МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА ВЕБ ДОДАТКУ ДЛЯ ОПИТУВАННЯ КОРИСТУВАЧА ЗА ДОПОМОГОЮ РОЗПІЗНАВАННЯ ЖЕСТІВ»**

Виконав(ла): студент(ка) 2 курсу, групи 8.1210
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми інженерія програмного забезпечення
(назва освітньої програми)
А.В. Татаринова
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, доцент, к.ф.-м.н. Кудін О.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри фундаментальної та прикладної математики, доцент, к.ф.-м.н. Панасєнко Є.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти магістр

Спеціальність 121 інженерія програмного забезпечення

(шифр і назва)

Освітня програма інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.

(підпис)

« 09 » 06 2021 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ

Татариновій Аліні Володимирівні

(прізвище, ім'я та по-батькові)

1. Тема роботи (проекту) Розробка веб додатку для опитування користувача за допомогою розпізнавання жестів

керівник роботи (проекту) Кудін Олексій Володимирович, доцент, к.ф.-м.н., доцент

(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 09 » червня 2021 року № 851-с

2. Строк подання студентом роботи 24.11.2021

3. Вихідні дані до роботи 1. Постановка задачі.

2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Постановка задачі, аналіз предметної області.

2. Моделювання та проектування програмного доповнення.

3. Реалізація системи

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

Презентація, Діаграма бази даних, діаграми варіантів використання

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Кудін О.В., доцент, к.ф.-м.н.	20.06.2021	
2	Кудін О.В., доцент, к.ф.-м.н.	28.09.2021	
3	Кудін О.В., доцент, к.ф.-м.н.	01.11.2021	

7. Дата видачі завдання 20.06.2021

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	20.06.2021	
2.	Збір вихідних даних.	3.09.2021	
3.	Обробка методичних та теоретичних	16.09.2021	
	джерел.		
4.	Розробка першого розділу.	25.09.2021	
5.	Розробка другого розділу.	14.10.2021	
6.	Оформлення та нормоконтроль	12.11.2021	
	кваліфікаційної роботи.		
7.	Захист кваліфікаційної роботи.	17.12.2021	

Студент _____ А. В. Татаринова
(підпис) (ініціали та прізвище)

Керівник роботи _____ О. В. Кудін
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ С.П. Швидка
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка веб додатку для опитування користувача за допомогою розпізнавання жестів»: 61 с., 31 рис., 5 табл., 23 джерела, 1 додаток.

ANGULAR, FRONT-END, HANDPOSE, NGRX, TENSORFLOW.JS, TYPESCRIPT, WEBSTORM.

Об’єкт дослідження – процес розпізнавання жестів.

Мета роботи: розробка веб-додатку для опитування користувача за допомогою розпізнавання жестів.

Методи дослідження – методи збору та аналізу вимог до програмного забезпечення, методи моделювання, проектування, конструювання та тестування програмного забезпечення.

Предмет дослідження – бібліотека tensorflow.js та її моделі машинного навчання.

При розробці додатку був проведений аналіз предметної області, обрано та спроектовано архітектуру системи за допомогою UML у вигляді наборів діаграм варіантів використання, послідовностей, компонентів, розгортання та діяльності, а також було вивчено та проаналізовано наукові статті з теми. Реалізовано підключення та обробку даних машинної моделі HandPose, додано сервіси для обробки жестів, а також розроблено клієнтську частину за допомогою фреймворку Angular та бібліотек Flex-Layout, Angular Material.

В результаті роботи отримано веб-додаток, яким користувач може керувати за допомогою жестів рук.

SUMMARY

Master’s qualifying paper “Development of the Web Application for User Survey using Gesture Recognition”: 61 pages, 31 figures, 5 tables, 23 references, 1 supplement.

ANGULAR, FRONT-END, HANDPOSE, NGRX, TENSORFLOW.JS, TYPESCRIPT, WEBSTORM.

The object of the study is gesture recognition process.

The aim of the study is development of the web application for user survey using gesture recognition.

The methods of research are methods of collection and analysis of software requirements, modeling, design, construction and testing of software.

The subject of research is the tensorflow.js library and its models of machine learning.

In the development the subject area was analyzed, architecture and development tools modules were chosen. System architecture was designed using UML diagrams in a set of use cases, sequences of components and activities, scientific articles on the topic were studied and analyzed as well. The connection and data processing of the HandPose machine model have been implemented, gesture processing services have been added, and the frontend part has been developed using the Angular framework and the Flex-Layout and Angular Material libraries.

As a result of work a web application that the user can control with hand gestures was received.

ЗМІСТ

ВСТУП

В наш час використання машинного навчання має дуже велику популярність серед різних сфер:

- освіта;
- медицина;
- пошукові системи;
- digital-маркетинг;
- прогнозування на ринку акцій, криптовалют;
- аналітика;
- математичні розрахунки складних систем;
- хімія;
- біологія;
- інженерія і т. д.

Це досить перспективна технологія, що може застосовуватися для різних задач: починаючи від обчислень складних математичних формул, розпізнання рукописної інформації та аналізу документів і закінчуючи керуванням голографічним інтерфейсом, розпізнаванням та аналізом емоційного стану людини та навіть моделюванням поведінки пристроїв і роботів в залежності від різних обставин навколишнього середовища.

Одним із популярних напрямків машинного навчання є аналіз жестів та рухів людського тіла. Потенційні додатки, засновані на такій технології, включають людино-машинну взаємодію, керування віртуальною реальністю та різними пристроями (наприклад, дронами), а також голографічними інтерфейсами, розпізнавання мови жестів і т. д. [1].

Отож метою даної кваліфікаційної роботи є вивчення бібліотек та інструментів, що дозволяють побудувати платформу для людино-машинної взаємодії у заданій сфері – у вебі.

Актуальність дослідження: актуальність теми зумовлена відсутністю подібних веб-додатків.

З огляду на це, можна виділити наступні цілі та задачі даного дослідження:

Мета: розробка веб-додатку для опитування користувача за допомогою розпізнавання жестів.

Задачі:

1) огляд бібліотек та інструментів, що дають змогу побудувати подібний функціонал;
2) сформулювати вимоги до системи;
3) спроєктувати та побудувати архітектуру системи;
4) реалізувати інтерфейс користувача.

Об’єкт дослідження – бібліотека tensorflow.js та її моделі машинного навчання.

Предмет дослідження – створення сервісу, керування яким відбувається за допомогою жестів рук.

Методи дослідження: аналіз, об’єктно-орієнтований, структурно-функціональний, системний підхід.

Перший розділ присвячено огляду моделей машинного навчання, принципів функціонування машинного навчання, javascript бібліотек та інструментів для роботи з ними.

У другому розділі схематично розглянуто етапи проєктування системи.

В межах третього розділу роботи розроблено систему, керування якою можливо здійснювати за допомогою жестів рук.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Машинне навчання. Бібліотеки для роботи з ML

Машинне навчання (від англ. ML – «Machine learning») – це один зі способів застосування штучного інтелекту в комп'ютерних технологіях при роботі з різними даними. Завдяки машинному навчанню програмні додатки можуть точніше прогнозувати результати та аналізувати дані. Основна мета та ідея машинного навчання – дозволити комп'ютерам навчатися самим, автоматично та без втручання людини [2].

Весь процес прийняття рішень штучним інтелектом в машинному навчанні будується на трьох базових параметрах:

1) база даних – вибірки даних різного виду, надані системі розробником;

2) ознаки – всі необхідні задачі, які має виконувати продукт;

3) алгоритми – методики, за якими працює програма виявлення помилок [4].

Машинне навчання – надзвичайно складна система, в основі якої лежить тривалий і трудомісткий процес навчання.

Модель – основна складова частина нейронної мережі, яка реалізує певний заданий алгоритм, що був утворений після навчання.

Завершивши навчання моделі, можливо застосувати її для прийняття рішень та виконання прогнозів за даними, які раніше не зустрічалися. Припустімо, потрібно створити програму для розпізнавання жестів рук користувача. Для цього потрібно навчити модель певним набором зображень позицій рук, кожне з яких позначено тегом певного жесту, а потім застосувати цю модель у програмі для розпізнавання жестів рук користувача.

Методи машинного навчання – це сукупність завдань, вкладених у перевірку припущень, а також пошук оптимальних рішень за допомогою штучного інтелекту.

Виділяють три основні групи методів машинного навчання:

- навчання з учителем;
- навчання без учителя;
- глибоке навчання.

У випадку навчання з учителем в аналітичну систему завантажується масив даних за конкретним завданням і задається мітка – мета аналізу.

При навчанні без учителя, навчання будується на тому, що людині та програмі невідомі правильні відповіді заздалегідь, є лише деякий масив даних. Аналітична машина, обробляючи інформацію, сама шукає взаємозв'язки. Найчастіше на виході маємо неочевидні рішення.

Глибоке машинне навчання часто (але не стовідсотково) пов’язане з аналізом «Великих даних». Тобто у випадку, коли одним комп'ютером (програмою) переробити об’єм інформації просто неможливо або якщо поставлена задача є досить складною та нелінійною (в цьому випадку даних може бути не так вже й багато). Тому використовуються нейронні мережі. Суть такого навчання полягає в тому, що величезне поле інформації поділяється на невеликі сегменти даних, обробка яких делегується іншим пристроям [5].

Існує декілька бібліотек, що дозволяють програмістам створювати, навчати та використовувати моделі машинного навчання.

1.1.1 TensorFlow

TensorFlow – безкоштовна open-source Python бібліотека машинного навчання для високоточних чисельних обчислень. Це найпопулярніша бібліотека машинного та глибокого навчання у сучасному світі.

За допомогою TensorFlow можна побудувати глибокі нейронні мережі для розпізнавання образів і рукописного тексту та рекурентні нейронні мережі для

обробки природніх мов. Також є модулі для векторизації слів та розв'язання диференціальних рівнянь у часткових похідних.

Цей фреймворк має відмінну архітектурну підтримку, що дозволяє з легкістю проводити обчислення на різних платформах, у тому числі на стаціонарних комп’ютерах, серверах і мобільних пристроях.

Дана бібліотека написана на Python, але має є порт JavaScript: tensorflow.js. Його поява пов'язана зі зростанням популярності JavaScript після релізу Node.js [6].

1.1.2 PyTorch

PyTorch – це бібліотека глибокого навчання, створена Facebook, написана на Python та заснована на бібліотеці Torch.

В основному, вона використовується для таких програм, як комп'ютерний зір та обробка природної мови.

Це повністю готова до роботи бібліотека машинного навчання Python з відмінними прикладами, додатками та варіантами використання.

PyTorch відмінно адаптована до графічного процесора (GPU) (як і, власне, Tensorflow), що дозволяє використовувати його, наприклад, у додатках NLP (обробка природніх мов). Глибокі нейронні мережі та тензорні обчислення із прискоренням на GPU – дві основні сильні сторони PyTorch. Бібліотека також включає компілятор машинного навчання під назвою Glow, який досить сильно підвищує продуктивність фреймворків глибокого навчання.

1.1.3 Scikit-learn

Scikit-learn – ще одна відома бібліотека машинного навчання Python з широким спектром алгоритмів кластеризації, регресії та класифікації.

Вона відмінно взаємодіє з іншими науковими бібліотеками Python, такими як NumPy та SciPy, тому що фактично створена на основі NumPy, SciPy і matplotlib [6].

До основних переваг даної бібліотеки відносять:

- зниження розмірності;
- алгоритми, побудовані на вирішальних деревах;
- побудова вирішальних поверхонь;
- аналіз та вибір ознак;
- просунуте імовірнісне моделювання;
- класифікація та кластеризація без вчителя [7].

Ця бібліотека підтримує алгоритми навчання як з учителем, так і без учителя.

1.1.4 Keras

Keras – одна з основних бібліотек Python з відкритим вихідним кодом, написана для побудови нейронних мереж та проєкт машинного навчання. Keras може працювати спільно з Deeplearning4j, MXNet, Microsoft Cognitive Toolkit (CNTK), Theano або TensorFlow.

У цій бібліотеці реалізовані практично всі автономні модулі нейронної мережі, включаючи оптимізатори, нейронні шари, функції активації шарів, схеми ініціалізації, функції витрат та моделі регуляризації. Це дозволяє будувати нові модулі нейромережі, просто додаючи функції чи класи. І оскільки модель вже визначено в коді, розробнику не доводиться створювати для неї окремі файли конфігурації.

Keras особливо зручна для розробників-початківців, які хочуть проєктувати і розробляти власні нейронні мережі, а також для швидкого експериментування з глибокими нейронними мережами [6].

1.1.5 Theano

Theano – це бібліотека Python для швидких числових обчислень (може працювати як на CPU, так і на GPU) і компілятор, що додатково оптимізує пам'ять та обчислювальні потужності машин для маніпулювання та оцінки математичних виразів, особливо матрично-значних [6].

За своєю суттю, це наукова математична бібліотека, яка дозволяє визначати, оптимізувати та обчислювати математичні вирази, у тому числі у вигляді багатовимірних масивів.

Основною більшістю ML і AI додатків є багаторазове обчислення складних та нетривіальних математичних виразів. Theano дозволяє проводити такі обчислення в сотні разів швидше, до того ж вона має модуль для символьного диференціювання, а також пропонує широкі можливості для тестування коду. Її єдиний недолік – не надто простий синтаксис, особливо для новачків [7].

1.2 Задача розпізнавання зображень

Нейронна мережа (НМ) – це математична модель у вигляді програмного та апаратного втілення, що будується на принципах функціонування біологічних нейромереж: основним елементом виступають нейрони, з'єднані між собою, які утворюють шари, кількість яких може бути різною залежно від складності нейромережі та її призначення [12].

Нейромережі застосовують для прогнозування, розпізнавання образів, машинного перекладу, розпізнавання аудіо тощо.

Мабуть, найпопулярнішим завданням нейромереж є розпізнавання візуальних образів: розпізнати символи на папері та банківських картках, підписи на офіційних документах, визначати об'єкти тощо [12].

Розпізнавання зображення належить до завдання введення зображення в нейронну мережу та присвоєння будь-якої мітки для цього зображення. Мітка,

яку виводить мережа, буде відповідати заздалегідь визначеному класу. Можливе присвоєння як відразу кількох класів, так і лише одного. Якщо є лише один клас, зазвичай застосовується термін «розпізнавання», тоді як завдання розпізнавання кількох класів часто називається «класифікацією».

При навчанні НМ для розпізнавання образів з учителем є вибірка з вірними відповідями на питання – мітками класів. Нейромережі подаються на вхід ці зображення, після чого обчислюється помилка, що порівнює вихідні значення з мітками класів. Залежно від ступеня та характеру невідповідності передбачення НМ, її ваги коригуються, відповіді НМ підлаштовуються під вірні, поки помилка не стане мінімальною [12].

У навчанні без вчителя у навчальної вибірки немає міток класів, і перед НМ стоїть завдання знайти заздалегідь невідомі відповіді. Нейронна мережа намагається самостійно знайти закономірності у даних, витягуючи корисні ознаки та аналізуючи їх.

При змішаному навчанні (з частковим залученням вчителя) навчальна вибірка містить як розмічені, так і нерозмічені дані. Цей метод особливо корисний, коли розмітити всі об'єкти – трудомістке завдання. Тим не менш, нейронна мережа може отримати інформацію з невеликої частки розмічених даних і покращити точність передбачень у порівнянні з моделлю, що навчається виключно на нерозмічених даних [12].

1.2.1 Звичайні нейромережі

Звичайною часто називають повнозв'язну нейронну мережу (ПНМ). У ній кожен вузол (крім вхідного та вихідного) виступає як входом, так і виходом, утворюючи прихований шар нейронів, і кожен нейрон наступного шару з'єднаний з усіма нейронами попереднього. Входи подаються з вагами, які в процесі навчання налаштовуються і не змінюються надалі. При цьому кожен нейрон має поріг активації, після проходження якого він приймає одне з двох можливих значень: -1 чи 1, або 0 чи 1 [12].

Подача даних на вхід полягає у виразі двомірної матриці зображення як одномірного вектора. Тобто, для зображення рукописної цифри розміром 28x28 буде використано 784 входи. Далі відбувається вибір архітектури.

Важливо пам’ятати, що кількість нейронів у прихованому шарі має бути значно більше кількості входів, отож кількість нейронів у прихованих шарах візьмемо близько 15000 (10 000 у другому шарі і 5000 у третьому). При цьому для конфігурації з двома прихованими шарами кількість зв'язків, що налаштовуються і навчаються, буде дорівнювати приблизно 10 млн. між входами і першим прихованим шаром + 50 млн. між першим і другим + 50 тис. між другим і вихідним, якщо вважати, що є 10 виходів, кожен з яких позначає цифру від 0 до 9. Разом приблизно 60 000 000 зв'язків.

Проте є недолік – із кожним шаром відбувається втрата топології зображення, тобто взаємозв'язку між окремими частинами. Крім того, завдання розпізнавання передбачає вміння нейромережі бути стійкою до невеликих зміщень, поворотів і зміни масштабу зображення, тобто вона повинна витягувати з даних деякі інваріанти, що не залежать від почерку тієї чи іншої людини, чого не скажеш про звичайну нейронну мережу [10].

1.2.2 Згорткові нейронні мережі

Згорткова нейронна мережа (ЗНМ) має спеціальну архітектуру, яка дозволяє їй максимально ефективно розпізнавати образи. Сама ідея ЗНМ ґрунтується на чергуванні згорткових та субдискретизуючих шарів, причому сама структура є односпрямованою.

ЗНМ отримала свою назву від операції згортки, яка передбачає, що кожен фрагмент зображення буде помножено на ядро згортки поелементно, при цьому отриманий результат повинен підсумовуватися і записуватися в схожу позицію вихідного зображення. Така архітектура забезпечує інваріантність розпізнавання щодо зсуву об'єкта, поступово укрупнюючи «вікно», на яке «дивиться» згортка, виявляючи дедалі більші структури та патерни в зображенні [12].

Ідея згорткових нейронних мереж полягає в чергуванні згорткових шарів (C-layers), субдискретизуючих шарів (S-layers) та наявності повнозв'язних (F-layers) шарів на виході.

Така архітектура містить у собі 3 основні парадигми:

- локальне сприйняття;
- роздільні ваги;
- субдискретизація.

Локальне сприйняття передбачає, що на вхід одного нейрона подається не в повному обсязі зображення (чи вихід з попереднього шару), а лише деяка його область. Такий підхід дозволив зберігати топологію зображення від шару до шару.

Концепція ваг, що розділяються, передбачає, що для великої кількості зв'язків використовується дуже невеликий набір ваг. Тобто, якщо на вході є зображення розміром 32x32 пікселі, то кожен з нейронів наступного шару прийме на вхід тільки невелику ділянку цього зображення розміром, наприклад, 5x5, причому кожен із фрагментів буде оброблений одним і тим же набором. Важливо розуміти, що самих наборів ваг може бути багато, але кожен з них буде застосований до всього зображення. Такі набори часто називають ядрами. Даний механізм забезпечує досить високу якість розпізнавання.

Суть субдискретизації та S-шарів полягає у зменшенні просторової розмірності зображення. Тобто, вхідне зображення грубо (методом усереднення) зменшується в задану кількість разів. Найчастіше в 2 рази, хоча може бути і не рівномірна зміна, наприклад, 2 по вертикалі та 3 по горизонталі. Субдискретизація необхідна для забезпечення інваріантності до масштабу.

Зазвичай після проходження кількох шарів карта ознак вироджується у вектор або у скаляр, але таких карт ознак стає сотні. У такому вигляді вони подаються на один-два шари повної мережі. Вихідний шар такої мережі може мати різні функції активації. Це може бути тангенціальна функція або також успішно використовуються радіальні базові функції [10].

1.2.3 Нейронна мережа адаптивного резонансу

Її переваги перед іншими типами нейромереж у тому, що така мережа вирішує проблему стабільності та пластичності: ці мережі у процесі функціонування не змінюють та не руйнують результати попереднього навчання, а також існує можливість запам'ятовування нових образів.

Нейромережа адаптивного резонансу призначена на вирішення завдання розпізнавання образів. Вона працює з бінарними векторами та навчається без вчителя.

Під час функціонування нейромережі вхідний вектор проходить через два критерії схожості. На першій стадії в пам'яті мережі шукається запам'ятований вектор (еталон), на який найбільше схожий образ, що розпізнається.

Другий критерій подібності визначає, чи досить схожий вхідний вектор на знайдений стандарт. У разі позитивної відповіді запам'ятований образ змінюється, чи навчається, таким чином у пам'яті нейромережі створюється нова категорія для вхідного образу. Навчання відбувається за правилом кон'юнкції: обчислюється логічний перетин елементів вхідного та запам'ятованого векторів [23].

1.2.4 Мережа радіальних базисних функцій

Мережа радіальних базисних функцій – нейронна мережа прямого поширення сигналу, що містить проміжний (прихований) шар радіально-симетричних нейронів. Такий нейрон перетворює відстань від даного вхідного вектора до відповідного йому центру за деяким нелінійним законом (зазвичай функції Гауса) [23].

Штучні нейронні мережі на основі радіально-симетричних (радіально-базових) функцій можуть використовуватися для вирішення широкого кола завдань, серед яких, найчастіше, – апроксимація, класифікація та кластеризація даних [3].

Основна властивість радіально-симетричних функцій – це монотонна та симетрична щодо деякої вертикальної осі симетрії зміна (зменшення або зростання) їх відгуків.

Серед переваг такої архітектури нейронних мереж виділяють:

- наявність єдиного прихованого шару, достатнього для моделювання яскраво виражених нелінійних залежностей;
- простота алгоритму оптимізації вагових коефіцієнтів;
- гарантоване знаходження глобального оптимуму функції помилки при знаходженні вагових коефіцієнтів нейронів вихідного шару;
- висока швидкість навчання.

До обмежень або недоліків нейронних мереж на основі радіально-симетричних функцій можна віднести:

- необхідність спеціального налаштування параметрів радіально-симетричних функцій;
- складність налаштування при великій кількості прихованих радіальних елементів;
- неможливість екстраполювання моделі за межами вихідного інтервалу зміни вхідних значень навчальної вибірки.

1.3 Огляд публікацій на тему розпізнавання жестів

Основною задачею при розпізнаванні жестів є, як правило, визначення розташування та пози руки у просторі. Фактично, завдання будь-якої навченої моделі зводиться до аналізу положення руки та окреслення координат основних точок.

Існує досить багато публікацій, присвячених різним методам та алгоритмам аналізу положення руки.

Наприклад, метод часових згорток (від англ. «Temporal Convolutions») та механізму уваги [13]. Цей метод пов'язаний з новою архітектурою TC-HGR (від

англ. «Temporal Convolutions-Hand Gesture Recognition»), яка призначена для розпізнавання жестів рук за розрідженими багатоканальними сигналами sEMG (від англ. «surface electromyogram»). Запропонована модель продемонструвала, що можна отримувати часову інформацію з сигналу sEMG і покращувати продуктивність в той же час. Більше того, дана архітектура може зменшити необхідну кількість параметрів, що навчаються, в порівнянні з сучасним рівнем техніки, що, в свою чергу, дозволяє знизити складність і в результаті вбудовувати моделі на основі DNN (від англ. «Deep Neural Networks»), наприклад, в контролери протезів.

TC-HGR покликана знизити обчислювальне навантаження при збереженні високої точності, що має першорядне значення перетворення результатів класифікації в плавні дії.

Для навчання та оцінки запропонованої архітектури TC-HGR використано дані, що були зібрані бездротовою ЕМГ-системою Delsys Trigno, яка реєструє електричну активність м'язів на частоті 2 кГц. Цей набір даних складається з сигналів sEMG від 40 здорових суб'єктів (28 чоловіків та 12 жінок віком 29,9 ± 3,9 року, з них 34 правші та шість шульг), що виконують 50 різних жестів руками. Кожен жест повторюється 6 разів, по 5 секунд. з наступним відпочинком 3 секунди.

Запропонований метод привів до 81,65% точності класифікації. Кількість параметрів для навчання запропонованою архітектурою TC-HGR у 11,9 раза менша, ніж у його сучасних аналогів [13].

Таким чином, метод часових згорток та механізму уваги є, безперечно, досить ефективним, коли необхідно впроваджувати нейронну мережу у сервіси та пристрої, які не мають великих ресурсів для проведення складних, великомірних обчислень.

Ще одним досить ефективним методом є використання навченого багатошарового класифікатора перцептрону для системи дистанційного керування [14].

У статті була представлена система дистанційного управління, яка включає сприйняття робота і прогнозування намірів за жестами рук. Модуль сприйняття ідентифікує об'єкти, що знаходяться у робочому просторі робота, а модуль прогнозування намірів – який об'єкт користувач, ймовірніше всього, захоче схопити.

Дистанційне керування – це проміжне рішення для керування роботами в сценаріях, де завдання мають вирішуватися в режимі реального часу, наприклад, при ліквідації наслідків стихійних лих, автономному керуванні або допоміжних пристроях.

Спільне управління було досліджено для ефективного поєднання користувацького та автономного управління під час дистанційної роботи.

Компоненти, необхідні для такої системи:

1) конвеєр сприйняття, здатний ідентифікувати та відстежувати об'єкти;

2) система оцінки намірів, яка може визначити, які об'єкти та як захоплювати;

3) система планування руху, яка може здійснювати точне маніпулювання рухом відповідно до намірів людини.

Для відстеження жестів рук у статті описано контролер Leap Motion – безмаркерний датчик руху, який відстежує жести рук і рухи пальців з частотою до 200 Гц. Його точність нижче 2,5 мм, проте іноді він показує нестабільну роботу через обмежений сенсорний діапазон. Проте його простота і здатність відстежувати руку в 6-D є причинами його використання.

Щоб класифікувати цільовий об'єкт та напрямок захоплення був навчений багатошаровий перцептрон, використовуючи навчання з учителем. Для цього був заданий фіксований набір об'єктів (m = 3) разом з їхнім положенням та двома можливими напрямками захоплення (вгорі/праворуч, n = 2).

Вхідні дані включають вісім функцій: відстані від руки об'єктів, компонент x положення руки, компонент x напрямку руки, x, y-компоненти вектора нормалі долоні та обертання руки по осі y. Модель складається з трьох щільних шарів з

64 прихованих одиниць, які з'єднані з двома окремими шарами з двох одиниць і виводять мітки класів.

У результаті система поєднала в собі певний компроміс у точності прогнозу, а раннє планування і виконання руху, засноване на прогнозуванні цілі, призвело до більш короткої тривалості епізоду (приблизно на 5 секунд швидше, ніж тоді, коли робот почав планування руху після того, як користувач завершив траєкторію).

В цілому, результати показують, що запропонована система управління може поліпшити продуктивність дистанційного керування роботом за допомогою жестів рук.

Однак, оскільки більшість роботи була виконана в змодельованому середовищі, при застосуванні системи в реальних умовах у робота можуть виникнути деякі обмеження [14].

Наступними були розглянуті методи, що засновані на евристиці та нейронній мережі [17].

Було досліджено велику кількість пристроїв та методів введення, і виявлено, що розпізнавання жестів руки на основі skeleton є популярним вибором через його стійкість до змін фону та світла.

Було розроблено два класифікатори жестів з урахуванням різних сценаріїв використання. Обидва класифікатори працюють на рівні ключових точок. Відповідно, точна оцінка пози руки в тривимірному просторі є життєво важливим компонентом для класифікації жестів як на основі кута, так і на основі декількох знімків.

Класифікатор на основі евристики легше створювати та розширювати без необхідності в даних навчання, і він більш інтуїтивно зрозумілий для розробки та усунення несправностей. Класифікатор на основі нейронної мережі (NN) точніший, особливо для граничних випадків.

Розроблений HGR працює в режимі реального часу зі швидкістю 30 кадрів за секунду на звичайних мобільних пристроях і складається з двох частин:

трекера скелета руки, покращеного порівняно з MediaPipe Hands, та класифікатора жестів.

Розглянемо евристичний класифікатор жестів.

На основі трекера скелета руки створюється одноразовий евристичний класифікатор для невеликого набору статичних жестів. Простий підхід до класифікації жестів спочатку виводить набір кутів між різними двовимірними ключовими точками руки, потім застосовує граничні значення до отриманих кутів, щоб визначити дискретний стан для кожного пальця (наприклад, зігнутий або прямий), і нарешті визначає статичний жест як логічний вираз, заснований на станах пальців.

Для оцінки даного підходу було зібрано та анотовано власний набір даних жестів, який містить 1882 короткі відеокліпи, що охоплюють різні кути та умови освітлення для 21 статичного жесту рукою (6 жестів, які використовуються як позитивні вибірки, і 15 жестів – як негативні).

В результаті класифікатор досягає 0,86% помилкових спрацьовувань і 44,4% відкликів відповідно до набору даних.

Розглянемо класифікатор жестів з використанням нейронної мережі.

Було зібрано та проаналізовано набір даних, що містить 7307 зображень від 6478 користувачів, що представляють широкий спектр форм рук як жестів, так і не жестів у дикій природі. На додаток до звичайних позитивних зразків також надано прості та складні негативні зразки.

Класифікатор нейронної мережі був навчений розрізняти шість статичних жестів рук, а також фоновий клас Negative.

Модель складається з 3 повністю пов'язаних шарів по 50 нейронів у кожному. Вхідними даними моделі є внутрішні та зовнішні характеристики, обчислені при евристичній класифікації.

Даний класифікатор досягає середньої швидкості відклику 87,9% за 6 класами статичних жестів при частоті помилкових спрацьовувань 1%.

В результаті виконаної роботи, був отриманий розпізнавач жестів рук, який може використовуватися для програм, таких як віртуальний сенсорний

екран для настільних комп'ютерів, надання візуальних команд роботам, контролер для ігрових систем віртуальної реальності та пульт дистанційного керування для великих екранів [17].

Розглянемо ще один приклад використання нейронної мережі, в основі якої лежить ідея трансферного навчання [19].

Глибоке навчання – це гілка машинного навчання, заснована на штучній нейронній мережі (ШНС), здатній імітувати поведінку людського мозку. ШНС складається з кількох прихованих шарів з декількома прихованими блоками (нейронами).

Це новий підхід, який широко застосовується в таких галузях, як семантичний та синтаксичний аналіз, трансферне навчання, комп'ютерний зір, обробка природньої мови тощо.

По суті, моделі глибокого навчання вимагають тисяч вибірок даних та важких обчислювальних ресурсів, таких як графічний процесор, для точної класифікації та прогнозного аналізу. Однак існує гілка машинного навчання, широко відома як «трансферне навчання», яка не обов'язково потребує великих обсягів даних для оцінки.

Трансферне навчання – це метод машинного навчання, у якому модель, розроблена для одного завдання, повторно використовується для другого, пов'язаного з ним. Це стосується ситуації, коли «знаходження» одного налаштування використовується для покращення оптимізації в іншому.

Трансферне навчання зазвичай застосовується до нового набору даних, який зазвичай менший, ніж вихідний набір даних, що використовується для навчання попередньо навченої моделі. Трансферне навчання може вирішити основну проблему нестачі даних для навчання.

Набір даних, в основі якого відео, що складається з жестів миття рук, було створено за допомогою 30 учасників. Тривалість відео складає 25-30 секунд. За кожним етапом миття рук була пауза, під час якої учаснику пропонувалося прибрати руки від камери.

Формат відео для цього набору даних – файл MP4 з розміром діапазону 40-60 МБ та частотою кадрів 29,84 к/с. Усі шість рухів миття рук були записані в одному відео кожного учасника.

Крім набору даних про миття рук, також були записані в окремому відеозаписі жести однією рукою, такі як лінійні та кругові рухи руки.

У цій статті виконується попередня класифікація для виявлення наявності однієї руки, двох рук, відсутності руки із створеного набору даних.

Відеозаписи були розкладені на 704 зображення та розділені на три класи. Зображення в класах були рівномірно розподілені, щоб уникнути перекосу при навчанні моделі.

В результаті, у міру збільшення кількості тренувальних кроків (епох), крива втрат продовжує знижуватися для обох наборів. До останнього етапу навчання показник точності досяг 1,0%.

100% точність досягається через відсутність надійного набору даних та меншої кількості даних, що використовуються в цьому експерименті; однак трансферне навчання дає багатообіцяючі результати та робить правильні прогнози для вхідних зображень [19].

Наступний метод присвячений розпізнаванню жестів на смартфонах за допомогою нечутних звуків та глибокого навчання [15].

Пропоноване рішення полягає в тому, щоб мати активну акустичну сенсорну систему для класифікації різних жестів рук.

Суть методу активного сприйняття полягає в тому, що звуковий сигнал випромінюється і приймається одним і тим самим смартфоном. У такому разі вбудовані динаміки смартфона передають звуковий сигнал на певній частоті та записують зміни сигналу від руху руки навколо смартфона на основі ефекту Доплера.

Для виявлення жестів рук було використано ультразвукову хвилю, тому що ультразвук не чутний людському слуху, на такі звукові сигнали не впливають умови освітлення (тому система може працювати вдень та вночі без перебоїв), має високу точність визначення дальності та низьку вартість розгортання.

Крім того, система записує сигнал одночасно, діючи як активна система сонара. Записуючи сигнали, застосовується короткочасне перетворення Фур'є до форми сигналу, щоб перетворити тимчасову область в частотну з метою візуалізації жесту чи руху руки як ефект Доплера на спектрограмі. Після цього, використовуючи методи глибокого навчання, відбувається класифікація жестів рук.

У роботі було використано різні типи моделей CNN із варіаціями на вході. Однак, всі моделі CNN приймають вхідні дані у вигляді спектрограми. Перша модель сприймає двоканальний звук як одноканальне зображення спектрограми (Basic CNN). Друга – приймає два входи спектрограми, один як верхній мікрофон, а інший – як нижній, а потім ці дані об'єднуються і вводяться в модель (Раннє злиття). Третя – приймає два входи спектрограм верхнього та нижнього каналів як два окремі входи, навчає кожен окремо, після чого об'єднує виведені дві моделі для отримання результату (Пізнє злиття).

Дані були зібрані в зашумленому та не зашумленому середовищі. Шум включав звук кондиціонера, розмови людей та музику. Дані було розділено на аудіо файл (.WAV) та зображення спектрограми. Зображення спектрограми були відфільтровані між 19,7 та 20,3 кГц, оскільки це діапазон, в якому відображаються сонар та жест. Він також видаляє шум, що походить від нижчих частот, без використання смугового фільтра. Причому дані записувалися як стерео (таким чином, вони містять два канали: верхній та нижній мікрофони).

Набір даних для навчання було складено 4 суб'єктами, які мали загалом 1920 навчальних аудіо файлів (.wav) та зображень спектрограм таких аудіо файлів.

Набір даних тестування включає 576 аудіо файлів та зображень спектрограм, і 3 суб'єкти зібрали їх (крім тих, що включені до набору даних для навчання). Розмір набору даних 576, тому що він становить 30% навчального набору даних. Цей набір даних використовувався для тестування CNN моделей.

В результаті тестування було виявлено, що об'єднання двоканальних входів дало кращий результат порівняно з методом одноканального введення, а

також, що використання поділу спектрограми для верхнього та нижнього мікрофонів дає більш високі результати, ніж використання комбінованої спектрограми.

Зрештою, система успішно класифікувала 6 жестів рук з точністю 93,58% [15].

1.4 Технічне завдання

1.4.1 Введення

Система є веб-додатком, керування яким можливо здійснювати за допомогою жестів рук, тобто, використовуючи жестовий інтерфейс в режимі реального часу, користувач має змогу змінювати сторінки додатку, давати відповідь на запитання, пропускати його або повертатися до попереднього.

Метою створення системи є:

1) з точки зору творців системи:
   - сформувати клієнтську базу для надання додаткових послуг;
   - побудувати сервіс, що дає змогу користувачам спробувати користування жестовим інтерфесом;
2) з точки зору клієнта:
   - впровадження нового сервісу в свої клієнтські веб-додатки, а потім і в більш великі сторонні системи;
   - зацікавити нових потенційних клієнтів;
   - спростити процедуру керування веб-додатком.

1.4.2 Загальний опис

Продукт призначений для надання можливості користувачу керувати інтерфейсом за допомогою жестів рук.

Продукт зчитує відеопотік з веб-камери, аналізує зображення та відображає оновлений відеопотік з нанесеними координатами руки.

Доцільно також описати функції продукту:

1) система дозволяє користувачеві давати відповіді на запитання за допомогою двох жестів («Так» і «Ні»);

2) система дозволяє перейти до наступного питання використовуючи жест «Свайп вліво»;

3) система дозволяє повернутися до попереднього питання використовуючи жест «Свайп вправо»;

4) система відображає прилади доступних жестів на сторінці з питанням.

Також в системи є певні обмеження:

- використання останніх версій браузерів: Mozilla Firefox, Chrome;
- швидкість інтернету не нижче 3G.

1.4.3 Детальні вимоги

Інтерфейси користувача:

1) інтерфейс користувача повинен надавати можливість відповідати на запитання як за допомогою жестів, так за допомогою миші;

2) система повинна відображати коректно інтерфейс користувача з розширенням від 1024x1366 до більш ніж 1920x1080 пікселів;

3) система повинна відображати приклади усіх доступних для використання жестів;

4) система повинна відображати список усіх доступних жестів.

Функціональні вимоги:

- увійти до системи;
- увести свої дані у форму;
- відповісти на питання жестом;
- відповісти на питання мишею;

- пропустити питання;
- повернутися до попереднього питання;
- змінити свою відповідь;
- переглянути статистику відповідей.

Нефункціональні вимоги:

- система повинна працювати для наступних браузерів останніх версій: Mozilla Firefox, Google Chrome, Opera, Safari;
- система повинна стабільно працювати з 3G швидкістю інтернету;
- система не повинна дозволяти доступ до питань, якщо користувач не заповнив форму «Про себе» при вході до системи.

1.5 Висновки до розділу 1

В розділі 1 було розглянуто загальні принципи машинного навчання, розглянуто основні етапи створення та використання цієї технології, а також приклади її практичного застосування у різних сферах діяльності людини.

Було проведено аналіз бібліотек, що дають можливість побудувати, інтегрувати та використовувати технологію машинного навчання, їх переваг та недоліків.

Додатково було проаналізовано статті з даної теми.

У першому розділі також представлено технічне завдання до розроблюваної системи.

Відповідно до результатів проведеного аналізу предметної області та доступних засобів реалізації додатку було виділено наступні технології для досягнення мети кваліфікаційної роботи:

- бібліотека tensorflow.js;
- ML модель HandPose;
- Angular framework.

2 ПРОЄКТУВАННЯ

2.1 Загальна логіка системи

Система дозволяє користувачу керувати додатком за допомогою жестів рук.

Для використання функціоналу сервісу користувачу потрібно перейти на сайт, заповнити форму «Про себе» та перейти, власне, до анкетування.

Під час відповіді на питання, користувач жестами може обирати потрібну йому відповідь «Так» чи «Ні», а також пропустити поточне питання чи повернутися до попереднього. В кінці анкетування користувач може переглянути статистику своїх відповідей.

2.2 Проєктування системи

Проєктування ПЗ – процес визначення архітектури, компонентів, інтерфейсів, інших характеристик системи й кінцевого результату [11].

Для проєктування інформаційних систем широко використовується мова UML – уніфікована мова візуального моделювання. Це мова візуалізації, специфікації, конструювання і документування програмних систем [9].

2.2.1 Етап концептуального проєктування

На даному етапі було розроблено діаграму варіантів використання.

Діаграма варіантів використання (діаграма прецедентів) (див. рис. 2.1) – це діаграма, на якій зображено відношення між акторами та прецедентами в системі [8].

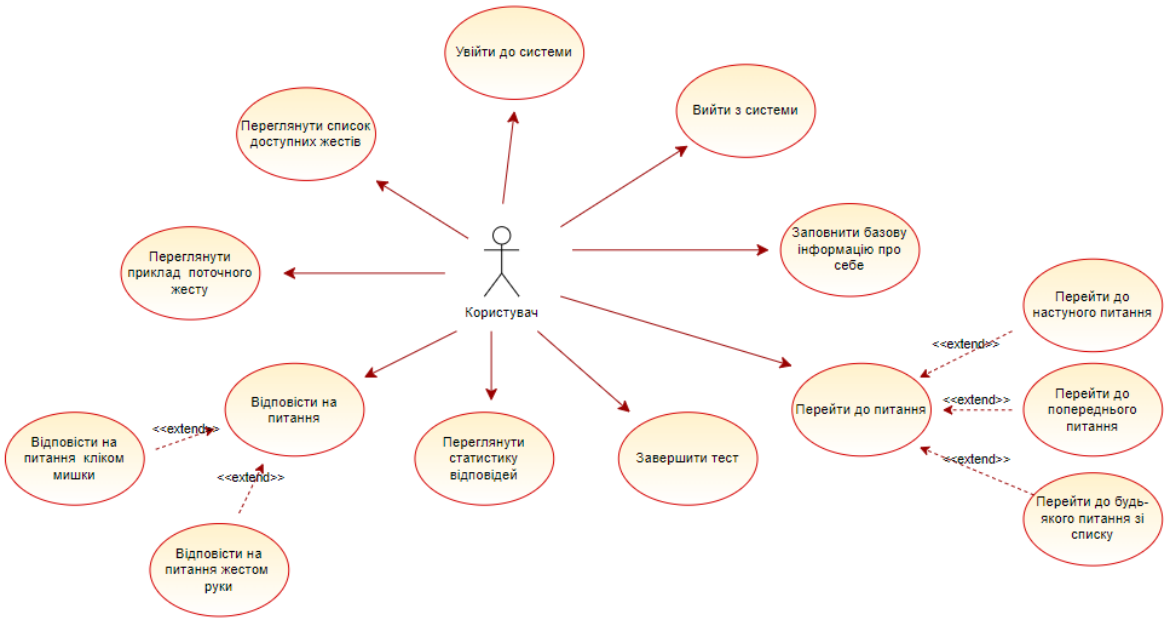


Рисунок 2.1 – Діаграма прецедентів

Також на цьому етапі доцільно описати основні варіанти використання.

Прецедент «Заповнити базову інформацію про себе».

Призначення: даний варіант використання надає можливість користувачу додати інформацію про себе до системи.

Основний потік подій: прецедент починає виконуватися, коли користувач входить до системи. Система пропонує вказати ім’я, прізвище та вік користувача. Після того, як користувач ввів дані, система перенаправляє його на сторінку з першим питанням тесту.

Альтернативний потік: якщо користувач не вводить дані або не повністю заповнює форму, відображається повідомлення про помилку.

Передумова: користувач повинен увійти до системи.

Прецедент «Відповісти на питання жестом руки».

Призначення: прецедент надає можливість відповісти на питання тесту жестом руки.

Основний потік подій: даний варіант використання починає виконуватися, коли користувач переходить на сторінку питання тесту. Система відображає віконце з відео з веб-камери користувача та відео з прикладами жестів для відповіді.

Альтернативний потік: якщо користувач не бажає/не має змоги відповісти жестом, він може обрати потрібний варіант кліком миші.

Передумова: користувач повинен увійти до системи.

Прецедент «Перейти до наступного питання».

Призначення: даний прецедент надає можливість перейти до наступного питання тесту, не відповідаючи на поточне.

Основний потік подій: варіант використання починає виконуватися, коли користувач переходить на сторінку тесту та бажає пропустити поточне питання.

Передумова: користувач повинен увійти до системи та перейти до сторінки з питанням.

Прецедент «Переглянути список доступних тестів».

Призначення: даний прецедент надає можливість переглянути усі доступні жести.

Основний потік подій: варіант використання починає виконуватися, коли користувач переходить на сторінку тесту. Система відображає відео доступного жесту та користувач може переключити режим відображення на «Список».

Передумова: користувач повинен увійти до системи та перейти до сторінки з питанням.

Прецедент «Переглянути статистику відповідей».

Призначення: даний варіант використання надає можливість переглянути відповіді на усі питання тесту.

Основний потік подій: прецедент починає виконуватися, коли користувач відповідає на останнє питання тесту.

Передумова: користувач повинен увійти до системи та завершити відповідь на останнє питання тесту.

Також на даному рівні проєктування було розроблено діаграму послідовності.

На цій діаграмі (див. рис. 2.2-2.3) показано процес тестування в цілому та більш детально послідовність подій, що відбуваються під час відповіді на запитання тесту.

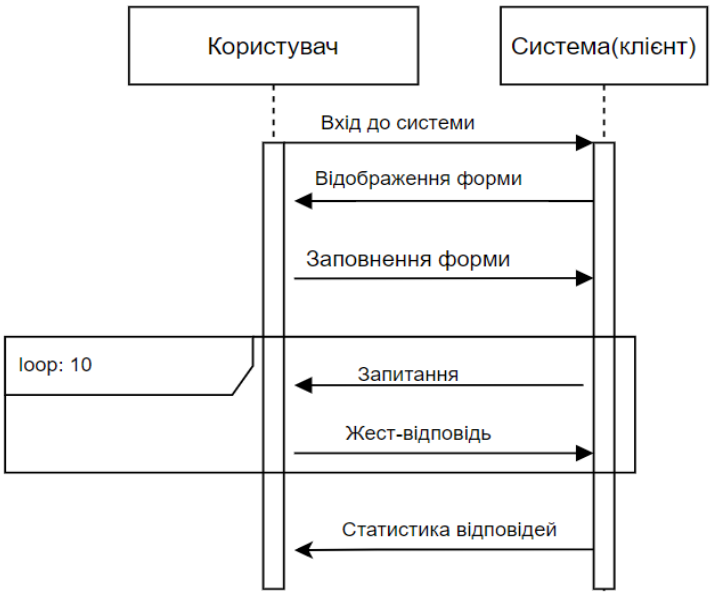


Рисунок 2.2 – Діаграма послідовності «Процес тестування»

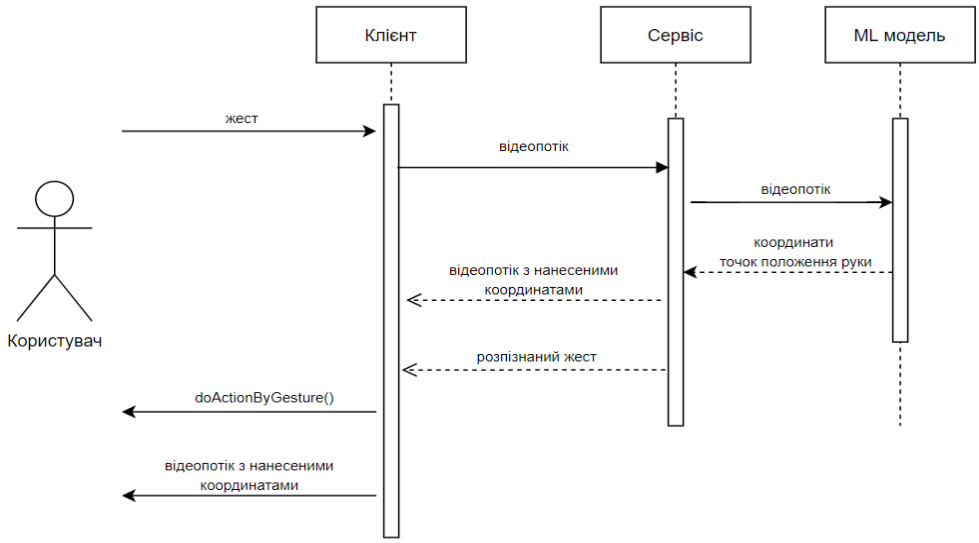


Рисунок 2.3 – Діаграма послідовності «Відповідь на запитання тесту»

2.2.2 Етап логічного проєктування

На етапі логічного проєктування розроблено діаграму діяльності.

На ній (див. рис. 2.4) показано процес взаємодії користувача з додатком від моменту входу в систему до завершення анкетування.

Основна логіка системи реалізовується при відповіді на питання, коли користувач повністю взаємодіє з додатком жестами руки.

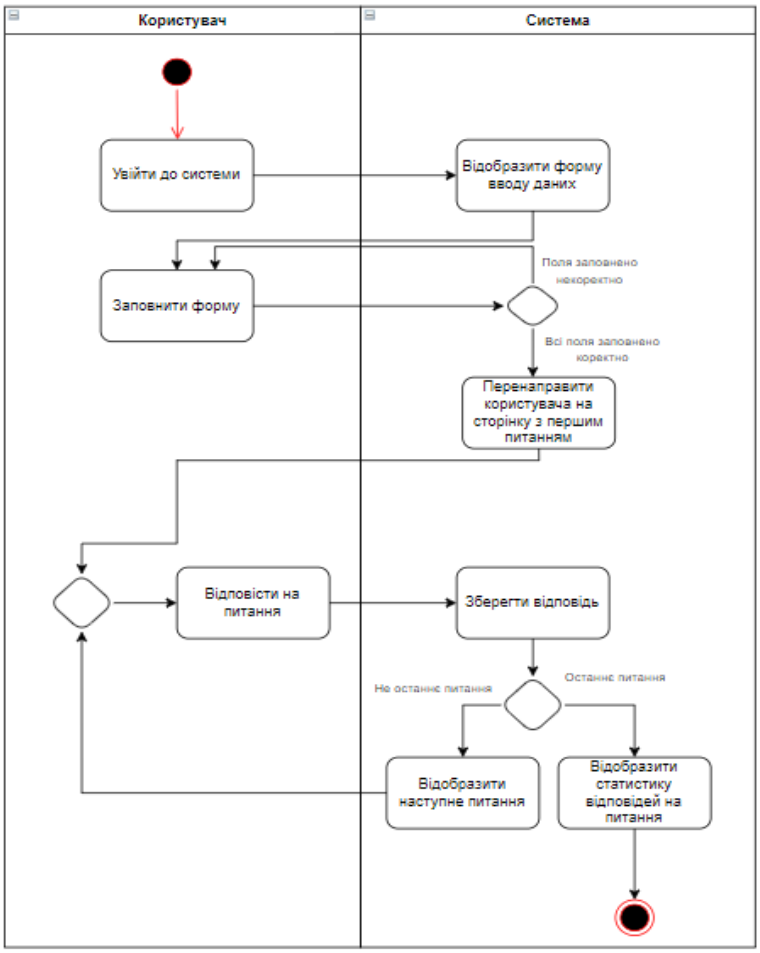


Рисунок 2.4 – Діаграма діяльності

2.2.3 Етап фізичного проєктування

Етап фізичного проєктування полягає в створенні діаграми розгортання, на якій зображені вузли, які є фізично наявними елементами системи – сервер, сервер бази даних, персональний комп’ютер (див. рис. 2.5).

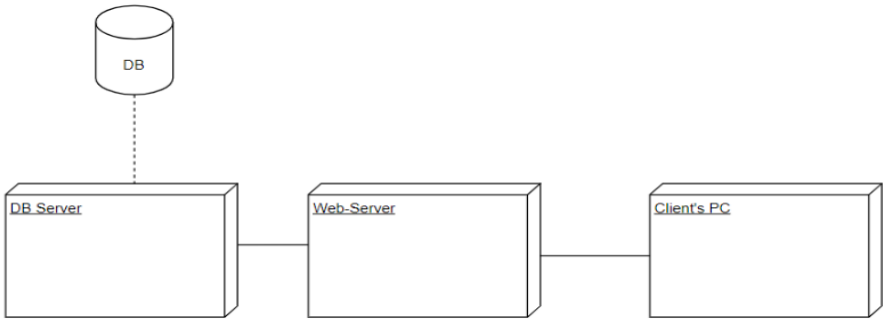


Рисунок 2.5 – Діаграма розгортання системи

2.2.4 Проєктування структури даних

На даному етапі було спроєктовано загальну структуру інформаційної моделі даних розроблюваної системи (див. рис. 2.6).

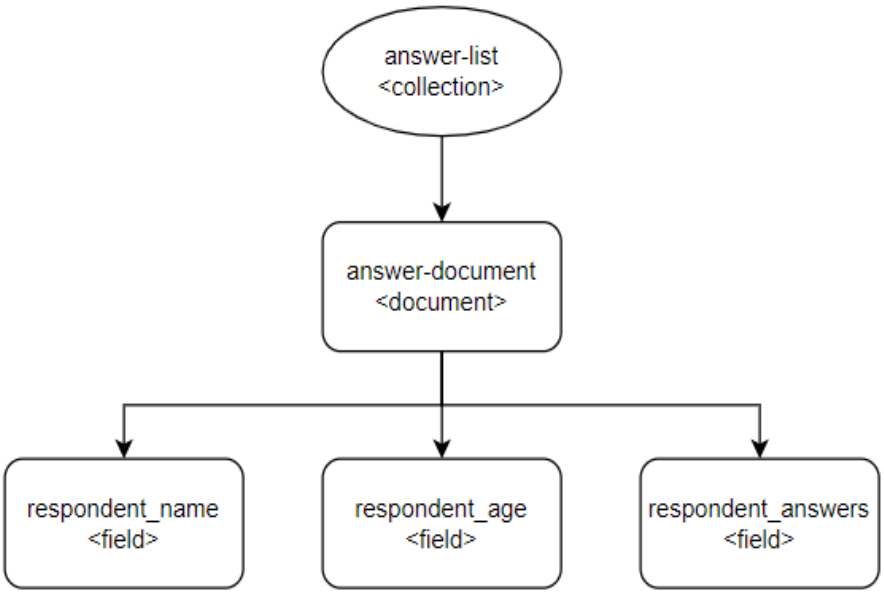


Рисунок 2.6 – Модель структури даних системи

Розглянемо детальніше основну структуру інформаційної моделі системи.

Оскільки Firestore є NoSQL базою даних, то дані зберігаються у вигляді колекції, яка, своєю чергою, поділяється на документи. За бажанням, документ можна назвати або дозволити СУБД присвоїти документу унікальний id, що складається з 15-20 буквено-цифрових символів.

У документі «answer-document» зберігаються дані про всі проходження опитування (див. табл. 2.1).

Таблиця 2.1 – Answer-document

Назва поля	Тип даних	Опис
respondent_first_name	string	Ім’я користувача
respondent_last_name	string	Прізвище користувача
respondent_age	string	Вік користувача
respondent_answers	Array({question_text: string, answer: string})	Відповідь на питання

2.2.5 Проєктування архітектури додатку

На даному етапі була розроблена NGRX-структура додатку (див. рис. 2.7).

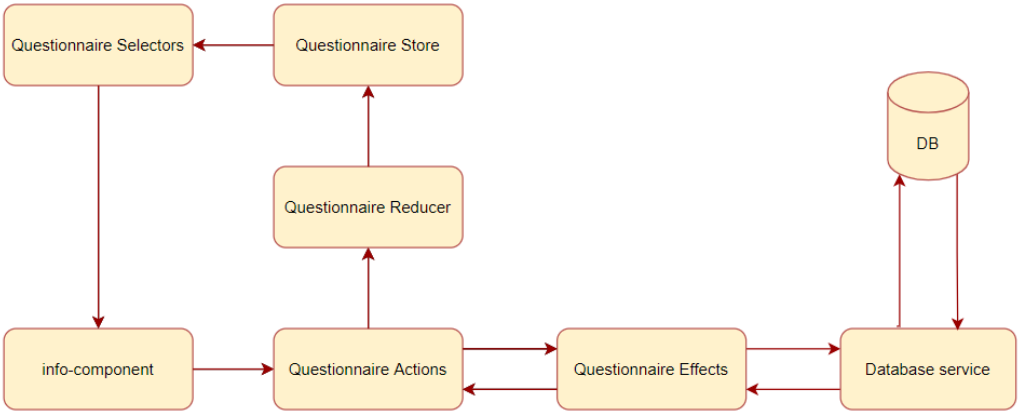


Рисунок 2.7 – NGRX-структура компонентів системи

Розглянемо більш детально структуру компонентів додатку.

Компонент є основною точкою входу, з якої запускається екшен. У структурі додатку задіяно 13 екшенів (див. табл. 2.2).

Таблиця 2.2 – Екшени

Назва екшену	Опис
loadQuestionnaire	Відпрацьовує при завантаженні сторінки з першим питанням, ініціює завантаження списку питань.
loadQuestionnaireSuccess	Відпрацьовує у разі успішного завантаження списку питань.
loadQuestionnaireFailure	Відпрацьовує у разі помилки в процесі завантаження списку питань
loadWebcamQuestionnaire	Відпрацьовує при завантаженні сторінки з першим питанням, ініціює завантаження зображення з вебкамери.
loadWebcamQuestionnaireSuccess	Відпрацьовує у разі успішного завантаження зображення з вебкамери.
loadWebcamQuestionnaireFailure	Відпрацьовує у разі помилки при завантаженні зображення з вебкамери.
changeGesture	Відпрацьовує, коли був зафіксований новий жест
changeNextPrevIndexOfAnswerCard	Відпрацьовує при зміні поточного питання
answerOnCurrentQuestion	Відпрацьовує, коли користувач відповів на поточне питання
allAnswered	Відпрацьовує, коли користувач відповів на усі питання
sendAnswers	Відпрацьовує, коли система відправляє до БД відповіді користувача
sendAnswersSuccess	Відпрацьовує у разі успішної відправки відповіді користувача.
sendAnswersFailure	Відпрацьовує у разі помилки в процесі відповіді користувача.

Для реагування на екшени, а також ініціювання запит до серверу та відправки даних до бази в системі реалізовано 3 ефекти (див. табл. 2.3).

Завдяки своїй структурі, ефекти можуть як реагувати, так і ініціювати запуск екшенів у системі, як правило, отримавши відповідь про статус виконання запиту від серверної частини чи бази даних.

Таблиця 2.3 – Ефекти

Назва ефекту	Опис
loadQuestionnaire	Запускається, коли відпрацьовує loadQuestionnaire екшен, робить запит на сервер та повертає loadQuestionnaireSuccess або loadQuestionnaireFailure в залежності від статусу з бекенду.
allAnswered	Запускається, коли відпрацьовує answerOnCurrentQuestion та запускає allAnswered екшен, що, в свою чергу, запускає редюсер для оновлення інформації у сторі.
sendAnswers	Запускається, коли відпрацьовує sendAnswers екшен, робить запит на сервер та повертає sendAnswersSuccess або sendAnswersFailure в залежності від статусу з бекенду.

Для отримання даних, що зберігаються у хранилищі, в NGRX-структурних додатках використовують селектори (див. табл. 2.4). Як правило, для отримання останніх актуальних даних на селектори необхідно здійснити підписку.

Таблиця 2.4 – Селектори

Назва ефекту	Опис
getLoadingWebcamStatus	Повертає статус завантаження зображення з камери.
getErrorLoadedWebcamStatus	Повертає помилку, що виникла при завантаженні зображення з камери.
getCurrentGesture	Повертає поточний жест.
getLoadingQuestionnaireStatus	Повертає статус завантаження питання.
getErrorLoadedQuestionnaireStatus	Повертає помилку, що виникла при завантаженні питання.
getQuestionnaireAll	Повертає список всіх питань.
getQuestionnaireTotal	Повертає загальну кількість питань.
getIndexOfAnswerCard	Повертає номер поточного питання.
getAllAnsweredStatus	Повертає значення флагу. що містить дані про те, чи на всі питання було дано відповідь.

3 РЕАЛІЗАЦІЯ

3.1 Опис технологій

3.1.1 NGRX

NGRX – це фреймворк для створення реактивних додатків на Angular [18].

Ця бібліотека реалізує принцип роботи Redux для додатків Angular. Головна мета NGRX – централізувати та зробити максимально зрозумілим керування всіма станами програми.

Мета досягається завдяки закладеним у бібліотеці кільком фундаментальним принципам:

- наявність єдиного джерела даних про стан – Сховище (Store);
- доступність стану тільки для читання;
- зміна стану здійснюється тільки через екшени (actions), які обробляються редюсерами (reducers), що є чистими функціями [16].

Складовими частинами архітектури NGRX є екшени (actions), редюсери (reducers), ефекти (effects), селектори (selectors) та Сховище (Store) (див. рис. 3.1).

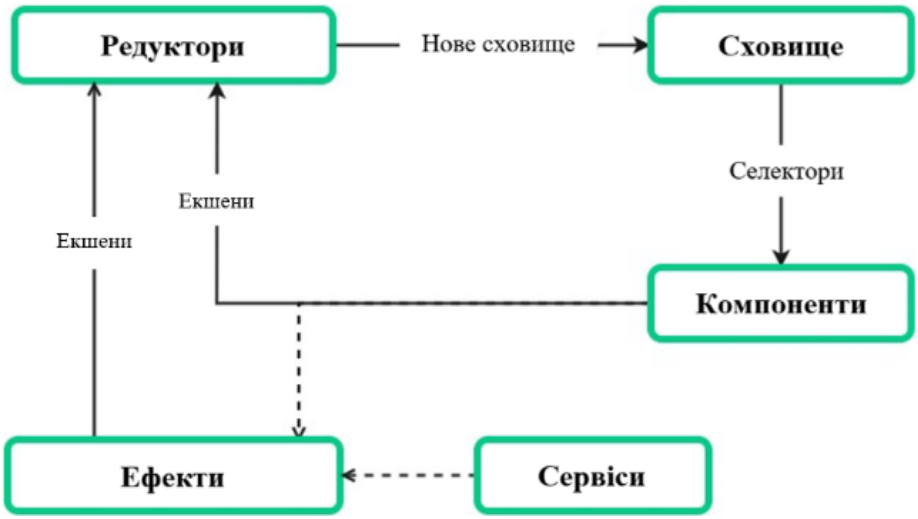


Рисунок 3.1 – Складові NGRX структури

Екшени відображають унікальні події, що відбуваються в Angular додатку, та використовуються для передачі даних у Сховище. Екшени є єдиним джерелом даних для Сховища.

Якщо екшени описують як щось, що сталося в додатку, то редюсери визначають, як має змінитися стан Angular програми у відповідь на екшен, що виник.

NGRX Reducers реєструються у сховищі та мають бути чистими функціями. Редюсер приймає вихідний стан і екшен, що виник, як аргументи, але для застосування змін він повинен повернути новий стан, в жодному разі не змінювати безпосередньо вихідний.

Якщо в редюсері не передбачена обробка екшену, то він просто повинен повернути вихідний стан.

Сховище містить у собі повний стан всього додатка і є у єдиному екземплярі. Формування NGRX Store відбувається шляхом об'єднання станів, що повертаються кожним із редюсерів (хоча найпростіший Angular додаток може обійтися і одним редюсером).

Основні функції NGRX Store:

- зберігання стану програми та надання до неї доступу;
- надання можливості оновити стан за допомогою методу dispatch() через певні екшени;
- реєстрація функцій за допомогою subscribe(), які будуть викликані за будь-якої зміни стану.

Створення екшенів здійснюється за допомогою інтерфейсу Action, який має єдине обов'язкове строкове поле type.

Селектори є чистими функціями і використовуються для отримання певних частин глобального стану. Відмінні риси селекторів:

- мобильність;
- мемоізація;
- можливість побудови композиції селекторів;
- легкість тестування.

NGRX Effects реалізують граничні ефекти, що працюють на основі бібліотеки RxJS. Відстежуючи потік екшенів, що відправляються в Store, вони можуть генерувати нові екшени, наприклад, на основі результатів виконання запитів HTTP або повідомлень, отриманих через Web Sockets.

Цілі та функції NGRX Effects:

- зняти навантаження з компонента з управління станом та виконання побічних ефектів та звести його роботу до отримання станів та генерації екшенів;
- відстеження та фільтрація потоку екшенів для виконання ефекту при виникненні конкретного з них;
- виконання синхронних та асинхронних ефектів [16].

Отже, NGRX є основою більшості додатків, створених на основі фреймворку Angular. Розглянемо переваги та недоліки використання цієї бібліотеки.

Переваги використання NGRX:

- оскільки є єдине джерело правди, безпосередньо змінити стан додатку не є можливим, завдяки чому додатки працюватимуть більш узгоджено;
- використання redux шаблону дає багато цікавих функцій, що полегшують відладку;
- тестування додатків стає простішим, оскільки введені прості функції для обробки змін стану, а також тому, що обидва (NGRX і rxjs) мають безліч зручних засобів для тестування.

Недоліки використання NGRX:

- NGRX потребує певного досвіду чи глибокого розуміння деяких програмних патернів, для початківців дана архітектура спочатку може бути трохи заплутаною;
- кожного разу, коли додається будь-яка властивість у стан, потрібно додавати екшени, редюсери, можливо оновити або додати селектори, ефекти, якщо такі є, та оновити Сховище;
- NGRX не є частиною Angular бібліотек ядра [20].

3.1.2 RxJS

Реактивне програмування – це парадигма асинхронного програмування, пов'язана з потоками даних та поширенням змін.

RxJS (Reactive Extensions for JavaScript) – це бібліотека для реактивного програмування з використанням об'єктів, що спостерігаються, яка спрощує створення асинхронного коду.

RxJS надає реалізацію Observable типу, який необхідний доти, доки він не стане частиною мови і поки браузери не підтримають його. Бібліотека також надає службові функції для створення об'єктів, що спостерігаються, і роботи з ними.

Ці службові функції можна використовувати для:

- перетворення існуючого асинхронного коду на змінні, що спостерігаються:
- перебору значень у потоці;
- зіставлення значень з різними типами;
- фільтрації потоків;
- складання кількох потоків [21].

Основні концепції RxJS, які реалізують асинхронне керування подіями:

- observable: представляє ідею викликаної колекції майбутніх значень чи подій;
- observer: це набір зворотних викликів, який знає, як прослуховувати значення, що передаються Observable;
- subscription: представляє виконання Observable, в першу чергу корисне для скасування виконання;
- operators: чисті функції, які дозволяють функціональному стилю програмування мати справу з операціями над колекціями (наприклад, map, filter, concat, reduce тощо);
- subject: еквівалент EventEmitter та єдиний спосіб багатоадресної передачі значення або події кільком спостерігачам;

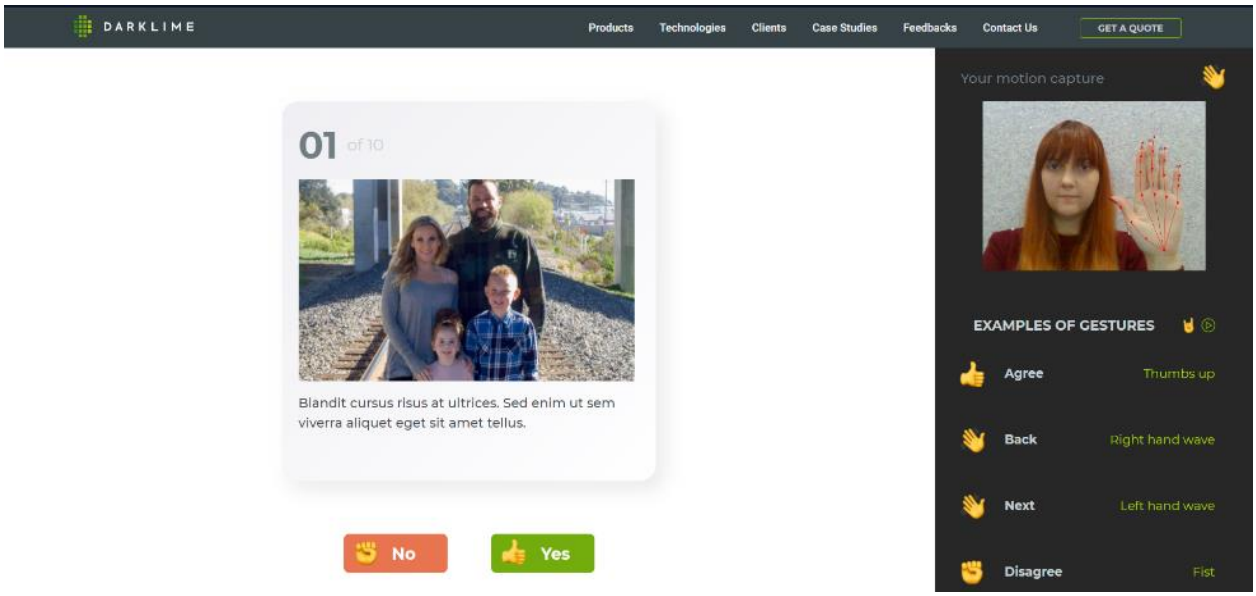
– schedulers: централізовані диспетчери управління паралелізмом, що дозволяють координувати, коли обчислення відбуваються на, наприклад, setTimeout або requestAnimationFrame тощо [22].

3.2 Ядро системи

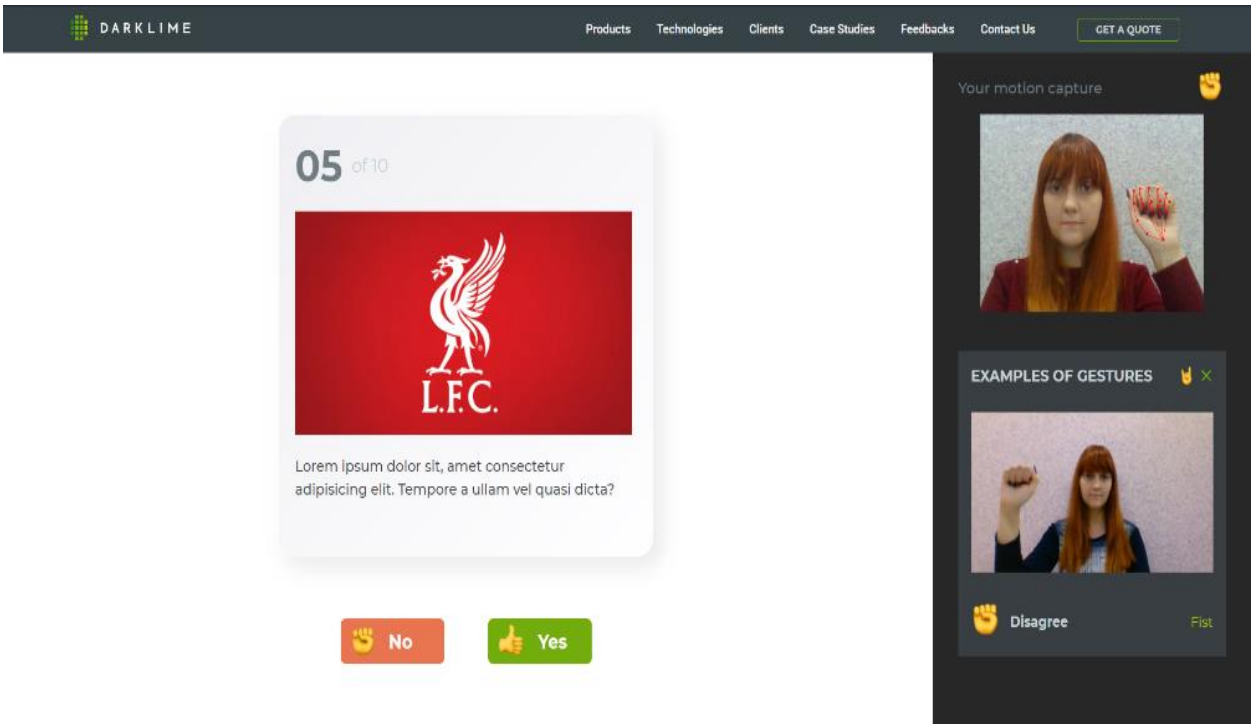
3.2.1 Розробка інтерфейсу користувача

Інтерфейс користувача є однією з найважливіших складових програмного продукту, адже саме від зручності та функціональності цієї частини додатку залежить, чи буде користувач використовувати його. Саме тому важливо якісно реалізувати цю складову.

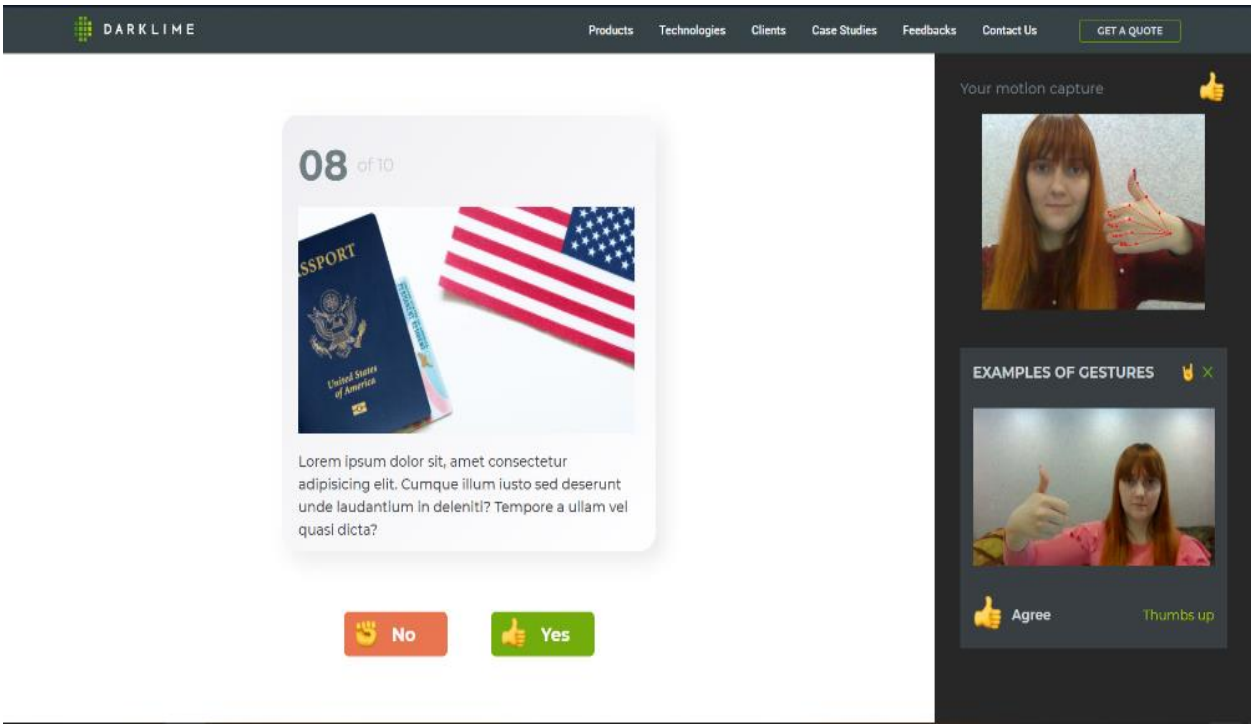
Найперше, що було розроблено – система реагування на жести, спочатку 1-2 жести і керування лише одним елементом сторінки, потім – декількома, і на останньому етапі – керування перемиканням сторінок чотирма жестами з можливістю розпізнавання ще двох. Після цього було реалізовано стильове оформлення додатку. І в результаті отримано наступний дизайн (див. рис. 3.2).



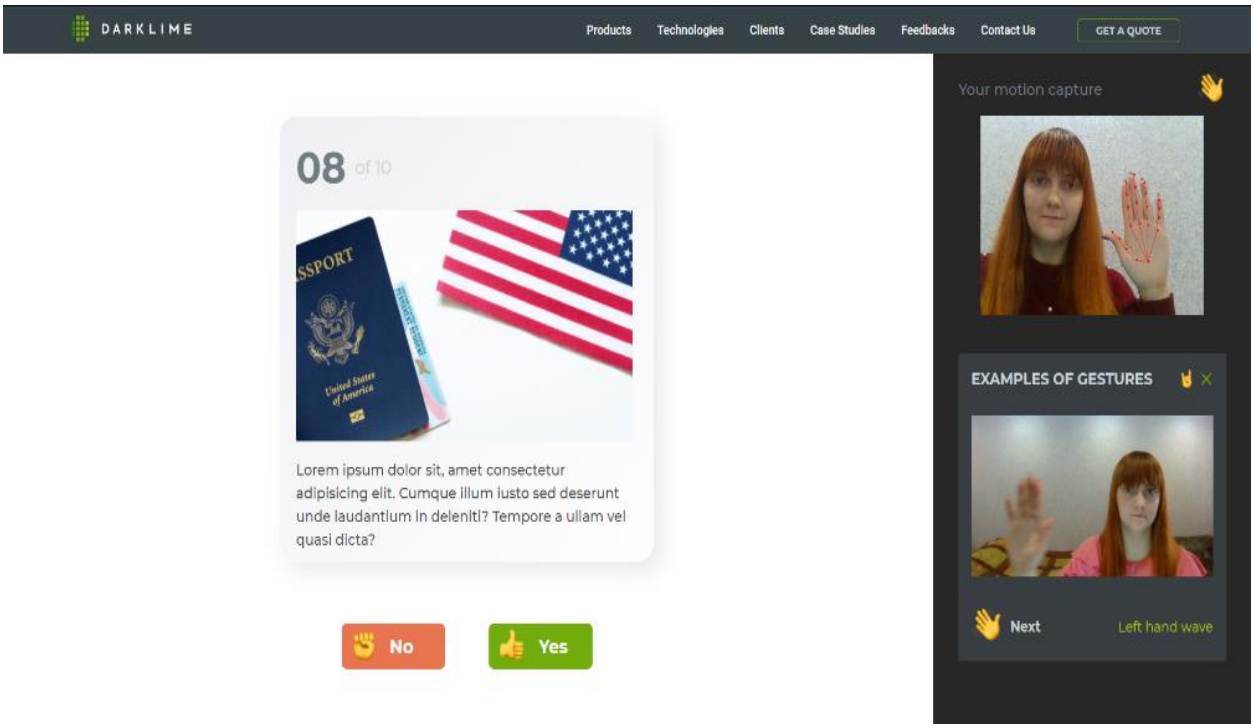
а)


DARKLIME
Products
Technologies
Clients
Case Studies
Feedbacks
Contact Us
GET A QUOTE
05 of 10
L.F.C.
Lorem ipsum dolor sit, amet consectetur adipisicing elit. Tempore a ullam vel quasi dicta?
No
Yes
Your motion capture
EXAMPLES OF GESTURES
Disagree
Fist

б)


DARKLIME
Products
Technologies
Clients
Case Studies
Feedbacks
Contact Us
GET A QUOTE
08 of 10
Lorem ipsum dolor sit, amet consectetur adipisicing elit. Cumque illum iusto sed deserunt unde laudantium in deleniti? Tempore a ullam vel quasi dicta?
No
Yes
Your motion capture
EXAMPLES OF GESTURES
Agree
Thumbs up

в)



г)

Рисунок 3.2 – Відображення сторінок з питаннями анкети

Далі було розроблено форму входу, на якій користувач вводить деякі свої особисті дані (див. рис. 3.3).

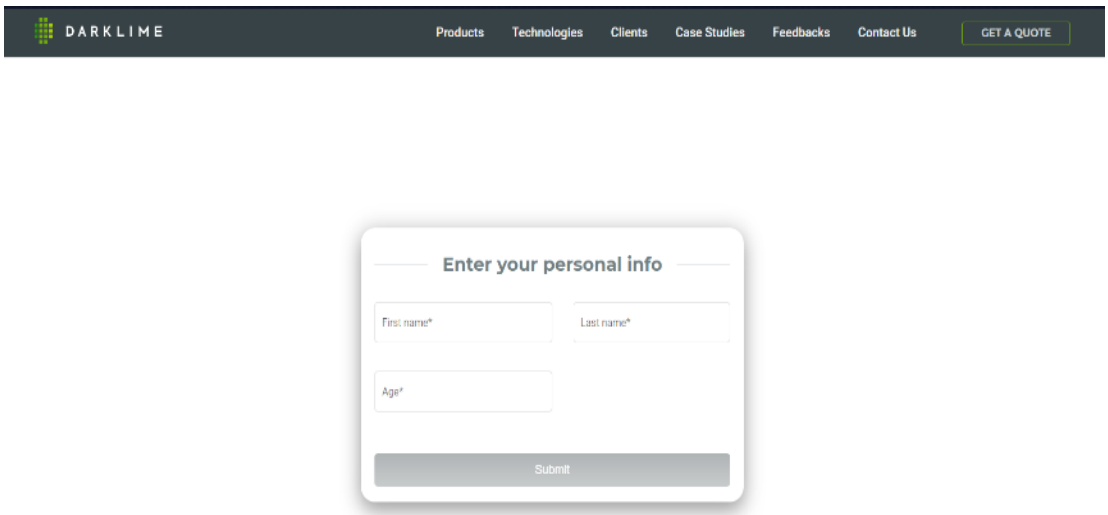


Рисунок 3.3 – Відображення сторінки з формою введення даних

Усі поля форм мають перевірку на коректність вводу.

Це зроблено для того, щоб некоректні дані не записалися в базу та сховище Store і тим самим не нанесли шкоду вже існуючим даним, а також, щоб система змогла правильно інтерпретувати дані та передати їх від клієнта до сервера чи бази даних (див. рис. 3.4-3.6).

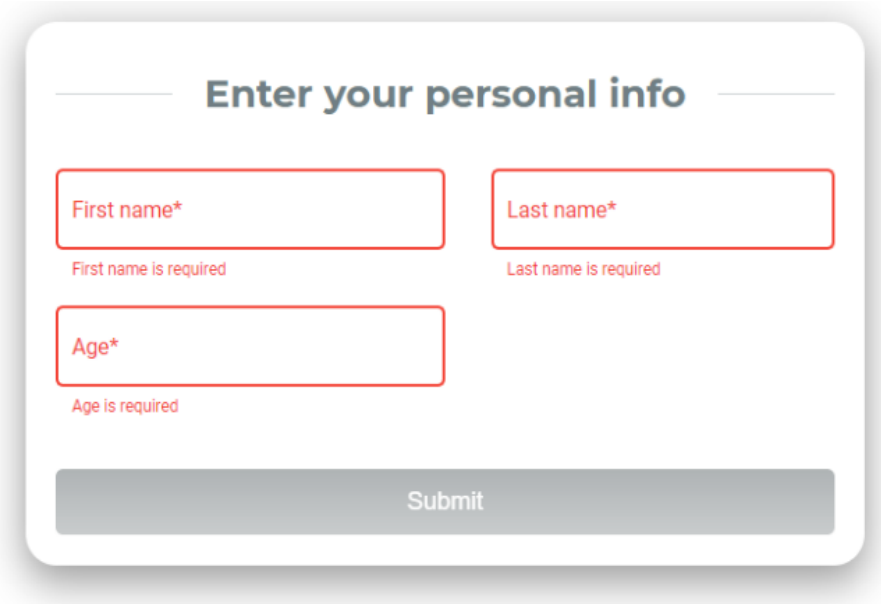


Рисунок 3.4 – Валідація форми на відсутність даних

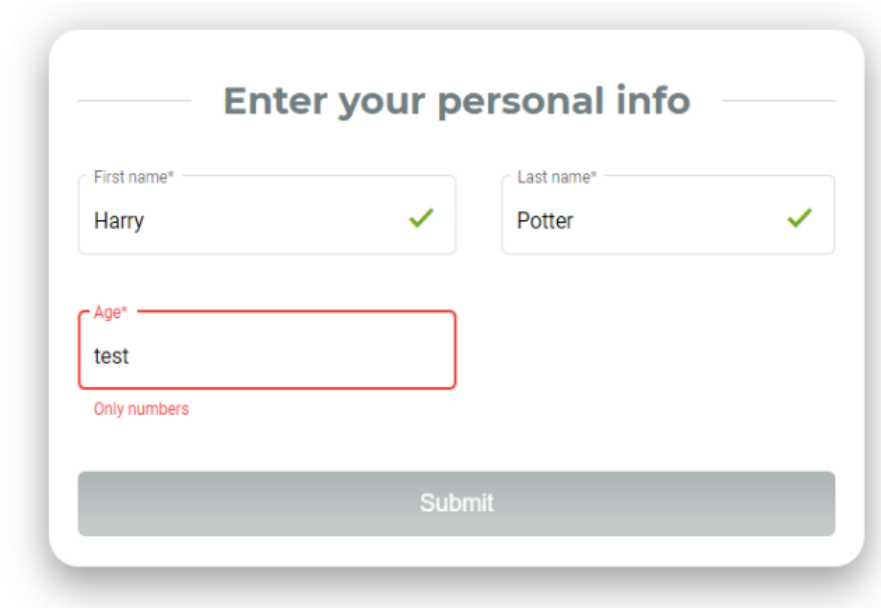


Рисунок 3.5 – Валідація форми на відповідність типів даних

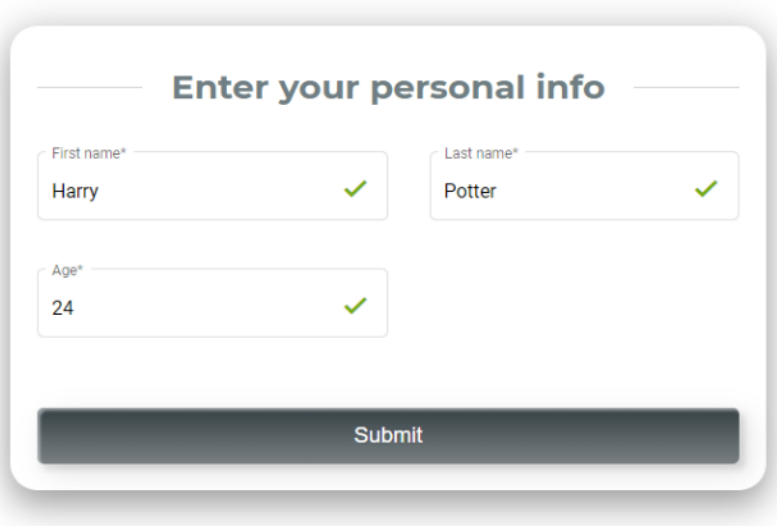


Рисунок 3.6 – Коректне заповнення форми

Для зберігання отриманих під час анкетування даних було використано Cloud Firestore базу даних.

Cloud Firestore - це гнучка масштабована NoSQL база даних для розробки мобільних та веб-додатків, а також серверів від Firebase та Google Cloud. Як і Firebase Realtime Database, Cloud Firestore підтримує синхронізацію даних між клієнтськими програмами за допомогою слухачів у реальному часі та пропонує автономну підтримку для мобільних пристроїв та Інтернету, щоб можливо було створювати адаптивні програми, які працюють незалежно від затримки в мережі або підключення до Інтернету. Досить зручна, має простий API для інтеграції у додатки та розширену, наповнену прикладами документацію.

За підключення та додавання запису до БД відповідає database-service розробленого додатку (див. рис. 3.7).

```
sendAnswersQuestionnaire(
  data: IRequestAnswersQuestionnairePartial
): Promise<IResponseAnswersQuestionnairePartial> {
  return this.firestore
    .collection('answer-list')
    .add(data).then(res => {
      return ({success: true});
    });
}
```

Рисунок 3.7 – Підключення та запис даних до бази

Основну конфігурацію бази описано у environment.ts файлі та ініціалізовано у файлі підключення модулів (див. рис. 3.8).

```
imports: [
  BrowserModule,
  AppRoutingModule,
  BrowserAnimationsModule,
  HttpClientModule,
  AngularFireModule.initializeApp(environment.firebaseConfig),
  AngularFirestoreModule.enablePersistence({synchronizeTabs: true}),
  Material,
  FlexLayoutModule,
  StoreModule.forRoot(reducers, {metaReducers}),
  StoreDevtoolsModule.instrument({maxAge: 25, logOnly: environment.production}),
  EffectsModule.forRoot([QuestionnaireEffects]),
  StoreRouterConnectingModule.forRoot(),
  !environment.production ? StoreDevtoolsModule.instrument() : [],
```

Рисунок 3.8 – Ініціалізація модуля БД

База даних складається з колекції answer-list (див. рис. 3.9), а також з переліку документів з полями, структура якого також описана в database сервісі (див. рис. 3.10).

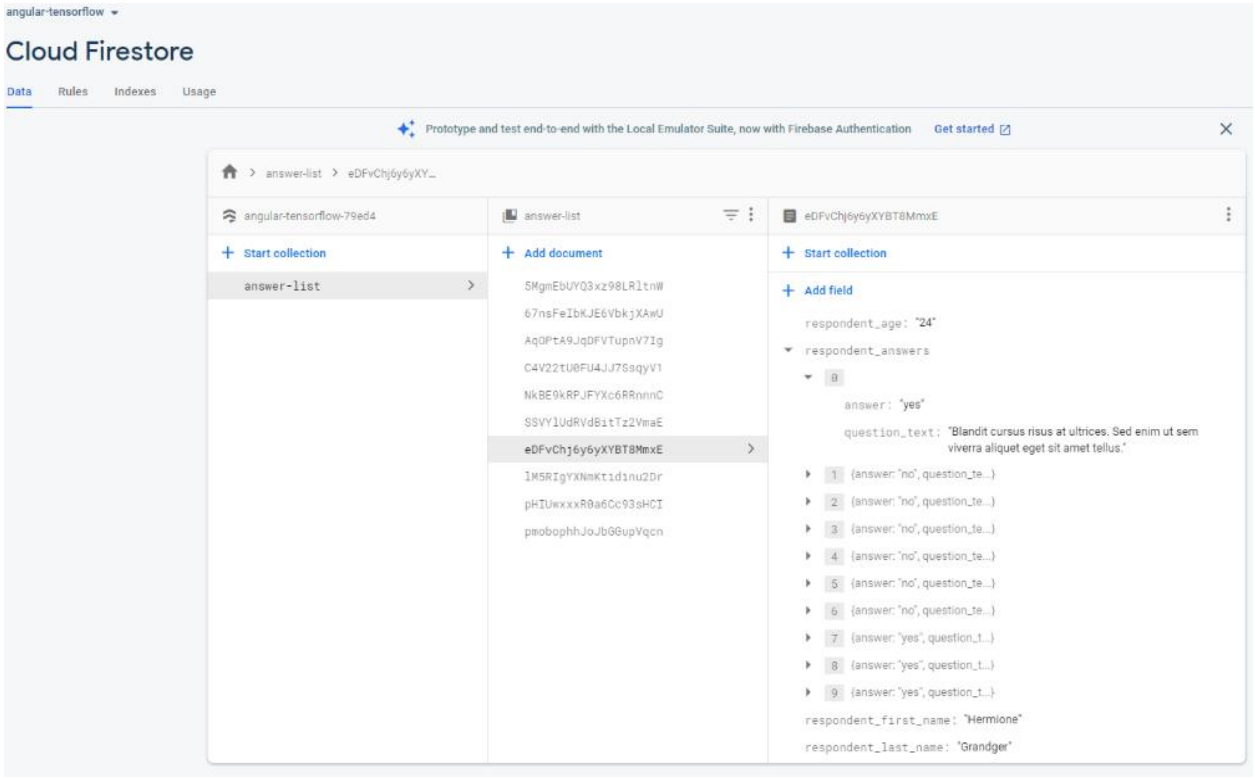


Рисунок 3.9 – База даних Cloud Firestore

```
TS database.service.ts M
src > app > core > services > TS database.service.ts > DatabaseService
1   import {Injectable} from '@angular/core';
2   import {Observable} from 'rxjs';
3   import {IQuestionnaire} from '../store/questionnaire/questionnaire.reducer';
4   import {AngularFirestore} from '@angular/fire/firestore';
5
6   interface IRequestAnswersQuestionnairePartial extends Partial<IQuestionnaire> {
7     respondent_answers: Array<IRespondentAnswer>;
8     respondent_first_name: string;
9     respondent_last_name: string;
10    respondent_age: string;
11  }
12
13  export interface IRespondentAnswer {
14    question_text: string;
15    answer: string;
16  }
17
```

Рисунок 3.10 – Структура документа БД

Інтерфейс веб-додатку було розроблено, використовуючи фреймворк Angular та бібліотеки Angular Material та Flex-Layout (див. рис. 3.11).

```
example.component.html ●
src > app > questionnaire > info > example > example.component.html > ...
    Go to component
1   <div class="example" [ngClass]="{opend: isOpened}" fxFlex="0 1 auto" fxLayout="column">
2     <div class="example__title" fxFlex fxLayout="row" fxLayoutAlign="start center">
3       <span>Examples of gestures</span>
4       <mat-icon class="example__title_rock" svgIcon="rock-fingers" aria-hidden="false"></mat-icon>
5       <div class="example__title_status" fxFlex="0 0 auto" fxLayout="row" fxLayoutAlign="center center"
6           (click)="toggleStatus()">
7         <mat-icon [@opacityAnimations] data-opened *ngIf="isOpened" matTooltip="List mode" svgIcon="example-close"
8                   aria-hidden="false"></mat-icon>
9         <mat-icon [@opacityAnimations] *ngIf="!isOpened" matTooltip="Preview mode" svgIcon="example-play"
10                  aria-hidden="false"></mat-icon>
11      </div>
12    </div>
13    <div class="example__content">
14      <app-preview [@opacityAnimations] *ngIf="isOpened" fxFlex="100%" fxLayout="column"
15                   [selectedGesture]="selectedGesture"></app-preview>
16      <app-list [@opacityAnimations] *ngIf="!isOpened" fxFlex="100%" fxLayout="column"
17                (selectedGesture)="changeSelectedGesture($event)"></app-list>
18    </div>
19  </div>
```

Рисунок 3.11 – Код шаблону компонента example

Також в процесі розробки було розроблено два quards з реалізованими методами CanActivate() (див. рис. 3.12-3.13).

TS answers.guard.ts ●

src > app > core > guards > TS answers.guard.ts > ...

```ts
import { Injectable } from '@angular/core';
import { CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot, UrlTree, Router } from '@angular/router';
import { Observable } from 'rxjs';
import { tap } from 'rxjs/operators';
import { QuestionnaireSelectors } from '../store/questionnaire/questionnaire.selectors';

@Injectable({
  providedIn: 'root'
})
export class AnswersGuard implements CanActivate {

  constructor(private questionnaireSelectors: QuestionnaireSelectors, private router: Router) {
  }

  canActivate(
    route: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree {
    const permisson =
      this.questionnaireSelectors.getAllAnsweredStatus()
        .pipe(tap(status => (!status && this.router.navigate(['/']))));
    return permisson;
  }
}
```

Рисунок 3.12 – CanActivate() метод answers.quard

TS questionnaire.guard.ts ●

src > app > core > guards > TS questionnaire.guard.ts > ...

```ts
import {Injectable} from '@angular/core';
import {CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot, UrlTree, Router} from '@angular/router';
import {Observable} from 'rxjs';
import {tap} from 'rxjs/operators';
import {QuestionnaireSelectors} from '../store/questionnaire/questionnaire.selectors';

@Injectable({
  providedIn: 'root'
})
export class QuestionnaireGuard implements CanActivate {

  constructor(private questionnaireSelectors: QuestionnaireSelectors, private router: Router) {
  }

  canActivate(
    route: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree {
    const permisson =
      this.questionnaireSelectors.getUserFilledDataStatus()
        .pipe(tap(status => (!status && this.router.navigate(['/']))));
    return permisson;
  }
}
```

Рисунок 3.13 – CanActivate() метод questionnaire.quard

Guards дозволяють обмежити навігацію певними маршрутами.

Наприклад, якщо для доступу до певного ресурсу потрібна наявність автентифікації або інших умов, залежно від яких ми можемо (чи не можемо)

надати користувачеві доступ. CanActivate представляє один із типів guards, який дозволяє керувати доступом до ресурсу під час маршрутизації.

У розробленому додатку, як вже було сказано, було створено два quards:

– questionnaire.quard (надає доступ до анкетування тільки тоді, коли користувач заповнив форму при вході в систему);

– answers.quard (надає доступ до сторінки статистики (див. рис. 3.14) тільки тоді, коли користувач відповів на усі питання анкети).

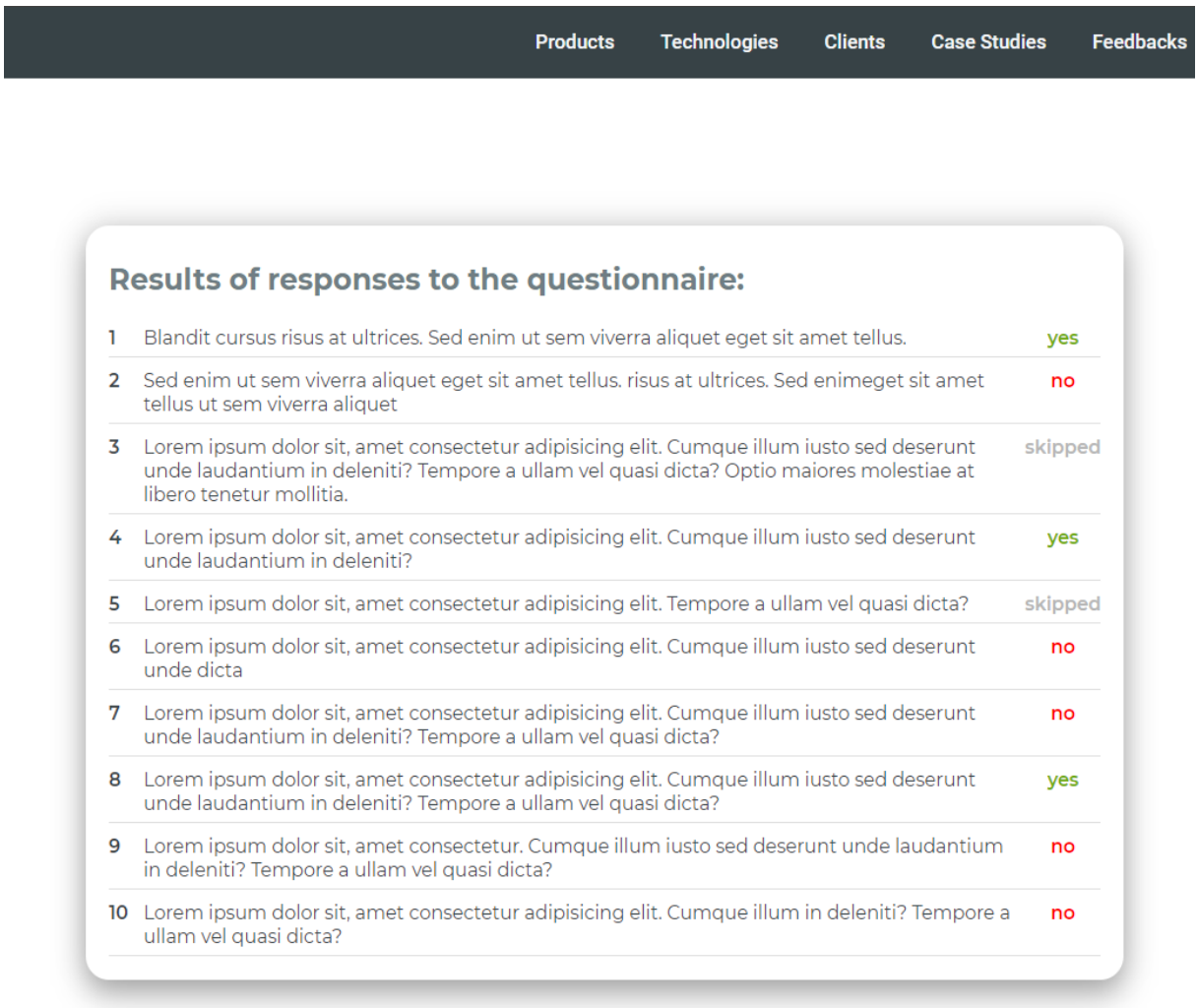


Рисунок 3.14 – Статистика відповідей користувача

В процесі розробки системи було реалізовано NGRX-store структуру для керування станом додатку (див. рис. 3.15).

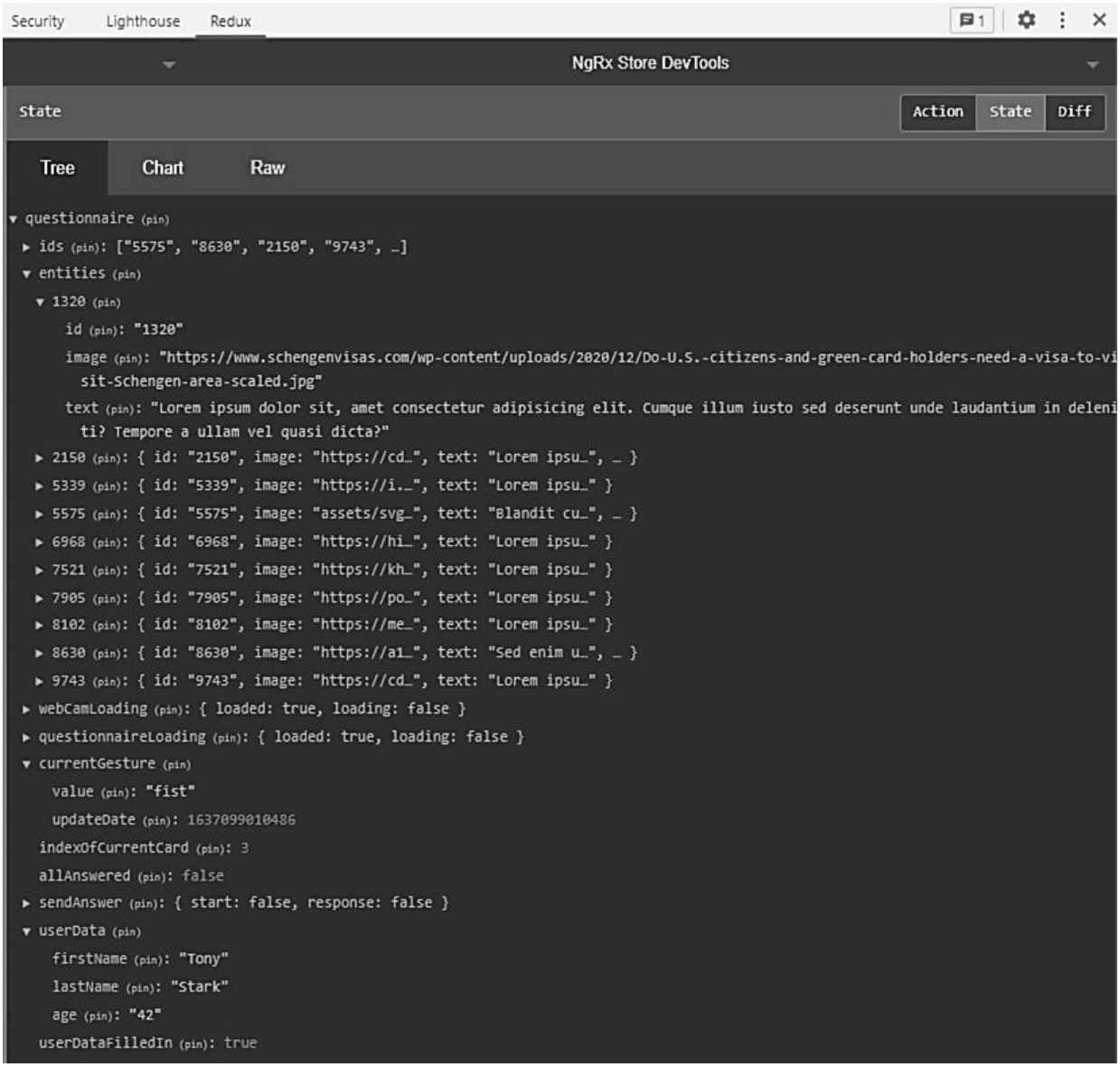


Рисунок 3.15 – NGRX-store структура системи

3.2.2 Розробка сервісу для роботи з HandPose моделлю

При роботі з додатком користувач бачить невелике віконце із зображенням з веб-камери і, коли камера розпізнає руку в кадрі, координати основних точок долоні підсвічуються червоним.

За нанесення цих самих координат на кадр відео відповідає hand-gesture-service.

Як тільки відеопотік з веб-камери надійшов у програму, запускається метод ініціалізації розпізнавача точок у сервісі (див. рис. 3.18). Усередині нього відбувається ініціалізація стартових значень ключових точок, потім

завантаження моделі та запуск методу оцінки положення руки моделі HandPose, після чого сервіс малює скелетон руки за отриманими координатами.

Наступний крок – аналіз отриманих координат ключових точок руки з метою визначення зміни поточного жесту і, якщо координати не збігаються з координатами поточного, визначення нового жесту по координатах.

Як тільки жест визначено, відбувається виклик екшену changeCurrentGesture().

Розглянемо детально структуру самого hand-gesture-service.

У сервісі оголошуємо список всіх доступних жестів, які здатна розпізнати модель (див. рис. 3.16).

Рисунок 3.16 – Перелік доступних жестів моделі HandPose

У системі, що розробляється, будуть використовуватися тільки 4 з них:

- thumbs_up;
- fist;
- wave_hand_left;
- wave_hand_right.

У сервісі оголошено змінні для жестів, потоків, останнього жесту (тобто нового жесту, який не збігається з поточним), а також гетери для відеопотоку та його налаштувань (дані гетери використовуються в компоненті, який відповідає за відображення відео з камери) (див. рис. 3.17).

hand-gesture.service.ts

```
export class HandGesture {
  private _swipe$ = new BehaviorSubject<Direction>( value: 'none');
  readonly swipe$ = this._swipe$.asObservable();

  private _gesture$ = new BehaviorSubject<Gesture>( value: 'none');
  readonly gesture$ = this._gesture$.asObservable();

  private _initiated = false;
  private _initialTimestamp = -1;
  private _stream!: MediaStream;
  private _dimensions!: Size;
  private _lastGestureTiemstamp = -1;
  private _lastGesture: any = null;
  private _emitGesture = true;

  get stream(): (MediaStream | undefined) {
    return this._stream;
  }

  private _streamSettings!: MediaTrackSettings;
  get streamSettings(): MediaTrackSettings | undefined {
    return this._streamSettings;
  }
```

Рисунок 3.17 – Перелік змінних та гетерів hand-gesture-service

Метод initialize() (див. рис. 3.18) є ключовим для роботи сервісу.

Всередині нього відбувається ініціалізація та завантаження моделі та ключових параметрів відео, а також виклик методу оцінки положення руки з моделі та підписка на observable результат (що дозволяє уникнути постійних викликів даного методу та отримувати завжди оновлені дані).

На вхід у цей метод подається canvas-елемент і відео, та, в результаті зроблених операцій в даному методі, відбувається відображення ключових точок долоні за допомогою методу drawKeypoints() сервісу hand-renderer (див. рис. 3.19) і накладання цієї відрисовки на відео, завдяки чому користувач бачить на сторінці не просто відео з веб-камери, а вже оброблене з підсвіченим червоним кольором скелетоном своєї долоні (див. рис. 3.2 а)-г)).

Крім того, в методі initialize() відбувається виклик методів обчислення жесту за координатами скелетону долоні _process() (чи є жест свайпом) (див. рис.

3.20) і _processGesture() (визначення пальцевого жесту) (див. рис. 3.21), який також визначає, чи змінився жест, і, якщо так, вказує на це системі.

hand-gesture.service.ts

```
initialize(canvas: HTMLCanvasElement, video: HTMLVideoElement): void {
  this.actionsQuestionnairesService.loadingWebcam();
  this._dimensions = [video.width, video.height];
  navigator.mediaDevices
    .getUserMedia( constraints: {video: true}) Promise<MediaStream>
    .then((stream : MediaStream ) => {
      this._stream = stream;
      this._streamSettings = stream.getVideoTracks()[0].getSettings();
      return handpose.load();
    }) Promise<HandPose>
    .then((model : HandPose ) => {
      const context: CanvasRenderingContext2D = canvas.getContext( contextId: '2d');
      context.clearRect( x: 0,  y: 0, video.width, video.height);
      context.strokeStyle = 'red';
      context.fillStyle = 'red';
      context.translate(canvas.width,  y: 0);
      context.scale( x: -1,  y: 1); // mode mirrored

      // the image is rendered at the frequency of the webcam
      this.runAnimation( callback: (data : IRunAnimationCallbackStart ) => {
        model.estimateHands(video).then((predictions : AnnotatedPrediction[] ) => {
          data.startAnimation( data: {
            render: () => { // Render
              context.drawImage(
                video,
                sx: 0,
                sy: 0,
                video.width,
                video.height,
                dx: 0,
                dy: 0,
                canvas.width,
                canvas.height
              );
              if (predictions && predictions[0]) {
                drawKeypoints(context, predictions[0].landmarks);
                this._process(predictions[0].boundingBox);
                this._processGesture(predictions[0].landmarks);
              }
            }
          });
        });
      }, this._streamSettings.frameRate);
    }) Promise<void>
    .catch((err) => {
      this.actionsQuestionnairesService.errorLoadedWebcam(err);
      console.error(err);
    });
}
```

Рисунок 3.18 – Метод ініціалізації вихідного відеопотоку

hand-renderer.ts

```ts
const fingerLookupIndices: any = {
  thumb: [0, 1, 2, 3, 4],
  indexFinger: [0, 5, 6, 7, 8],
  middleFinger: [0, 9, 10, 11, 12],
  ringFinger: [0, 13, 14, 15, 16],
  pinky: [0, 17, 18, 19, 20],
};

export const drawKeypoints = (ctx: any, keypoints: any) => {
  const keypointsArray = keypoints;

  // tslint:disable-next-line:prefer-for-of
  for (let i = 0; i < keypointsArray.length; i++) {
    const y = keypointsArray[i][0];
    const x = keypointsArray[i][1];
    drawPoint(ctx,  y: x - 2,  x: y - 2,  r: 3);
  }

  const fingers = Object.keys(fingerLookupIndices);
  // tslint:disable-next-line: prefer-for-of
  for (let i = 0; i < fingers.length; i++) {
    const finger = fingers[i];
    const points = fingerLookupIndices[finger].map((idx: any) => keypoints[idx]);
    drawPath(ctx, points,  closePath: false);
  }
};

function drawPoint(ctx: any, y: number, x: number, r: number): void {
  ctx.beginPath();
  ctx.arc(x, y, r,  startAngle: 0,  endAngle: 2 * Math.PI);
  ctx.fill();
}

function drawPath(ctx: any, points: any, closePath: any): void {
  const region = new Path2D();
  region.moveTo(points[0][0], points[0][1]);
  for (let i = 1; i < points.length; i++) {
    const point = points[i];
    region.lineTo(point[0], point[1]);
  }

  if (closePath) {
    region.closePath();
  }
  ctx.stroke(region);
```

Рисунок 3.19 – Метод відрисовки скелетону долоні

```
private _process(rect: Rect): void {
  const middle = this._getMiddle(rect);
  if (this._aroundCenter(middle)) {
    this._initialTimestamp = Date.now();
    this._initiated = true;
    return;
  }
  if (!this._initiated) {
    return;
  }
  if (
    this._inRegion( start: 0,  end: 0.1, middle) &&
    this._toSeconds( ms: Date.now() - this._initialTimestamp) < 2
  ) {
    this._swipe$.next( value: 'right');
    this.actionsQuestionnairesService.changeGesture(GestureMap.wave_hand_right);
    this._initiated = false;
    return;
  }
  if (
    this._inRegion( start: 0.9,  end: 1, middle) &&
    this._toSeconds( ms: Date.now() - this._initialTimestamp) < 2
  ) {
    this._swipe$.next( value: 'left');
    this.actionsQuestionnairesService.changeGesture(GestureMap.wave_hand_left);
    this._initiated = false;
    return;
  }
}
```

Рисунок 3.20 – Метод визначення жесту свайп за координатами точок долоні

```
private _processGesture(landmarks: Coords3D): void {
  const {gestures} = GE.estimate(landmarks, 7.5) || [];
  const gesturesArray = (Array.prototype.slice.call(gestures,  argArray: 0) as Array<IGestures>);
  let gesture = null;
  if (gesturesArray.length > 0) {
    const gesturesNames = gesturesArray.map(x => x.name);
    if (!gesture) {
      gesture = gesturesNames.find(x => x === 'thumbs_up');
    }
    if (!gesture) {
      gesture = gesturesNames.find(x => x === 'four_fingers');
    }
    if (!gesture) {
      gesture = gesturesNames.find(x => x === 'three_fingers');
    }
    if (!gesture) {
      gesture = gesturesNames.find(x => x === 'victory');
    }
    if (!gesture) {
      gesture = gesturesNames.find(x => x === 'one_finger');
    }
    if (!gesture) {
      gesture = gesturesNames.find(x => x === 'fist');
    }
  }
  if (this._lastGesture !== gesture) {
    this._lastGesture = gesture;
    this._lastGestureTiemstamp = Date.now();
    this._emitGesture = true;
  } else {
    if (
      this._emitGesture &&
      this._toSeconds( ms: Date.now() - this._lastGestureTiemstamp) > 1
    ) {
      this._gesture$.next(GestureMap[this._lastGesture] as Gesture);
      this.actionsQuestionnairesService.changeGesture(GestureMap[this._lastGesture]);
      this._emitGesture = false;
    }
  }
}
```

Рисунок 3.21 – Метод визначення жесту пальців за координатами долоні

ВИСНОВКИ

В результаті роботи реалізовано веб-додаток, яким користувач може керувати за допомогою жестів рук із застосуванням наступних технологій:

- HandPose модель бібліотеки tensorflow.js;
- Cloud Firestore database;
- Angular 12 Framework для реалізації front-end;
- Flex-Layout та Angular Material для реалізації інтерфейсу користувача.

Розроблена система на базі Angular та tensorflow.js реалізує наступні базові функції:

- відповідь на питання жестом;
- перехід до наступного/попереднього питання жестом;
- перегляд статистики відповідей;

Розроблена інформаційна система має наступні переваги:

- кросплатформна, залежить від web-оглядача;
- використані сучасні web-технології, що робить розробку актуальною сьогодні, гнучкою в модифікації та легкою в супроводі;
- не потребує потужних клієнтських робочих станцій;
- не прив’язана до робочого місця.

Код розробленої системи можна переглянути за посиланням: https://bitbucket.org/alinaTdev/ngtensorflow/src/master/.

ПЕРЕЛІК ПОСИЛАНЬ

1. Куракин А. В. Распознавание жестов ладони в реальном времени на основе плоских и пространственных скелетных моделей. *Информатика и её применение.* 2012. № 1. С. 114–121. URL : http://www.mathnet.ru/links /0abe0e65feed2e03b9739232035ea962/ia192.pdf.

2. Павлов І. Технологии машинного обучения: примеры современных тенденций. Дата оновлення: 14 вересня, 2020. URL : https://habr.com/ru/company /kauri_iot/blog/518974/ (дата звернення: 16.08.2021).

3. Дударов С. Нейронные сети на основе радиально-симметричных функций. URL : https://neuronus.com/theory/nn/954-nejronnye-seti-na-osnove-radialno-simmetrichnykh-funktsij.html (дата звернення: 20.06.2021).

4. Мусиенко Ю. Главные тренды и технологии машинного обучения в 2022. Дата оновлення: 6 жовтня 2021. URL : https://merehead.com/ru/blog/top-machine-learning-trends-and-technologies-2022/ (дата звернення: 15.10.2021).

5. Машинное обучение. URL : https://www.calltouch.ru/glossary /mashinnoe-obuchenie/ (дата звернення: 15.10.2021).

6. Хайбуллин А. 5 лучших библиотек машинного обучения. Дата оновлення: 15 вересня 2019. URL : https://medium.com/nuances-of-programming /5-лучших-библиотек-машинного-обучения-f8b6ddf50945 (дата звернення: 11.10.2021).

7. Топ 8 библиотек Python для машинного обучения и искусственного интеллекта. Дата оновлення: 4 травня 2020. URL : https://pythonist.ru/top-8-bibliotek-python-dlya-mashinnogo-obucheniya-i-iskusstvennogo-intellekta/ (дата звернення: 11.10.2021).

8. Діаграма прецедентів. URL : http://fitm.nusta.edu.ua/mediawiki /index.php (дата звернення: 13.12.2019).

9. Основи UML – проектування розподілених систем. URL : http://moodle.ipo.kpi.ua/moodle/mod/resource/view.php?inpopup=true&id=44416 (дата звернення: 13.12.2019).

10. Сиротенко М. Применение нейросетей в распознавании изображений. URL : https://habr.com/ru/post/74326/ (дата звернення: 15.12.2019).

11. Проектування програмного забезпечення. URL : https://d-learn.pnu.edu.ua/data/users/9633/L1.pdf (дата звернення: 15.12.2019).

12. Нейронные сети: распознавание образов и изображений с помощью ИИ. URL : https://center2m.ru/ai-recognition (дата звернення: 17.08.2021).

13. Hand Gesture Recognition Using Temporal Convolutions and Attention Mechanism. URL : https://arxiv.org/abs/2110.08717 (дата звернення: 17.08.2021).

14. Oh Y., Toussaint M., Mainprice J. A System for Traded Control Teleoperation of Manipulation Tasks using Intent Prediction from Hand Gestures. URL : https://arxiv.org/abs/2107.01829 (дата звернення: 15.08.2021).

15. Ibrahim A., El-Refai A., Ahmed S., Aboul-Ela M., Hesham M. Pervasive Hand Gesture Recognition for Smartphones using Non-audible Sound and Deep Learning. URL : https://arxiv.org/abs/2108.02148 (дата звернення: 17.08.2021).

16. О библиотеке NGRX. URL : https://angdev.ru/ngrx/about/ (дата звернення: 19.08.2021).

17. Sung G., Sokal K., Uboweja E., Bazarevsky V., Bazavan G. On-device Real-time Hand Gesture Recognition. URL : https://arxiv.org/abs/2111.00038 (дата звернення: 19.08.2021).

18. What is Angular? URL : https://angular.io/guide/what-is-angular (дата звернення: 15.08.2021).

19. Bakshi R. Hand Pose Classification Based on Neural Networks. URL : https://arxiv.org/abs/2108.04529 (дата звернення: 19.08.2021).

20. Nikivay. Angular: понятное введение в NGRX. URL : https://habr.com/ru/post/489674/ (дата звернення: 16.08.2021).

21. The RxJS library. URL : https://angular.io/guide/rx-library (дата звернення: 21.08.2021).

22. Introduction to RxJS. URL : https://rxjs.dev/guide/overview (дата звернення: 21.08.2021).

23. Ращенко Ю. Разработка модифицированных алгоритмов обучения для нейронной сети адаптивной резонансной теории. URL : https://core.ac.uk/download/pdf/217174152.pdf (дата звернення: 17.08.2021).

ДОДАТОК А

Перелік навчених моделей бібліотеки tensorflow.js

Таблиця А.1 – Навчені моделі бібліотеки tensorflow.js

Тип моделі	Назва моделі	Опис моделі
Зображення	MobileNet	Класифікація зображень за допомогою міток з ImageNet database
	HandPose	Виявлення пози руки у браузері в реальному часі
	Pose	Визначення пози людини у браузері в реальному часі
	Coco SSD	Локалізація та ідентифікація кількох об'єктів на одному зображенні
	BodyPix	Сегментація людей та частин тіла в режимі реального часу у браузері
	BlazeFace	Виявлення облич у браузері в режимі реального часу
	Face Landmark Detection	Виявлення тривимірних орієнтирів обличчя у реальному часі для визначення його приблизної геометричної поверхні
Аудіо	Speech Commands	Класифікація 1-секундних аудіофрагментів із speech commands dataset
Текст	Universal Sentence Encoder	Кодування тексту в 512-мірне вбудовування, яке буде використовуватися як вхідні дані для завдань обробки природньої мови, таких як класифікація тональності та текстова подібність