула:

$$x' = (\text{newMax} - \text{newMin}) \frac{x - \min(x)}{\max(x) - \min(x)} + \text{newMin}$$

При newMin = -1 та newMax = 1, формула спрощується до:

$$x' = \frac{x}{127.5} - 1$$

Цей метод дозволяє забезпечити збалансованість позитивних і негативних значень, що сприяє більш ефективному навчанню в мережах із нелінійними функціями активації.

2. Згортковий шар (Convolutional Layer)

Це ключовий компонент CNN, який виконує згортку - операцію, що дозволяє виділяти локальні ознаки зображення, такі як краї, кути або текстури.

Фільтри (ядра згортки): це невеликі матриці, переміщуються (ковзають) по всьому зображенню. Кожен фільтр виявляє певну ознаку, наприклад, лінії контурів горизонтальні або вертикальні.

Карта ознак (Feature Map): результат згортки, що містить інформацію про виділені ознаки. Наприклад, після застосування 16 фільтрів до одного зображення утворюється 16 карт ознак.

Параметри згортки:

Крок (stride) визначає, як далеко фільтр переміщується кожного разу.

Менший крок дає більш детальну карту ознак, але збільшує обчислювальну складність.

Доповнення (padding) додає рамку нулів навколо зображення, щоб зберегти розміри карти ознак після згортки.

3. Функція активації (Activation Function)

Після згорткової операції застосовується нелінійна функція активації для додання мережі здатності моделювати складні залежності.

Найпоширенішою є ReLU (Rectified Linear Unit):

$$f(x) = \max(0, x)$$

Ця функція замінює всі від'ємні значення на нулі, залишаючи тільки позитивні.

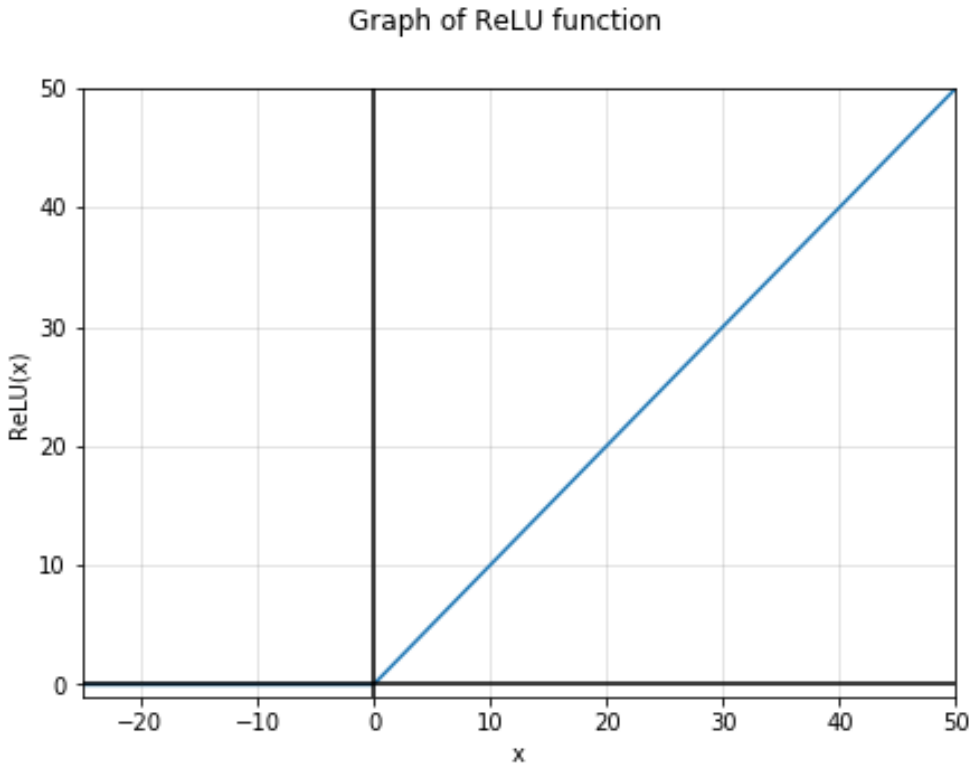

Рис 1.3 Графік функції ReLU

4. Шар підвибірки (Pooling Layer)

Шар підвибірки використовується для зменшення розмірності карти ознак, що зменшує обчислювальні витрати та знижує ризик перенавчання.

- Максимальне підвибірковування (Max Pooling): у кожному регіоні вибирається максимальне значення. Це дозволяє зберегти найбільш релевантні ознаки.
- Середнє підвибірковування (Average Pooling): обчислюється середнє значення для кожного регіону.

5. Нормалізація (Batch Normalization)

Цей компонент нормалізує вихід кожного шару, щоб зменшити вплив зміни значень під час навчання. Batch Normalization прискорює навчання та допомагає уникнути проблеми зникнення градієнтів у глибоких мережах.

6. Повнозв'язний шар (Fully Connected Layer)

Після кількох згорткових і підвибіркових шарів дані перетворюються в одномірний вектор (flattening) і передаються до повнозв'язних шарів. Ці шари виконують функцію класифікації, інтегруючи інформацію, виділену на попередніх етапах.

7. Вихідний шар

Остаточний результат формується вихідним шаром, який повертає ймовірності приналежності вхідних даних до певних класів. Для класифікаційних задач часто застосовується функція Softmax, яка перетворює виходи мережі на ймовірності:

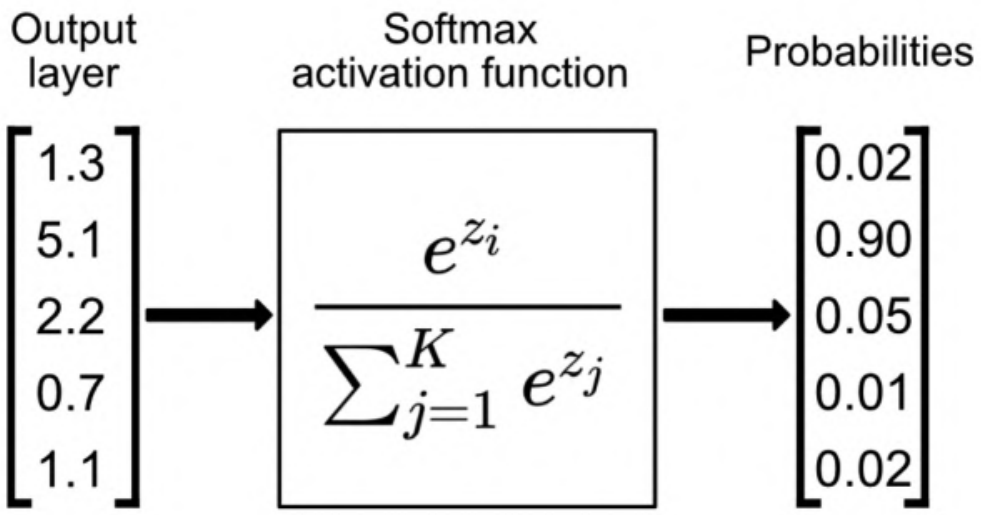

Рис 1.4 функція Softmax

Тут z_i представляє вхідне значення для функції Softmax для класу i, а знаменник є сумою експонент усіх сирих значень (оцінок) класів у вихідному шарі.

Принцип роботи CNN

Робота CNN складається з кількох етапів:

- Вхід зображення. Зображення подається до вхідного шару у вигляді

тензора.

- Згорткові операції. Кілька згорткових шарів виділяють ознаки різного рівня. Перший шар зазвичай виявляє базові елементи, такі як краї та текстури. Глибші шари визначають складніші структури, такі як форми чи об’єкти.
- Зменшення розмірності. Підвибіркові шари знижують розмірність, узагальнюючи інформацію.
- Класифікація. Повнозв’язні шари обробляють виділені ознаки та визначають клас об’єкта.

Навчання CNN

Навчання CNN відбувається через зворотне поширення похибки (Backpropagation):

Прямий прохід. Дані проходять через мережу, і модель генерує прогноз.

Обчислення похибки. Визначається різниця між прогнозом і фактичним значенням, наприклад, за допомогою функції втрат (Cross-Entropy Loss).

Зворотний прохід. Градієнти похибки поширюються назад через шари, коригуючи ваги фільтрів і нейронів.

Оновлення ваг. Алгоритм градієнтного спуску оновлює параметри мережі, зменшуючи похибку.

Цей процес повторюється протягом кількох епох, поки модель не досягне прийнятної точності. Важливо, що системи на основі глибокого навчання можуть адаптуватися до широкого спектра завдань, включаючи нові або нетипові жести, завдяки навчанню на великих наборах даних.

1.4 Висновки розділу

У межах розділу було проаналізовано основні підходи до розпізнавання жестів рук, включаючи контактні та безконтактні методи. Контактні методи, які використовують сенсорні пристрої, такі як рукавички або браслети, забезпечують високу точність, але мають обмеження через вартість і необхідність фізичного контакту. Безконтактні методи, що базуються на аналізі візуальної інформації за допомогою камер і алгоритмів комп’ютерного зору, є більш доступними та універсальними, хоча їх ефективність залежить від умов експлуатації, таких як освітлення чи якість обладнання.

Комп’ютерний зір є основою безконтактних методів, забезпечуючи автоматичну інтерпретацію зображень через етапи сегментації, виділення ознак і класифікації жестів. Використання камер із сенсорами глибини дозволяє створювати тривимірні моделі рухів, що підвищує точність і стабільність роботи систем. Глибокі нейронні мережі, зокрема згорткові (CNN), відіграють ключову роль у сучасних системах розпізнавання жестів. Вони автоматично виділяють релевантні ознаки з даних, забезпечуючи високу точність і ефективність роботи, особливо в умовах складних сцен.

Таким чином, розпізнавання жестів є перспективною технологією, яка знаходить застосування в робототехніці, інтерфейсах людина-комп’ютер та інтерактивних системах. Подальший розвиток цієї галузі спрямований на підвищення точності, стабільності та доступності, що сприятиме її ширшому впровадженню в повсякденне життя.

РОЗДІЛ 2. АНАЛІЗ ТЕХНОЛОГІЙ ІНТЕГРАЦІЇ ФУНКЦІЙ РОЗПІЗНАВАННЯ ЖЕСТІВ У ВЕБ-ДОДАТКИ

2.1 Огляд технологій інтеграції функцій розпізнавання жестів

Технології, які інтегрують функції розпізнавання жестів у веб-додатки, повинні відповідати низці вимог. Однією з ключових є продуктивність у реальному часі, оскільки такі системи зазвичай аналізують відеопотік у режимі онлайн. Це передбачає мінімальну затримку обробки та здатність працювати навіть на пристроях із обмеженими ресурсами. Важливою також є сумісність із сучасними веб-технологіями. Більшість веб-додатків розробляються на основі JavaScript або TypeScript, тому бібліотеки та фреймворки повинні бути адаптовані до цих мов програмування.

Якість розпізнавання жестів є ще однією критичною вимогою. Системи повинні забезпечувати високу точність розпізнавання, адаптуватися до різних умов, включаючи змінне освітлення, варіативність жестів, шум фону та відстань до камери.

Простота інтеграції також відіграє значну роль. Технологія має надавати API, приклади використання та детальну документацію, щоб розробники могли швидко впровадити функціонал у свої додатки. При цьому вона має бути гнучкою, щоб дозволяти налаштовувати моделі відповідно до специфічних потреб.

Обчислювальна ефективність є ще однією важливою вимогою. Виконання обробки жестів на стороні клієнта допомагає зменшити навантаження на сервери, а також забезпечити швидкість і зручність використання для кінцевих користувачів.

TensorFlow.js

TensorFlow.js - це потужна бібліотека для створення, навчання та виконання нейронних мереж безпосередньо в браузері або на сервері через Node.js. Вона забезпечує гнучкість у використанні моделей глибокого навчання, дозволяючи як створювати власні, так і використовувати попередньо навчені моделі. Основною перевагою TensorFlow.js є її здатність виконувати обчислення з використанням WebGL, що забезпечує значне прискорення на клієнтських

пристроях. Ця бібліотека дозволяє обробляти відеопотік у реальному часі, що робить її придатною для інтеграції функцій розпізнавання жестів.

Однак через високі обчислювальні вимоги, пов’язані з використанням глибоких нейронних мереж, TensorFlow.js вимагає достатніх апаратних ресурсів для досягнення максимальної продуктивності, особливо при роботі з великими відеопотоками в реальному часі.

Mediapipe

Mediapipe - це бібліотека, створена Google, яка пропонує набір готових рішень для задач комп’ютерного зору, зокрема для розпізнавання рук, обличчя та жестів. Вона виділяється своєю здатністю працювати в реальному часі навіть на пристроях із обмеженими ресурсами. Головною особливістю Mediapipe є її готові модулі, такі як Hands, які забезпечують точне визначення положення руки та жестів із використанням попередньо навчених моделей. Бібліотека інтегрується у веб-додатки через JavaScript API, що робить її зручною для використання у фронтенд-розробці. Завдяки підтримці WebAssembly Mediapipe забезпечує високу продуктивність і низьку затримку навіть у браузері.

Крім того, Mediapipe пропонує детальну документацію та приклади коду, що значно полегшує процес інтеграції. Її перевагою є мінімальні вимоги до налаштувань і висока якість розпізнавання жестів, що робить її ідеальним вибором для інтерактивних веб-додатків.

Handtrack.js

Handtrack.js — це легка JavaScript-бібліотека, орієнтована на базове розпізнавання жестів і трекінг рук у реальному часі. Вона побудована на основі TensorFlow.js і забезпечує просту інтеграцію в додатки, які потребують базового функціоналу. Ця бібліотека є зручною для швидкого прототипування, оскільки не вимагає глибоких знань у галузі машинного навчання. Вона підтримує роботу з відеопотоком у браузері, однак її точність та можливості адаптації обмежені порівняно з Mediapipe або TensorFlow.js. Handtrack.js підходить для невеликих проектів або навчальних цілей, але вимагає більше обчислювальних ресурсів для складніших задач розпізнавання жестів. Її обмеженість компенсується легкістю використання та доступністю.

2.2 Порівняльний аналіз доступних технологій

У сучасних веб-додатках інтеграція функцій розпізнавання жестів є важливою складовою для створення зручних і природних інтерфейсів взаємодії. Розробники мають доступ до різних технологій, таких як TensorFlow.js, Mediapipe та Handtrack.js, кожна з яких має свої переваги, обмеження та сфери застосування. У цьому аналізі розглядаються ключові аспекти цих технологій, включаючи якість розпізнавання, продуктивність, простоту інтеграції, обчислювальні вимоги та гнучкість, щоб визначити найкращий вибір для інтеграції у веб-додатки.

Якість розпізнавання

TensorFlow.js: Забезпечує високу якість розпізнавання, якщо моделі належним чином навчені та оптимізовані. Однак точність залежить від конкретної моделі, оскільки розробники можуть створювати як базові, так і складніші архітектури.

Mediapipe: Пропонує попередньо навчені моделі з високою точністю. Особливо добре працює в задачах трекінгу рук завдяки оптимізації для реального використання. Забезпечує стабільну роботу навіть за складних умов, таких як змінне освітлення чи складний фон.

Handtrack.js: Підходить для простих задач. Точність нижча, ніж у Mediapipe або TensorFlow.js, через обмеження в архітектурі та моделі.

Продуктивність у реальному часі

TensorFlow.js: Забезпечує гарну продуктивність завдяки використанню WebGL. Проте для складних моделей або великих відеопотоків потрібні потужні пристрої з дискретною графікою.

Mediapipe: Оптимізована для роботи в реальному часі на різних платформах, включаючи мобільні пристрої. Використання WebAssembly забезпечує низьку затримку та плавну обробку навіть за низької обчислювальної потужності.

Handtrack.js: Продуктивність задовільна для невеликих задач, але під час обробки складних сценаріїв може виникати затримка.

Гнучкість

TensorFlow.js: Найбільш гнучка бібліотека, яка дозволяє створювати

кастомні моделі для розпізнавання жестів, а також використовувати попередньо навчені рішення. Ідеально підходить для складних і специфічних задач.

Mediapipe: Менш гнучка, оскільки орієнтована на використання готових моделей. Підходить для проєктів, де потрібне швидке впровадження без значної кастомізації.

Handtrack.js: Обмежена у гнучкості, оскільки пропонує лише базові можливості трекінгу рук без можливості налаштування моделей.

Категорія	**TensorFlow.js**	**Mediapipe**	**Handtrack.js**
Якість розпізнавання	Висока, залежить від моделі	Висока, стабільна за різних умов	Задовільна для простих задач
Продуктивність	Висока, залежить від пристрою	Висока, оптимізована для реального часу	Задовільна, але з обмеженнями
Обчислювальні вимоги	Високі	Низькі	Низькі
Гнучкість	Максимальна	Середня	Обмежена

Таблиця 2.1 Порівняння технологій розпізнавання жестів

Вибір технології залежить від вимог конкретного проєкту. TensorFlow.js підходить для складних і кастомних рішень, але вимагає більше ресурсів і зусиль для інтеграції. Mediapipe забезпечує ідеальний баланс між точністю, продуктивністю та простотою використання, роблячи її найкращим вибором для більшості веб-додатків. Handtrack.js варто використовувати для невеликих проєктів або навчальних цілей, де потрібна проста інтеграція та базовий функціонал.

2.3 Архітектура веб-додатку

Архітектура веб-додатку складається з трьох основних компонентів: клієнтської частини, серверної частини та бази даних. Кожен із цих елементів виконує чітко визначену роль у забезпеченні функціональності системи, що дозволяє досягти її масштабованості, ефективності та зручності у використанні.

Клієнтська частина, або фронтенд, відповідає за взаємодію користувача із системою. Сучасні веб-додатки зазвичай реалізують клієнт за модульним підходом, що означає розділення функціональності на окремі компоненти.

Наприклад, різні модулі можуть обробляти такі завдання, як авторизація, відображення списків, робота з жестами або навігація між елементами. Така модульність спрощує розробку та підтримку коду, дозволяючи оновлювати або замінювати окремі частини без впливу на весь додаток. Для побудови клієнтської частини використовуються мови HTML, CSS, JavaScript або фреймворки, такі як Angular, React чи Vue.js, що забезпечують динамічну і гнучку взаємодію користувача з додатком.

Серверна частина, або бекенд, реалізує бізнес-логіку, забезпечує обробку запитів від клієнта та взаємодіє з базою даних. Архітектура сервера зазвичай базується на трирівневому підході (3-tier), який передбачає поділ на рівні презентації, бізнес-логіки та даних. Рівень презентації отримує запити від клієнтської частини, обробляє їх і формує відповіді. Рівень бізнес-логіки виконує всі основні операції, пов’язані з функціональністю системи, такі як верифікація користувача, обробка жестів або логіка роботи з презентаціями. Рівень даних забезпечує доступ до бази даних, яка є джерелом збереження та обробки інформації. Така структура підвищує гнучкість і безпеку додатка, оскільки дозволяє масштабувати кожен рівень незалежно.

База даних є центральним елементом зберігання даних, що забезпечує швидкий доступ до інформації. Для різних типів даних можуть використовуватися реляційні або нереляційні бази. Наприклад, реляційні бази даних, такі як MySQL чи PostgreSQL, застосовуються для структурованої інформації, яка має чіткі зв’язки між даними, тоді як нереляційні бази, наприклад MongoDB чи Firebase, підходять для роботи з більш гнучкими даними, такими як історії розпізнавання жестів чи налаштування користувача. База даних часто оптимізується за допомогою індексації або кешування (наприклад, Redis), щоб забезпечити ефективну обробку запитів.

У такій архітектурі клієнтська частина надсилає запити до сервера через API, наприклад REST. Сервер обробляє ці запити на рівні бізнес-логіки та, за потреби, звертається до бази даних. Після виконання запиту сервер повертає клієнту відповідь із результатами. Такий підхід забезпечує чіткий поділ обов’язків між компонентами, дозволяючи системі залишатися масштабованою, надійною та ефективною.

2.4 Висновки розділу

У межах другого розділу проведено аналіз технологій, які забезпечують інтеграцію функцій розпізнавання жестів у веб-додатки, а також розглянуто архітектурні аспекти створення таких систем. Було виконано огляд основних бібліотек і фреймворків, зокрема TensorFlow.js, Mediapipe і Handtrack.js, з метою визначення їх можливостей та особливостей. Результати порівняльного аналізу показали, що Mediapipe є оптимальним вибором для інтеграції функцій розпізнавання жестів у веб-додатки. Завдяки готовим модулям, високій точності та продуктивності в реальному часі вона є зручним і ефективним рішенням для більшості проєктів. TensorFlow.js продемонструвала високу гнучкість і придатність для створення кастомних моделей, однак вимагає більше апаратних ресурсів і знань у галузі машинного навчання. Handtrack.js, у свою чергу, є легкою у використанні бібліотекою для простих задач, але має обмежену функціональність та точність.

Також розглянуто архітектуру веб-додатку, яка забезпечує інтеграцію таких функцій. Було акцентовано увагу на модульній клієнтській частині, трирівневій серверній архітектурі та ролі бази даних для забезпечення зберігання та обробки даних. Такий підхід дозволяє досягти гнучкості, масштабованості та ефективності у реалізації систем із підтримкою розпізнавання жестів. Таким чином, у розділі закладено основу для вибору та впровадження технологій, які найкраще відповідають потребам веб-додатків, орієнтованих на інтерактивну взаємодію за допомогою жестів.

РОЗДІЛ 3. РОЗРОБКА АЛГОРИТМУ РОЗПІЗНАВАННЯ ЖЕСТІВ У РЕАЛЬНОМУ ЧАСІ

Алгоритм розпізнавання жестів у реальному часі побудований на інтеграції сучасних технологій роботи з відеопотоком, виділенні ключових точок рук та їх аналізі для ідентифікації жестів. Процес розробки починається з ініціалізації середовища веб-додатку, налаштування моделей розпізнавання та побудови циклічного оброблення відеоданих.

3.1 Опис роботи алгоритму

1. Ініціалізація середовища

Першим етапом є створення основного середовища для роботи алгоритму. На цьому кроці налаштовуються DOM-елементи, такі як відео, канвас (canvas) для відображення результатів роботи алгоритму.

DOM-елементи також використовуються для доступу до камери пристрою. Алгоритм запитує дозвіл на використання відеопотоку, після чого ініціалізується доступ до камери.

2. Підготовка необхідних моделей розпізнавання

Підготовка моделей розпізнавання включає вибір, завантаження та налаштування попередньо навчених моделей, таких як Mediapipe Hands. Ця модель здатна визначати до 21 ключової точки руки, включаючи кінчики пальців і суглоби, що є основою для розпізнавання жестів.

На етапі завантаження модель інтегрується через офіційний JavaScript API, який використовує WebAssembly для забезпечення продуктивності.

3. Запуск циклу розпізнавання в реальному часі

Цикл розпізнавання жестів у реальному часі забезпечує постійну обробку відеопотоку для виявлення орієнтирів рук, визначення жестів та їх відображення. Робота алгоритму складається з кількох ключових етапів:

- Обробка відеопотоку. На початку кожної ітерації обробляється поточний кадр із відеопотоку, отриманого з камери. Алгоритм передає цей кадр у модель розпізнавання, яка аналізує положення рук у кадрі. Важливим є точний аналіз просторового розташування ключових точок на руках, таких як кінчики пальців або суглоби.
- Очищення простору для візуалізації. Перед відображенням нових даних

полотно, яке використовується для візуалізації результатів, очищується. Це забезпечує уникнення накладання попередніх результатів на поточні, що підвищує чіткість і точність візуалізації.

- Візуалізація ключових точок та з’єднань Якщо модель успішно визначила ключові точки рук, вони відображаються на полотні у вигляді маркерів і ліній, які з’єднують ці точки. Відображення виконується у реальному часі, щоб надати користувачу можливість візуально оцінити роботу алгоритму.
- Збереження результатів аналізу. Отримані координати ключових точок зберігаються у внутрішньому сховищі додатку. Це дає можливість іншим компонентам системи використовувати ці дані для подальшого аналізу чи виконання певних команд.
- Аналіз та розпізнавання жестів. На основі зібраних координат алгоритм визначає, чи відповідають поточні положення рук одному з відомих жестів. Якщо жест визначено, його ідентифікатор зберігається для подальшої обробки, наприклад, для виконання інтерактивних дій або управління елементами додатку.

Для уникнення багаторазової реакції на один і той самий жест алгоритм вводить паузи між розпізнаваннями. Це дозволяє системі стабільно обробляти нові жести та уникати дублювання результатів.

Повтор ітерації. Після завершення всіх операцій алгоритм запускає новий цикл обробки, використовуючи механізм безперервного оновлення. Це забезпечує постійний аналіз відеопотоку та дозволяє системі працювати у режимі реального часу.

```
if (this.isRecognitionActive) {
  // Analyze the current frame from the video feed
  const recognitionResults = await this.handRecognitionService.recognizer.recognizeForVideo(
    this.cameraFeedElement.nativeElement,
    Date.now()
  );

  // Access the canvas context for rendering
  const canvasContext = this.outputCanvasElement.nativeElement.getContext('2d');

  // Clear previous drawings from the canvas
  canvasContext.clearRect(
    x: 0,
    y: 0,
    this.outputCanvasElement.nativeElement.width,
    this.outputCanvasElement.nativeElement.height
  );

  // Draw the hand landmarks and connections if available
  if (recognitionResults.landmarks) {
    recognitionResults.landmarks.forEach((landmarkSet : NormalizedLandmark[] ) => {
      drawConnectors(canvasContext, landmarkSet, HAND_CONNECTIONS, style: {
        color: '#01da0c',
        lineWidth: 1,
      });
      drawLandmarks(canvasContext, landmarkSet, style: {
        color: '#1d5001',
        radius: 1,
      });
    });

    // Store the landmark data in a shared service
    this.sharedStateService.storeLandmarks(recognitionResults.landmarks);
  }

  // Process gesture recognition results if any are detected
  if (recognitionResults.gestures.length > 0) {
    this.processGestureResult(recognitionResults);
  }

  // Manage the cooldown between gesture recognitions
  this.manageCooldown();
```

Рис. 3.1 Лістинг коду ітерації розробленого алгоритму

3.2 Висновки розділу

Такий підхід забезпечує безперервну роботу алгоритму, що дозволяє виконувати розпізнавання жестів із мінімальною затримкою та високою точністю, підтримуючи інтерактивність і зручність використання.

РОЗДІЛ 4. РОЗРОБКА ВЕБ-СИСТЕМИ ПЕРЕГЛЯДУ ПРЕЗЕНТАЦІЙ

1.1 Аналіз використаних технологій

Для розробки веб-системи було використано сучасні технології, які забезпечують ефективність, масштабованість та зручність у використанні. Система побудована на основі модульного клієнтського фреймворку Angular, серверної платформи Java Spring та реляційної бази даних MySQL. Такий стек технологій дозволив створити інтерактивний, продуктивний і стійкий до навантажень веб-додаток.

Фронтенд

Angular - це популярний фронтенд-фреймворк, який використовується для створення динамічних та модульних користувацьких інтерфейсів. У веб-системі Angular відповідає за реалізацію клієнтської частини, забезпечуючи:

- Модульність: розробка системи була виконана з поділом на незалежні компоненти, такі як модулі для управління жестами, відображення слайдів та навігації.
- Динамічність: використання двостороннього зв’язку даних (data binding) забезпечує миттєве оновлення інтерфейсу відповідно до змін на сервері.
- Інтерактивність: через Angular було реалізовано логіку відстеження жестів, обробку користувацьких дій і візуалізацію результатів, таких як відображення презентації чи перемикання слайдів.

Сумісність із REST API: Angular дозволяє легко інтегрувати серверну частину, забезпечуючи швидкий обмін даними між клієнтом і сервером.

Для централізованого управління станом додатку використовується **NgRx store**, зберігаючи дані про аутентифікацію користувача, інформація про жести та інші параметри в єдиному сховищі. Це забезпечує прозорість і легкість доступу до потрібної інформації, оскільки всі компоненти додатку можуть взаємодіяти з даними через спеціальні селектори, уникаючи дублювання запитів чи станів. NgRx впроваджує передбачувану модель змін стану, де всі оновлення відбуваються через ред’юсери, які визначають чіткі правила обробки дій. Це спрощує тестування та налагодження, роблячи додаток стабільним навіть за складної логіки. Крім того, NgRx дозволяє легко синхронізувати дані із

сервером завдяки ефектам, які обробляють асинхронні операції, такі як завантаження презентацій чи збереження змін.

Бекенд

Spring - це потужний серверний фреймворк на базі Java, який відповідає за реалізацію бекенд-частини системи. У веб-додатку Spring використовується для:

- Реалізації REST API: серверна частина забезпечує обробку запитів від клієнта.
- Управління бізнес-логікою: Spring дозволяє структурувати код таким чином, щоб основна функціональність (наприклад, логіка обробки жестів) була відокремлена від низькорівневих деталей, таких як з'єднання з базою даних.
- Безпека: через Spring Security реалізовано механізми аутентифікації та авторизації, які захищають систему від несанкціонованого доступу.
- Масштабованість: Spring забезпечує високу продуктивність і можливість обробки великої кількості одночасних запитів, що є важливим для інтерактивних веб-додатків.

Субд

MySQL - це реляційна база даних, яка використовується для зберігання даних. Завдяки зв'язкам між таблицями можна ефективно обробляти дані, наприклад, знаходити презентації, які належать конкретному користувачу.

Для забезпечення **безпеки** у веб-системі використовується механізм аутентифікації та авторизації на основі JWT (JSON Web Token). Цей підхід дозволяє надійно передавати дані між клієнтом і сервером, забезпечуючи контроль доступу до ресурсів та захист інформації від несанкціонованого доступу. JWT представляє собою компактний токен у форматі JSON, який включає три основні частини: заголовок, корисне навантаження та підпис. Заголовок містить інформацію про тип токена і метод шифрування, корисне навантаження включає ідентифікаційні дані користувача, його роль і термін дії токена, а підпис створюється за допомогою секретного ключа. Усі ці компоненти забезпечують цілісність і захист даних, які передаються між клієнтом і сервером.

Процес аутентифікації розпочинається із запиту користувача на вхід до системи. Користувач надає свої облікові дані, які сервер перевіряє, звіряючи з записами в базі даних. У разі успішної перевірки сервер генерує JWT, який містить інформацію про користувача та передає його клієнту. Токен зберігається у клієнтському сховищі, звідки він додається до кожного запиту клієнта для аутентифікації на сервері. При отриманні запиту сервер розшифровує токен і перевіряє його валідність за допомогою секретного ключа. Якщо токен дійсний, сервер надає доступ до відповідних ресурсів, ідентифікуючи користувача та його права доступу. У випадках, коли час дії токена минув, користувачеві необхідно повторно виконати вхід для отримання нового токена.

1.2 Огляд основних сторінок

Веб-система реалізує зручний та інтуїтивно зрозумілий інтерфейс, який забезпечує доступ до основних функцій через кілька ключових сторінок. Кожна сторінка виконує свою роль у забезпеченні функціональності системи.

Сторінки авторизації

Сторінки логіну та реєстрації забезпечують безпечний доступ до системи. Користувач вводить свої облікові дані, які перевіряються серверною частиною. У разі успішної автентифікації надається доступ до особистого кабінету.

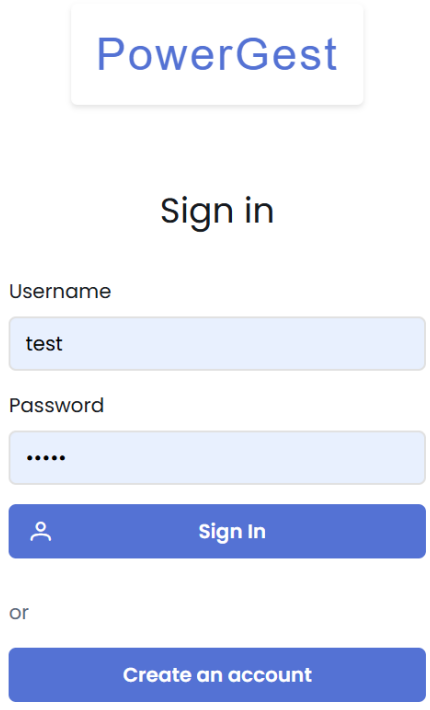

Рис. 4.1.1 Сторінка логіну

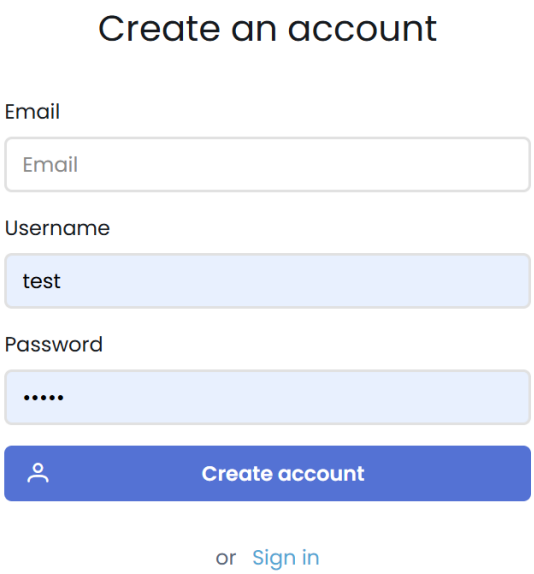

Рис. 4.1.2 Сторінка створення нового аккаунту

Головна сторінка

Головна сторінка веб-системи служить візитною карткою проекту, яка представляє його основну ідею, функціонал та особливості. Вона містить кілька ключових блоків, які інформують користувача про мету проекту, його особливості, застосовані технології та реалізовану методологію.

У нижній частині сторінки розміщено заклик до взаємодії, де пропонується зв’язатися з розробником через LinkedIn. Цей розділ оформлений у вигляді кнопки з посиланням, що робить взаємодію зручнішою для користувача.

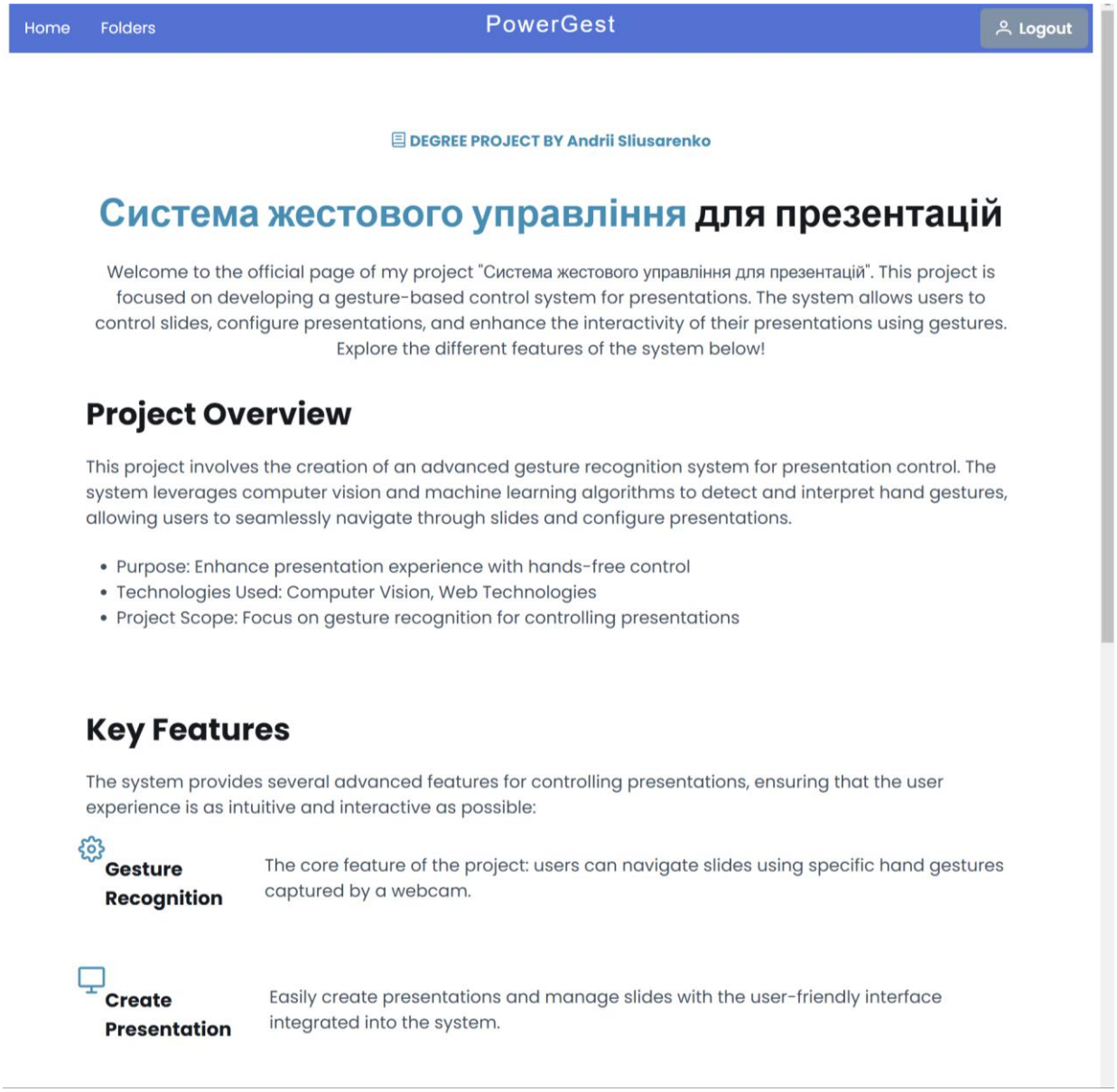

Рис. 4.2 Головна сторінка

Сторінка форматування презентацій

Сторінка забезпечує повноцінне інтерактивне середовище для створення та попереднього перегляду наявих слайдів, а також розподіл презентацій за категоріями (folder).

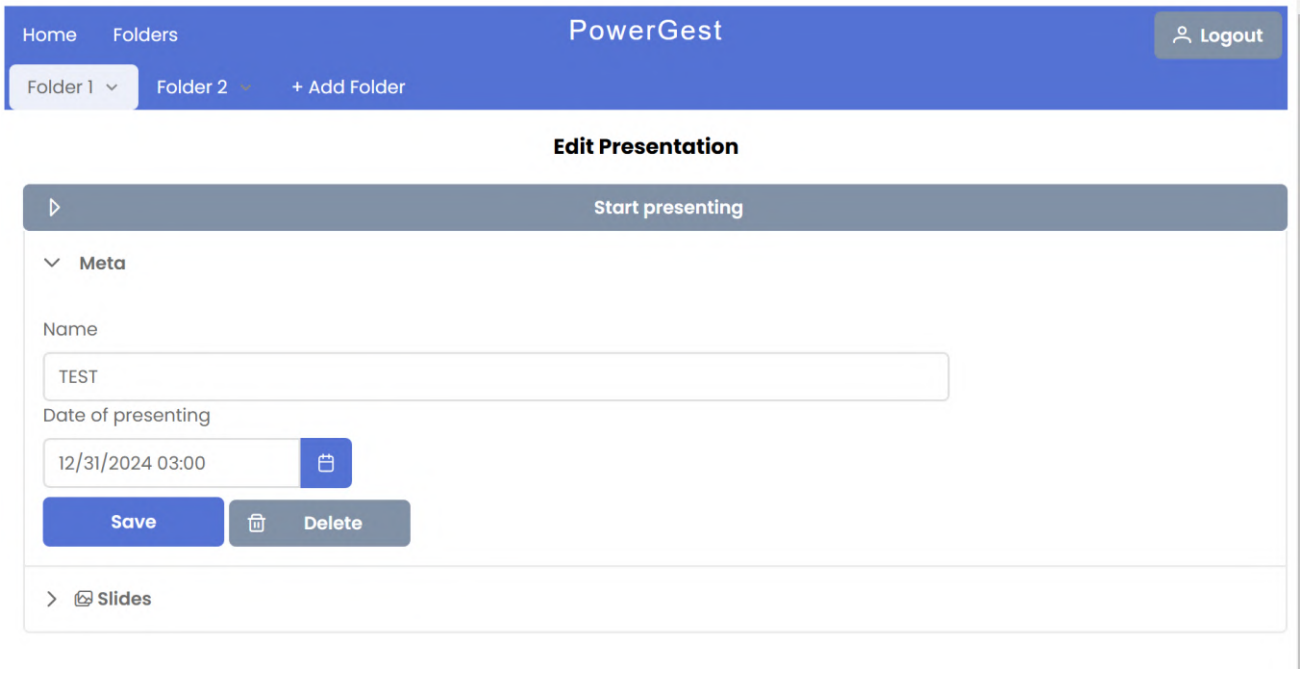

Рис. 4.3.1 Сторінка форматування презентацій

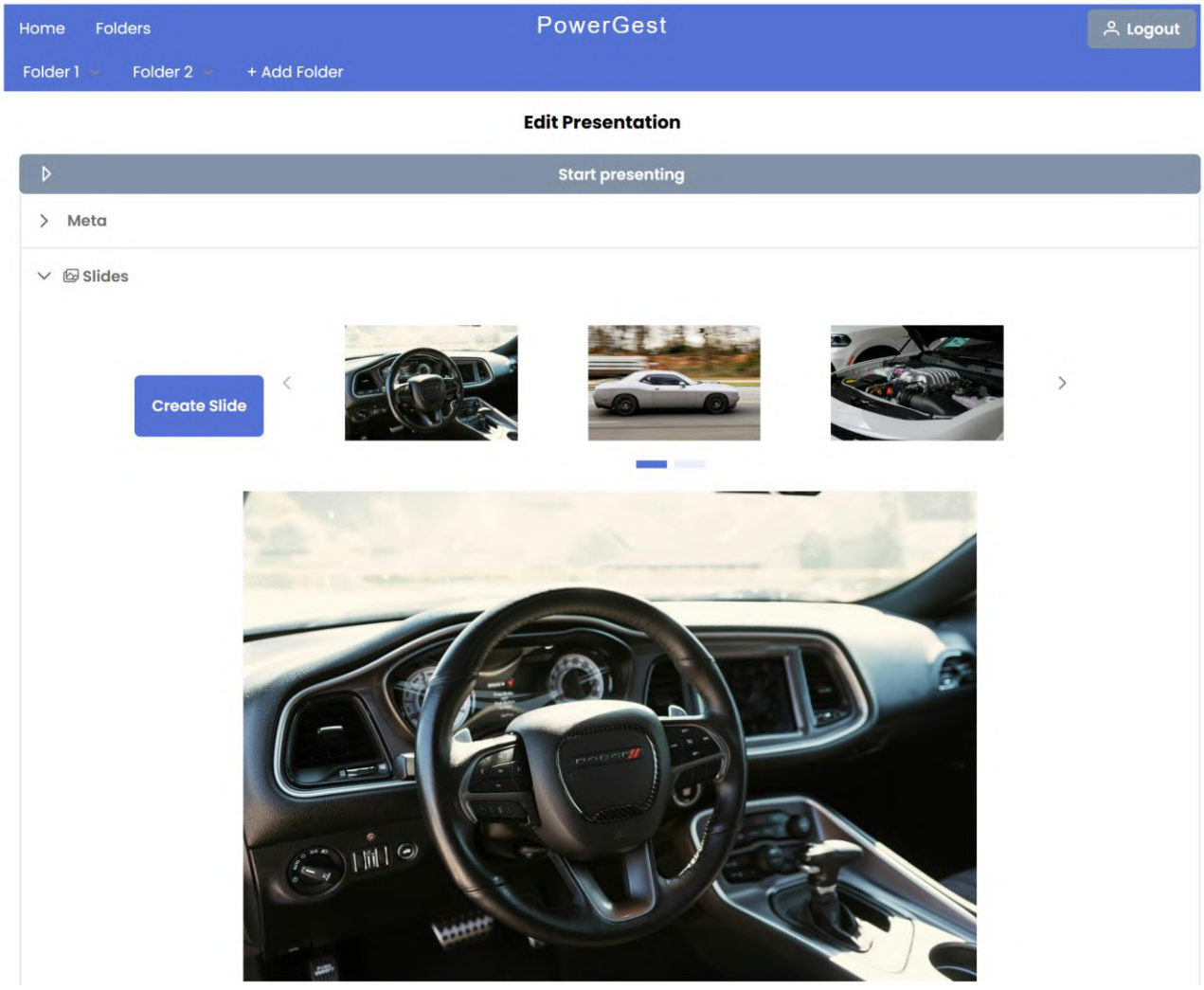

Рис. 4.3.2 Сторінка форматування презентацій

Сторінка демонстрації

Сторінка демонстрації веб-системи є центральним елементом, що дозволяє користувачам переглядати презентацію в інтерактивному режимі та

управляти слайдами використовуючи жести.

Функціонал жестового управління містить:

- Перемикання слайдів: Виконуючи жест розкритої долоні, користувач може переходити до наступного, а за допомогою закритої - попереднього слайду. Також є номер активного слайду.
- Живий зворотний зв'язок: Користувач бачить, як система реагує на його жести в реальному часі. Це досягається за рахунок візуалізації ключових точок руки.

1

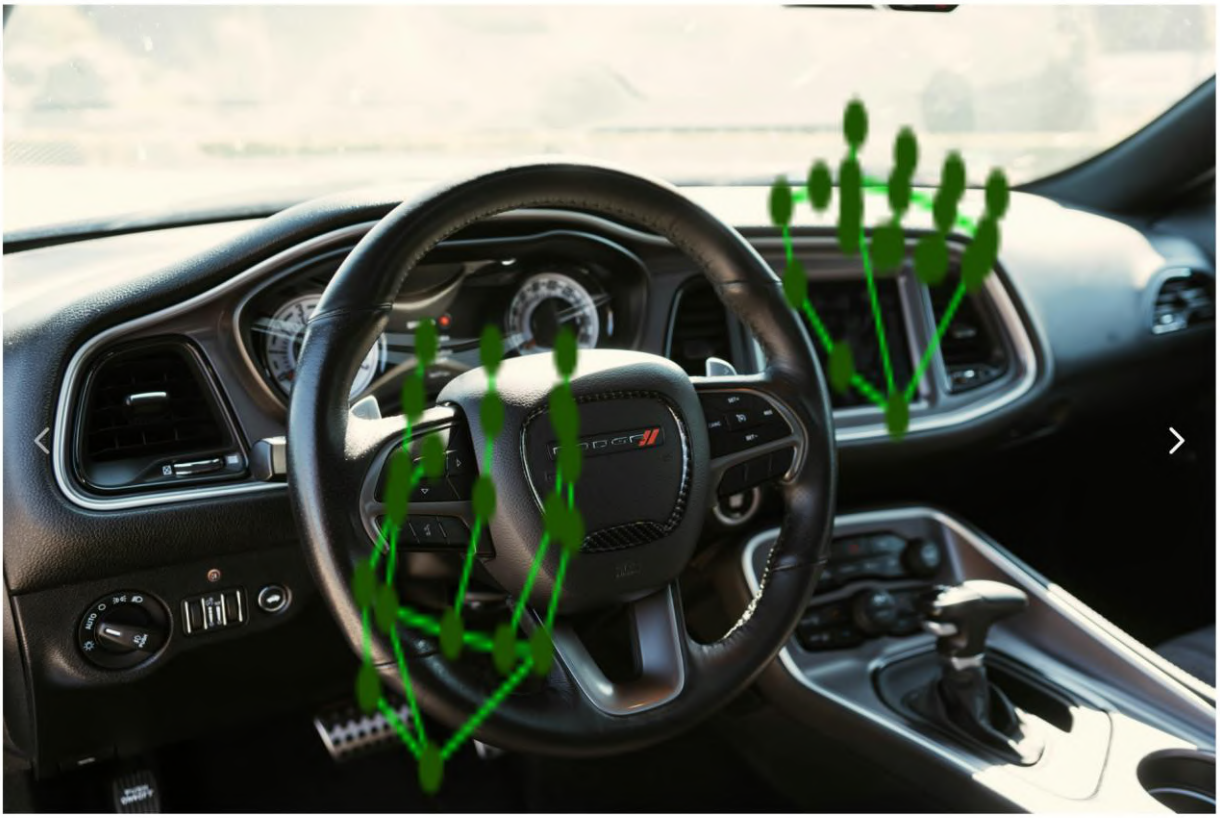

Рис. 4.4.1 Сторінка демострування із видимими 2 руками

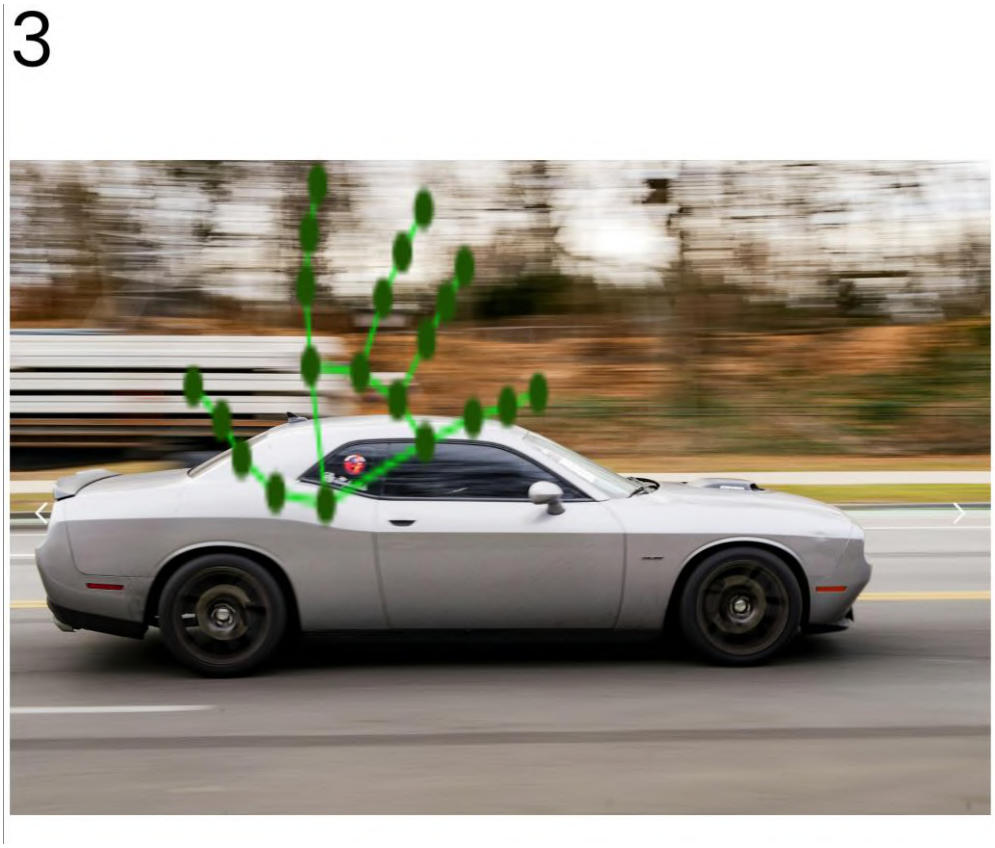

Рис. 4.4.2 Сторінка демострування із жестом відкритої долоні (перемикання на наступний слайд)

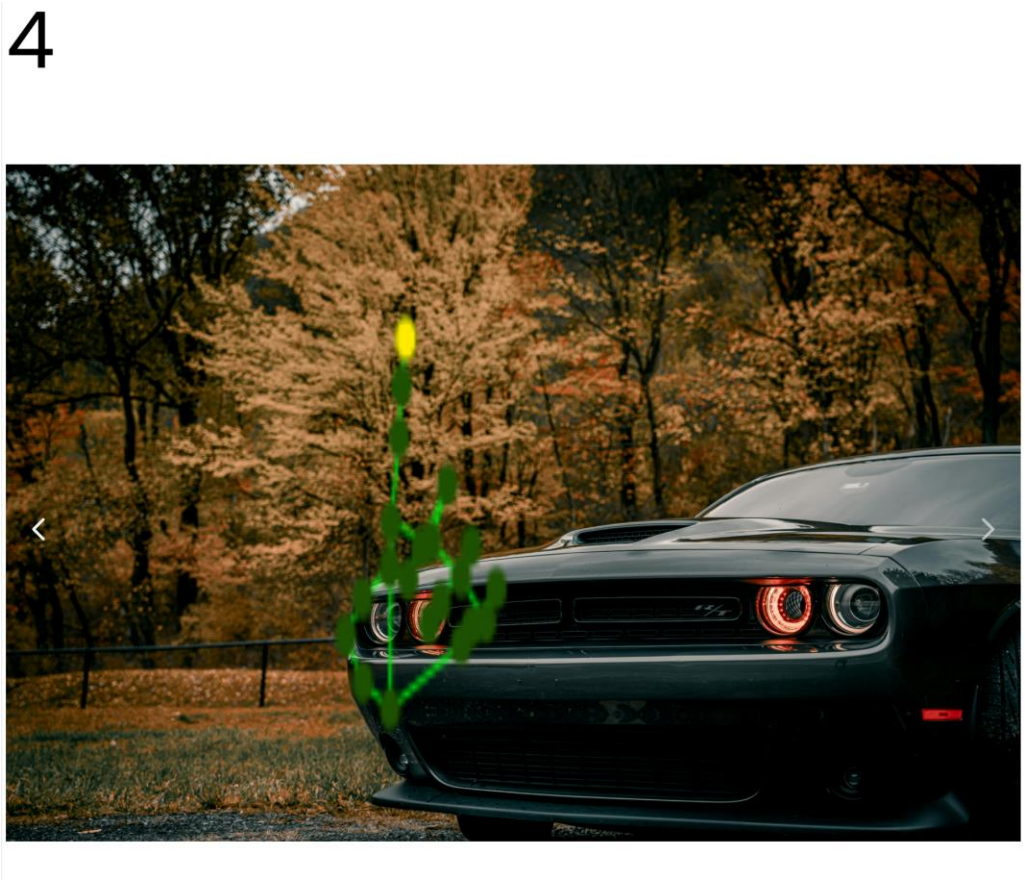

Рис. 4.4.3 Сторінка демострування із жестом, що виділяє точку на пальці

4.3 Висновки розділу

У четвертому розділі було проаналізовано використані технології та структуру веб-системи для перегляду презентацій. Розробка системи базувалася на сучасному технологічному стеку, включаючи Angular, Java Spring та MySQL, що забезпечило створення продуктивного, гнучкого та масштабованого рішення. Аналіз продемонстрував, як ці технології взаємодіють між собою для забезпечення інтерактивного користувацького досвіду. Також проведено огляд основних сторінок системи, кожна з яких виконує ключову роль у забезпеченні функціональності. Таким чином, цей розділ продемонстрував ефективність інтеграції сучасних технологій у реалізацію. Результатом є інтуїтивна та продуктивна платформа, що відповідає сучасним вимогам до веб-додатків.

ВИСНОВКИ

У ході роботи було досліджено, розроблено та впроваджено систему жестового управління для презентацій, що поєднує сучасні методи розпізнавання, інтеграцію у веб-додаток і створення зручного інтерфейсу.

У першому розділі розглянуто основні підходи до розпізнавання жестів, включаючи методи комп'ютерного зору та глибокого навчання. Особливу увагу приділено згортковим нейронним мережам, які дозволяють точно та швидко розпізнавати жести. Другий розділ проаналізував технології інтеграції функцій розпізнавання у веб-додатки. Було представлено архітектуру додатку з модульним клієнтом, трирівневим сервером і базою даних, що забезпечує гнучкість і масштабованість системи. У третьому розділі розроблено алгоритм для реального часу, що забезпечує обробку відеопотоку, визначення ключових точок і розпізнавання жестів. Оптимізація алгоритму дозволила досягти високої продуктивності навіть на пристроях із обмеженими ресурсами. У четвертому розділі реалізовано веб-систему для перегляду презентацій, включаючи функції створення, редагування та інтерактивного перегляду слайдів. Використання Angular, Java Spring та MySQL забезпечило стабільність і продуктивність системи.

Результати роботи демонструють ефективність поєднання методів комп'ютерного зору, глибокого навчання та веб-технологій. Система відкриває нові можливості у демострування презентацій та може бути використана для подальших досліджень і впроваджень у суміжних сферах.

ПЕРЕЛІК ПОСИЛАНЬ

1. Deep Learning [Електронний ресурс] // MIT Press. – 2016. – Режим доступу до ресурсу: https://www.deeplearningbook.org.
2. Mediapipe: Cross-Platform Machine Learning Solutions [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: https://google.github.io/mediapipe/.
3. Angular: Developer Documentation [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: https://angular.io/.
4. Spring Framework Documentation [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: https://spring.io/.
5. MySQL Reference Manual [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: https://dev.mysql.com/doc/.
6. TensorFlow: A System for Large-Scale Machine Learning [Електронний ресурс] // USENIX Symposium on Operating Systems Design and Implementation. – 2016. – Режим доступу до ресурсу: https://www.tensorflow.org.
7. NgRx: State Management for Angular [Електронний ресурс]. – 2024. – Режим доступу до ресурсу: https://ngrx.io/.
8. JSON Web Tokens Introduction [Електронний ресурс] // JWT.IO. – 2024. – Режим доступу до ресурсу: https://jwt.io/.
9. Deep Learning [Електронний ресурс] // MIT Press. – 2024. – Режим доступу до ресурсу: https://www.deeplearningbook.org.
10. An Introduction to Convolutional Neural Networks [Електронний ресурс] // Stanford University. – 2024. – Режим доступу до ресурсу: https://cs231n.github.io/convolutional-networks/.
11. Chollet, F. Deep Learning with Python [Електронний ресурс]. – Manning Publications. – 2024. – Режим доступу до ресурсу: https://www.manning.com/books/deep-learning-with-python.

ДОДАТКИ

Додаток 1

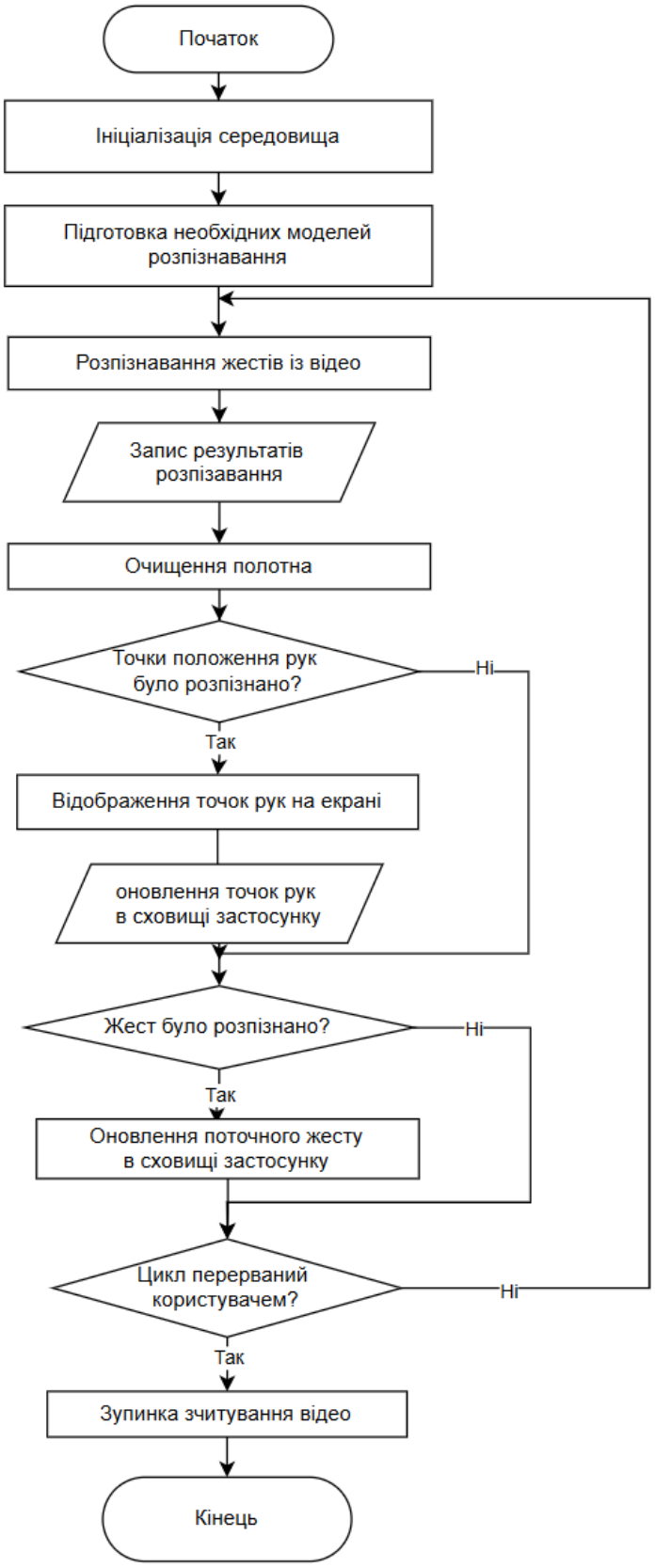

Рисунок Д.1 Блок схема алгоритму

Лістинг коду Д.2 Компонент розпізнавання жестів

```
export class HandGestureDetectionComponent implements AfterViewInit, OnDestroy {
  private cooldownDuration = 20; // Frames to wait before recognizing the next gesture
  private cooldownCounter = 0; // Tracks cooldown progress

  @ViewChild('cameraFeed') cameraFeedElement: ElementRef;
  @ViewChild('outputCanvas') outputCanvasElement: ElementRef;

  private gestureDetected = false;
  private isRecognitionActive = false;
  private videoStream: MediaStream;

  constructor(
    private handRecognitionService: HandGestureDetectionService,
    private sharedStateService: SharedStoreService
  ) {}

  /**
   * Starts the gesture recognition process by activating the recognition loop.
   */
  async launchDetection() {
    this.isRecognitionActive = true;
    this.executeRecognitionLoop();
  }

  /**
   * Lifecycle hook that is triggered after the component's view has been fully initialized.
   * Initializes the gesture recognition system and attempts to access the user's webcam.
   */
  async ngAfterViewInit() {
      // Initialize the gesture recognition system
      await this.handRecognitionService.setupRecognizer();

      // Verify if the device has access to media devices (camera)
      if (navigator.mediaDevices?.getUserMedia) {
        const videoOptions = { video: true };

        // Start the camera stream and bind it to the video element
        this.videoStream = await navigator.mediaDevices.getUserMedia(videoOptions);
        this.cameraFeedElement.nativeElement.srcObject = this.videoStream;
```

```
    } else {
      throw new Error('Camera access is not supported on this device.');
    }
  }

  /**
   * Handles gesture recognition results, ensuring that gestures are processed
   * only if the cooldown period has elapsed.
   */
  private processGestureResult(results: GestureRecognizerResult): void {
    const detectedGesture = <Gesture>results.gestures[0][0].categoryName;

    if (!this.gestureDetected) {
      // Update the recognized gesture in the shared state service
      this.sharedStateService.updateHandPosition(detectedGesture);
      this.gestureDetected = true;
    }
  }

  /**
   * Manages the cooldown state to prevent rapid-fire gesture detections.
   * Resets the gesture recognition state once the cooldown period ends.
   */
  private manageCooldown() {
    if (this.gestureDetected) {
      this.cooldownCounter++;

      // Reset the recognized gesture to "None" during cooldown
      this.sharedStateService.updateHandPosition(Gesture.None);

      // Check if the cooldown period has been reached
      if (this.cooldownCounter >= this.cooldownDuration) {
        this.gestureDetected = false;
        this.cooldownCounter = 0;
      }
    }
  }

  /**
   * Executes the recognition loop that continuously processes video frames
   * for gesture recognition.
   */
  private async executeRecognitionLoop(): Promise<void> {
    if (this.isRecognitionActive) {
      // Analyze the current frame from the video feed
      const recognitionResults = await
this.handRecognitionService.recognizer.recognizeForVideo(
        this.cameraFeedElement.nativeElement,
```

```
      Date.now()
    );

    // Access the canvas context for rendering
    const canvasContext =
this.outputCanvasElement.nativeElement.getContext('2d');

    // Clear previous drawings from the canvas
    canvasContext.clearRect(
      0,
      0,
      this.outputCanvasElement.nativeElement.width,
      this.outputCanvasElement.nativeElement.height
    );

    // Draw the hand landmarks and connections if available
    if (recognitionResults.landmarks) {
      recognitionResults.landmarks.forEach((landmarkSet) => {
        drawConnectors(canvasContext, landmarkSet, HAND_CONNECTIONS, {
          color: '#01da0c',
          lineWidth: 1,
        });
        drawLandmarks(canvasContext, landmarkSet, {
          color: '#1d5001',
          radius: 1,
        });
      });

      // Store the landmark data in a shared service
      this.sharedStateService.storeLandmarks(recognitionResults.landmarks);
    }

    // Process gesture recognition results if any are detected
    if (recognitionResults.gestures.length > 0) {
      this.processGestureResult(recognitionResults);
    }

    // Manage the cooldown between gesture recognitions
    this.manageCooldown();

    // Request the next frame for gesture recognition
    requestAnimationFrame(() => this.executeRecognitionLoop());
   }
  }
}

<app-header></app-header>
```

```
<section class="surface-section px-4 py-8 md:px-6 lg:px-8">
  <div class="text-center text-700">

    <!-- Project Title and Introduction -->
    <div class="text-blue-600 font-bold mb-3">
      <i class="pi pi-book"></i> DEGREE PROJECT BY Andrii Sliusarenko
    </div>

    <h1 class="text-900 font-bold text-5xl mb-3">
      <span class="text-blue-600">Система жестового управління</span> для презентацій
    </h1>

    <p class="text-700 text-lg mb-5">
      Welcome to the official page of my project "Система жестового управління для презентацій". This project is focused on developing a gesture-based control system for presentations. The system allows users to control slides, configure presentations, and enhance the interactivity of their presentations using gestures. Explore the different features of the system below!
    </p>

    <div class="text-left mb-8">
      <h2 class="text-900 font-bold text-4xl mb-3">Project Overview</h2>
      <p class="text-700 text-lg mb-4">
        This project involves the creation of an advanced gesture recognition system for presentation control. The system leverages computer vision and machine learning algorithms to detect and interpret hand gestures, allowing users to seamlessly navigate through slides and configure presentations.
      </p>
      <ul class="list-disc pl-5 text-left text-lg text-700 mb-5">
        <li>Purpose: Enhance presentation experience with hands-free control</li>
        <li>Technologies Used: Computer Vision, Web Technologies</li>
        <li>Project Scope: Focus on gesture recognition for controlling presentations</li>
      </ul>
    </div>

    <div class="text-left mb-8">
      <h2 class="text-900 font-bold text-4xl mb-3">Key Features</h2>
      <p class="text-700 text-lg mb-4">
        The system provides several advanced features for controlling presentations, ensuring that the user experience is as intuitive and interactive as possible:
      </p>
      <div class="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6 text-left">
        <div class="flex flex-col items-start">
          <i class="pi pi-cog text-3xl text-blue-600 mb-3"></i>
          <h3 class="text-900 font-bold text-xl mb-2">Gesture Recognition</h3>
```

```
        <p class="text-700 text-lg">The core feature of the project: users can navigate slides using specific hand gestures captured by a webcam.</p>
      </div>
      <div class="flex flex-col items-start">
        <i class="pi pi-desktop text-3xl text-blue-600 mb-3"></i>
        <h3 class="text-900 font-bold text-xl mb-2">Create Presentation</h3>
        <p class="text-700 text-lg">Easily create presentations and manage slides with the user-friendly interface integrated into the system.</p>
      </div>
      <div class="flex flex-col items-start">
        <i class="pi pi-images text-3xl text-blue-600 mb-3"></i>
        <h3 class="text-900 font-bold text-xl mb-2">Configure Slides</h3>
        <p class="text-700 text-lg">Customize and configure each slide, adjusting content, for a seamless presentation experience.</p>
      </div>
    </div>
  </div>

  <div class="text-left mb-8">
    <h2 class="text-900 font-bold text-4xl mb-3">Methodology</h2>
    <p class="text-700 text-lg mb-5">
      The development process involved implementing a combination of gesture recognition algorithms and real-time control mechanisms. The steps taken during the project include:
    </p>
    <ul class="list-decimal pl-5 text-left text-lg text-700 mb-5">
      <li>Stage 1: Research and selection of gesture recognition algorithms</li>
      <li>Stage 2: Development of the gesture recognition module</li>
      <li>Stage 3: Integration with the presentation software for real-time control</li>
      <li>Final Stage: Testing and optimization of the system for different presentation scenarios</li>
    </ul>
  </div>

  <div class="text-center mb-8">
    <h2 class="text-900 font-bold text-4xl mb-3">Connect with Me on LinkedIn</h2>
    <p class="text-700 text-lg mb-5">
      Feel free to reach out to me on LinkedIn for more details about my project or to connect professionally. I'd be happy to discuss my work further!
    </p>

    <a href="https://www.linkedin.com/in/your-linkedin-profile"
      target="_blank"
```

```
        class="font-bold px-6 py-3 p-button-raised p-button-rounded text-white bg-blue-600 hover:bg-blue-700 transition duration-300 transform hover:scale-105 active:scale-95"
        aria-label="LinkedIn Profile"
      >
        <i class="pi pi-linkedin mr-2"></i> Visit My LinkedIn
      </a>
    </div>

  </div>
</section>
```