МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

ГУРНЯК Владислав Ігорович

Програмна система виявлення зловживань під час онлайн-тестувань засобами машинного навчання / Software System for Detecting Abuses During Online Testing Using Machine Learning Tools

спеціальність: 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

Кваліфікаційна робота

Виконав студент групи КН-42
В.І. Гурняк

_____

Науковий керівник:
к.т.н., доцент Д.І. Загородня

_____

Кваліфікаційну роботу допущено до захисту
«___»_____ 20__ р.

Завідувач кафедри
_____ М.П. Комар

Тернопіль – 2024

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
Спеціальність 122 – Комп'ютерні науки
Освітньо-професійна програма – Комп'ютерні науки

«ЗАТВЕРДЖУЮ»
Завідувач кафедри ІОСУ
_____ Мирослав КОМАР
«____» _____20___р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Гурняку Владислав Ігорович

1. Тема кваліфікаційної роботи (укр. / англ. мовами) Програмна система виявлення зловживань під час онлайн-тестувань засобами машинного навчання / Software System for Detecting Abuses During Online Testing Using Machine Learning Tools
керівник проекту к.т.н., доцент Д.І. Загородня
затверджені наказом по університету від 12 грудня 2023 р. №753.

2. Строк подання студентом закінченої кваліфікаційної роботи 15 травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.

4. Основні питання, які потрібно розробити:
- провести аналіз предметної області;
- провести огляд та аналіз існуючих аналогів;
- визначити основні функціональні вимоги;
- розробити основні сценарії роботи системи;
- створити зрозумілий дизайн інтерфейсу;
- розробити функціональну веб-базовану систему.

5. Перелік графічного матеріалу в роботі:
- загальна структура системи;
- алгоритм входу в систему;
- алгоритм участі в тестуванні;
- Use Case діаграма.

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
Н. контроль	к.т.н., доцент Д.І. Загородня		

7. Дата видачі завдання 12 грудня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Затвердження теми кваліфікаційної роботи, ознайомлення з літературними джерелами та складання плану роботи.	до 01.01. 2024 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.03. 2024 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 01.04.2024 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 01.05. 2024 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 15.05.2024 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів.	до 20.05.2024 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту у системі «Unicheck».	до 10.06.2024 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 14.06.2024 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії.	до 14.06. 2024 р.	

Студент _____ Гурняк В. І.
(підпис)

Керівник проекту _____ Загородня Д.І.
(підпис)

АНОТАЦІЯ

Кваліфікаційна робота на тему «Програмна система виявлення зловживань під час онлайн-тестувань засобами машинного навчання» на здобуття освітнього ступеня «бакалавр» зі спеціальності 122 «Комп'ютерні науки» освітньої програми «Комп'ютерні науки» написана обсягом в 76 сторінок і містить 22 ілюстрації, одну таблицю, два додатки та 40 використаних джерел.

Метою даної кваліфікаційної роботи є розробка програмної системи, яка дозволяє виявляти зловживання під час онлайн-тестувань з використанням методів машинного навчання.

Предметом дослідження є методи і технології машинного навчання, які застосовуються для виявлення зловживань під час онлайн-тестувань, зокрема, аналіз зображень, відео та звуку з метою моніторингу поведінки студентів під час тестування.

Об'єктом дослідження є процеси онлайн-тестувань та потенційні зловживання, які можуть виникати під час цих тестувань.

Запропонована програмна система дозволяє покращити процес оцінки знань, забезпечуючи справедливість і чесність оцінювання студентів. Система використовує методи обробки відео та звуку, аналізуючи виявлені обличчя, рухи очей, рухи рота та наявність сторонніх осіб, що може свідчити про зловживання.

Тестування програмної системи довело її спроможність виявляти різні види зловживань, таких як відвертання погляду, використання додаткових осіб, спроби спілкування під час тестування. Система продемонструвала стабільну роботу, продуктивність та надійність.

Ключові слова: ОНЛАЙН-ТЕСТУВАННЯ, АКАДЕМІЧНА НЕДОБРОЧЕСНІСТЬ, МАШИННЕ НАВЧАННЯ, МОНІТОРИНГ, ДИСТАНЦІЙНА ОСВІТА.

ABSTRACT

The qualification work titled "Software System for Detecting Abuses During Online Testing Using Machine Learning Tools" for obtaining a bachelor's degree in the specialty 122 "Computer Science" of the educational program "Computer Science" is written in the volume of 76 pages and contains 22 illustrations, one table, two appendices, and 40 references.

The purpose of this qualification work is to develop a software system that allows detecting abuses during online tests using machine learning methods.

The subject of the study is the methods and technologies of machine learning used to detect abuses during online testing, particularly the analysis of images, video, and sound to monitor student behavior during testing.

The object of the study is the processes of online testing and potential abuses that may occur during these tests.

The proposed software system improves the knowledge assessment process, ensuring fairness and integrity in student evaluations. The system uses methods of video and sound processing, analyzing detected faces, eye movements, mouth movements, and the presence of unauthorized persons, which may indicate abuses.

Testing of the software system proved its ability to detect various types of abuses, such as looking away, using additional persons, and attempts to communicate during testing. The system demonstrated stable performance, efficiency, and reliability.

Keywords: ONLINE TESTING, ACADEMIC DISHONESTY, MACHINE LEARNING, MONITORING, DISTANCE EDUCATION.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – База Даних

ШІ – Штучний Інтелект

API – Application Programming Interface

CSS – Cascading Style Sheets

DFD – Data-Flow Diagram

HTML – HyperText Markup Language

SEB – Safe Exam Browser

UML – Unified Modeling Language

ВСТУП

Актуальність розробки системи виявлення зловживань під час онлайн-тестувань засобами машинного навчання обумовлена рядом важливих факторів, основним з яких можна вважати примусовий перехід навчальних закладів по всьому світу на дистанційне навчання в зв'язку з пандемією Covid-19, що викликав різке збільшення кількості онлайн-іспитів і тестувань, що в свою чергу призвело до значного зростання випадків академічної недоброчесності, таких як списування, використання сторонніх матеріалів та допомоги соронніх осіб. Традиційні методи моніторингу виявилися неефективними в онлайн-середовищі, оскільки студенти мали можливість використовувати численні технічні засоби для шахрайства. Відповідно, почало з'являтись багато комерційних продуктів для онлайн-прокторингу, проте вони є дорогими і не завжди забезпечують належний рівень безпеки та справедливості. Це підкреслює потребу у розробці більш доступних і ефективних рішень, які можуть бути інтегровані в освітні процеси різних навчальних закладів.

Розвиток технологій штучного інтелекту і машинного навчання відкрив нові можливості для створення інноваційних систем моніторингу. Використання методів аналізу зображень, відео та звуку дозволяє виявляти підозрілу поведінку студентів під час тестування, наприклад, відведення погляду, використання додаткових пристроїв або спілкування з іншими особами. Тому розробка такої системи дозволить покращити процес оцінки знань студентів, забезпечивши справедливість і чесність оцінювання. Це сприятиме підвищенню рівня академічної доброчесності, що є важливим аспектом у забезпеченні якісної освіти.

Метою даної кваліфікаційної роботи є розробка програмної системи, яка дозволяє виявляти зловживання під час онлайн-тестувань з використанням методів машинного навчання. Ця система покликана підвищити рівень академічної чесності та забезпечити справедливість у процесі оцінювання знань студентів під час онлайн-тестувань. Для досягнення цієї мети необхідно виконати наступні завдання:

- провести аналіз предметної області;
- провести огляд та аналіз існуючих аналогів;
- визначити основні функціональні вимоги;
- розробити основні сценарії роботи системи;
- створити зрозумілий дизайн інтерфейсу;
- розробити функціональну веб-базовану систему.

Об'єктом дослідження є процеси онлайн-тестувань та потенційні зловживання, які можуть виникати під час цих тестувань. Особливу увагу приділено розробці та впровадженню системи моніторингу, яка аналізує дані з відеокамери та мікрофону для виявлення зловживань під час онлайн-тестувань.

Предметом дослідження є методи і технології машинного навчання, які застосовуються для виявлення зловживань під час онлайн-тестувань. Зокрема, це аналіз зображень, відео та звуку з метою моніторингу поведінки студентів під час тестування.

Методи дослідження для розробки системи виявлення зловживань під час онлайн-тестувань засобами машинного навчання включають кілька важливих етапів. Спочатку було проведено детальний аналіз предметної області, щоб зрозуміти основні проблеми та виклики, пов'язані з академічною недоброчесністю під час онлайн-тестувань. Потім здійснено огляд та аналіз існуючих аналогів – систем онлайн-прокторингу, які вже використовуються в навчальних закладах, з метою визначення їхніх функціональних можливостей та обмежень. Основними методами дослідження стали методи машинного навчання, зокрема алгоритми обробки зображень, відео та аудіо для моніторингу поведінки студентів. Було застосовано технології розпізнавання облич, трекінгу рухів очей, аналізу рухів голови та губ, а також розпізнавання голосу. Для тестування та валідації запропонованої системи використовувались моделювання та симуляції.

Практичне значення запропонованої в кваліфікаційній роботі системи полягає в тому, що вона дасть змогу значно підвищити рівень академічної доброчесності, зменшуючи кількість випадків шахрайства під час онлайн-тестувань. Це забезпечить справедливе оцінювання знань студентів, що є важливим

для підтримання якості освіти. Також дана система система автоматизує сам процес процес моніторингу та виявлення зловживань, що зменшить навантаження на викладачів та прокторів, роблячи процес контролю більш ефективним та зручним. Крім того, використання методів машинного навчання дозволяє створити більш доступні рішення в порівнянні з дорогими комерційними продуктами, що сприяє впровадженню таких технологій у ширший спектр навчальних закладів. Забезпечення чесності та надійності онлайн-тестувань підвищить довіру до дистанційних форм навчання, сприяючи їх подальшому розвитку. Водночас розробка і впровадження такої системи надає цінний практичний досвід у сфері застосування штучного інтелекту та машинного навчання в освітніх проєктах.

Структура і розмір дослідження. Кваліфікаційна робота складається із вступу, трьох розділів, висновків та списку використаних джерел. Загальний обсяг роботи складає 76 сторінок і містить 22 рисунки, одну таблицю, два додатки та 40 використаних джерел.

В першому розділі описано аналіз систем виявлення зловживань під час онлайн-тестувань, включаючи опис предметної області, огляд та аналіз існуючих аналогів, а також постановку задачі дослідження. Другий розділ присвячений алгоритмічному та інформаційному забезпеченню системи, де детально розглянуто загальну структуру проектованої системи, алгоритмічне забезпечення, а також інформаційне забезпечення, що включає використання бібліотек для розпізнавання облич і моніторингу звуку. У третьому розділі розглянуто програмно-технологічне забезпечення системи, включаючи реалізацію програмного забезпечення, інтерфейс користувача та тестування програмного забезпечення.

Апробацію дослідження проведено на міжнародній науково-практичній конференції , м. Ізмаїл, 18 травня 2024 р. “Перспективні напрямки розвитку економіки, обліку, управління та права: теорія і практика” в секції “ Математичні методи, моделі та інформаційні технології в економіці” де була представлена доповідь на тему “Програмна система виявлення зловживань під час онлайн-тестувань засобами машинного навчання”.

1. АНАЛІЗ СИСТЕМ ВИЯВЛЕННЯ ЗЛОВЖИВАНЬ ПІД ЧАС ОНЛАЙН-ТЕСТУВАНЬ

1.1. Опис предметної області

Під час триваючої пандемії коронавірусної хвороби 2019 (Covid-19) вищі навчальні заклади по всьому світу перевели навчання в онлайн-режим, оскільки очне навчання було заборонено через швидке поширення вірусу. Як відомо з ЮНЕСКО (2020), близько 138 країн і 1,37 мільярда студентів по всьому світу навчалися онлайн вдома [1-4].

Проте з відходом пандемії онлайн-курси та іспити стали все більш поширеною практикою, що дало студентам можливість відвідувати курси та складати іспити онлайн. Освітні програми також еволюціонували, щоб забезпечити підтримку навчального процесу для студентів, що не мають змоги бути фізично присутніми в аудиторії. Також онлайн-навчальні курси почали широко впроваджуватись приватними установами, що стало додатковим стимулом для розробки спеціалізованих навчальних програм для різних навчальних платформ, що дозволило знизити витрати на проходження навчання [5].

У більшості навчальних закладів весняний семестр 2020 року пройшов в онлайн режимі, використовуючи системи дистанційної освіти, такі як Moodle, і для роботи з відеоконференціями використовувалась платформа Zoom, котра крім свого основного призначення, ще й виступала як простий метод перевірки фізичної присутності на онлайн-іспиті [5].

Однак, як спосіб контролю, засоби для телеконференцій не завжди приносять позитивні результати, оскільки студенти можуть вузько спрямовувати свої веб-камери, щоб було важко побачити, що вони читають з монітора або що вони пишуть (чи переглядають) з інших пристроїв, друкованих документів чи книг. Природно, що такий спосіб створює численні можливості для списування у студентів. Студенти можуть таємно робити знімки екрану з екзаменаційними анкетами та відповідями, та поширювати їх своїм одногрупникам використовуючи програми обміну онлайн-повідомленнями, смартфони, або усно ділитися

відповідями під час іспиту. Відповідно, багато викладачів відзначають підвищення оцінок під час онлайн-іспитів і часті випадки підозрілих результатів, оскільки група студентів надавала надзвичайно схожі відповіді на екзаменаційні запитання.

Така ситуація призвела до збільшення скарг студентів (особливо тих, хто чесно складав іспити), які хвилювалися, що оцінка їхнього курсу стане несправедливою: оскільки в університеті використовується відносна система оцінювання, «обман» студентів може безпосередньо зашкодити оцінкам «чесних» студентів.

Одним з методів протидії стало запровадження контролю за онлайн-іспитом з використанням безпечного браузера (SEB, Safe Exam Browser) [6]. Студенти повинні були отримати доступ до свого курсу через безпечний браузер, що перешкоджало їм отримати доступ до інших веб-сайтів, програмного забезпечення для обміну повідомленнями або функцій захоплення екрана на своїх комп'ютерах. Водночас студентів попросили увійти на зустріч у Zoom за допомогою своїх смартфонів. У той час як вони використовували свій комп'ютер для виконання тестів, вони використовували смартфони для показу своїх рук та монітору комп'ютера, які викладач, вчитель чи асистент учителя могли переглядати. Усі екзаменаційні сесії записувалися, щоб можна було переглядати записи, коли студенти помічали або повідомляли про підозрілі академічні проступки.

Крім того, в університетах часто застосовують «Кодекс честі студентів», котрий вони підписують на початку кожного семестру, тим самим студенти обіцяють, не списувати на іспитах і не вчиняти актів академічної нечесності.

Якщо розглядати причини академічної недоброчесності, то можна виділити декілька основних, таких як почуття незацікавленості або неготовності до теми, відчуття, що поведінка шахрайства є поширеною, і уявлення про те, що така поведінка не буде покарана, якщо її виявлять [7]. Ці причини описані науковцями як трикутник шахрайства, де виділено три основні фактори, що спонукають до шахрайства: стимул, можливість і раціоналізація [8].

Стимул стосується ролі внутрішнього та зовнішнього тиску як мотивації до шахрайства. Численні дослідження показали, що зовнішній тиск, викликаний

очікуваннями інших, а також важке навантаження, пов’язане зі складною навчальною програмою, можуть спричинити шахрайство [9]. Інші дослідники [10] виявили, що контексти навчання, які акцентують увагу на навчанні та применшують важливість оцінок, як правило, стримують списування. Подібним чином, дослідження також показали, що списування зменшується в навчальному середовищі, де студенти мотивовані до досягнення знань/майстерності, а не просто до демонстрації високих результатів у навчанні [11].

Другий елемент трикутника шахрайства - можливість, стосується здатності брати участь у нечесній поведінці через відсутність відповідних механізмів для запобігання шахрайству. Дослідження показали, що навчальне середовище, де відсутні чіткі та достатні правила та покарання щодо списування, може призвести до академічної нечесності [12]. Велика кількість досліджень відзначає переваги систем кодексу честі та зусиль, спрямованих на навчання студентів важливості академічної доброчесності [13].

Останній елемент трикутника шахрайства - раціоналізація, відноситься до уявлення про те, що студент вважає нечесну поведінку такою, що не порушує його/її власне почуття етики. Відповідно до цього, роль особистісних характеристик була ще одним важливим напрямком дослідження академічної нечесності. Дослідники висунули теорію про те, як особистісні риси Великої п’ятірки (нейротизм, екстраверсія, відкритість, приємність і сумлінність) можуть пом’якшити академічну нечесність і здатність учня раціоналізувати таку поведінку [14].

1.2. Огляд і аналіз існуючих аналогів

Програмне забезпечення для виявлення шахрайства під час онлайн-іспитів, набуває все більшої популярності, особливо після того, як пандемія Covid-19 спричинила перехід на онлайн-режим навчання.

Існують програмні засоби онлайн-нагляду, що працюють обмежуючи кількість програм, які комп’ютер може запускати протягом іспитового періоду [6], та інше програмне забезпечення для контролю, що використовує штучний інтелект та алгоритми машинного навчання для виявлення підозрілої поведінки під час вивчення записів онлайн-іспитів котрі можна назвати OLP [15].

Програми онлайн-нагляду (OLP), також відомі як дистанційний нагляд, використовують цифрові методи моніторингу та контролю за діяльністю студентів під час іспитів за допомогою веб-камер і Інтернету. Вони запобігають та виявляють можливі випадки неправомірної поведінки під час тестування. OLP записують дані через онлайн-сервіс, що дозволяє зберігати та переглядати поведінку студентів під час іспиту. Крім того, ці програми часто включають функції для автентифікації особи екзаменованого, щоб підтвердити, що це дійсно та людина, яка складає іспит. OLP можна класифікувати за кількома категоріями.

Перша категорія включає програми нагляду в реальному часі. У таких програмах проктор знаходиться у віддаленому місці і контролює діяльність випробуваного у режимі реального часу. Це дозволяє забезпечити автентифікацію учасника тестування та запобігти будь-яким формам недобросовісних дій. Якщо проктор помітить недбалість або порушення, він може перервати іспит.

Друга категорія включає записані програми прокторінгу. Ці програми не вимагають безпосереднього нагляду під час іспиту. Замість цього, поведінка студентів фіксується на відео, яке потім переглядають викладачі або інші відповідальні особи. Вони аналізують запис для виявлення можливих порушень чи підозрілих дій.

Третя категорія – автоматизовані програми прокторінгу. Вони є найсучаснішими і використовують розширені функції аналізу аудіо-відео. Під час тесту поведінка досліджуваних записується, а потім автоматизована система переглядає цей запис, щоб виявити будь-які аномальні чи незаконні дії. Ці системи здатні забезпечити високий рівень нагляду без залучення людини.

Нижче розглянуто системи, що охоплюють різні підходи до прокторингу, мають різний набір функцій і підходять для різних типів навчальних закладів і екзаменів.

1. ProctorU - Це одна з найбільш відомих систем прокторингу, що пропонує п'ять функцій ідентифікації, включаючи ідентифікацію документів, розпізнавання обличчя, автоматичну автентифікацію та аналіз натискання клавіш (рисунок 1.1).

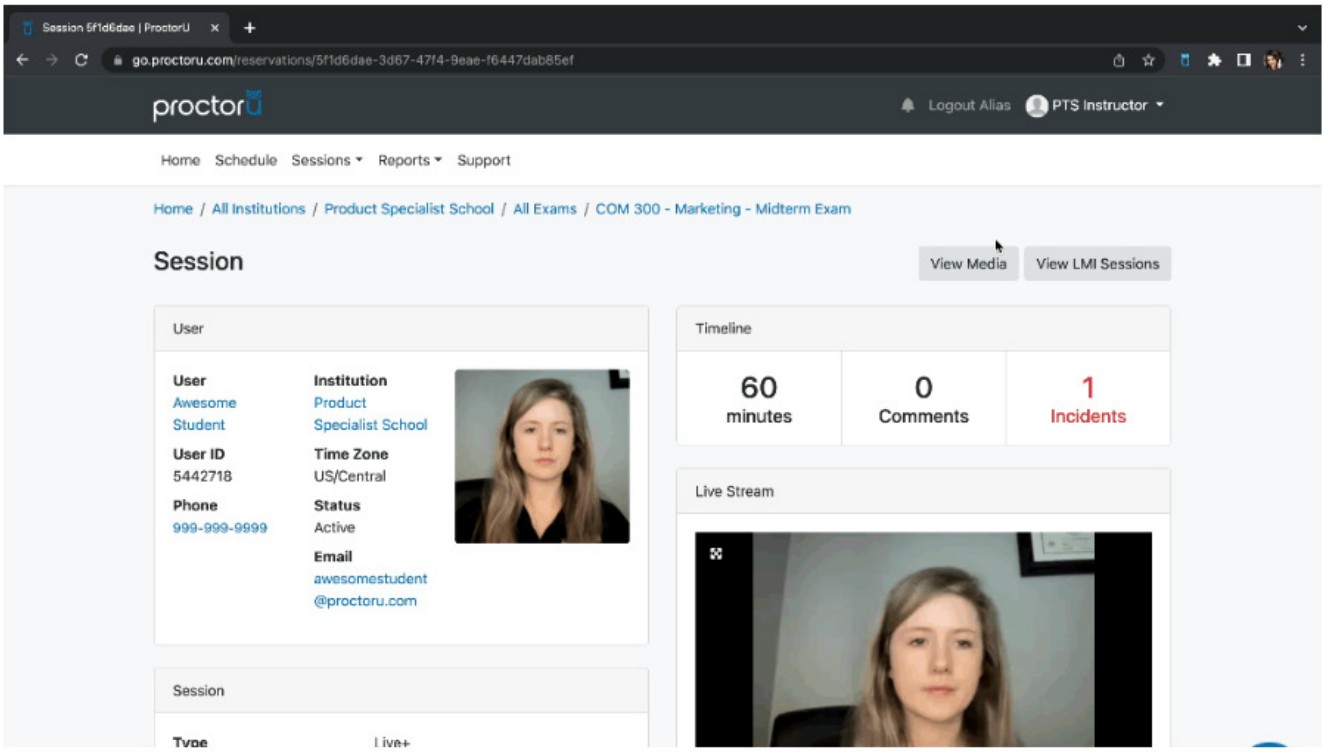


Рисунок 1.1 – Вікно роботи в системі ProctorU

Система ProctorU підтримує автоматизований і живий контроль. Перед початком іспиту студенти підтверджують свою особу, звіряючи свій студентський квиток із фотографією, яка зберігається в базі даних закладу. ProctorU доступний у трьох версіях: Live+ позначає та запобігає підозрілій поведінці шляхом негайного втручання людини-проктора. Review+ та Record+: використовують технології штучного інтелекту для виявлення та позначення підозрілих подій, які переглядаються після закінчення іспиту. Попередній аналіз натискання клавіш перевіряє, чи відвідує сеанс та сама особа. Для роботи ProctorU потрібне розширення Chrome або Firefox [16, 17].

2. Respondus Monitor - надає повністю автоматизоване рішення для онлайн-іспитів, яке можна інтегрувати з багатьма існуючими системами керування навчанням (рисунок 1.2).

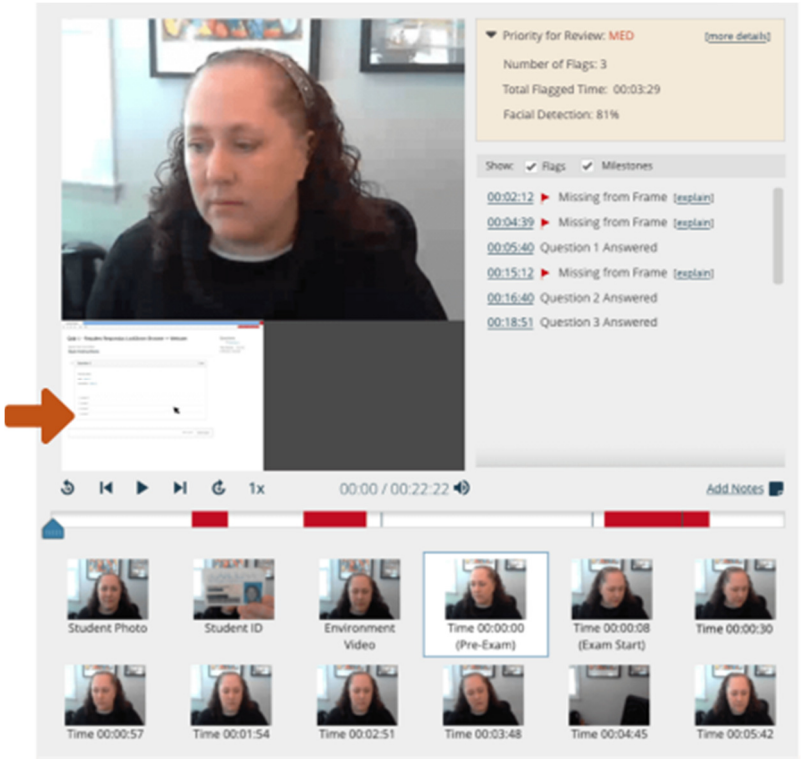


Рисунок 1.2 – Вікно роботи з модулем Respondus Monitor

Модуль Respondus Monitor який захищає екзаменаційні запитання блокуючи функції друку, комбінації клавіш, копіювання та вставлення, програми для захоплення екрана, доступ до інших програм або відвідування інших веб-сайтів. Після інсталяції програмне забезпечення автоматично запускається у веб-переглядачі студента для процесу налаштування, який включає перевірку аудіо/відео веб-камери. Послідовність попереднього обстеження необхідна для ідентифікації студента та сканування приміщення. Під час іспиту веб-камера записує студента, а система штучного інтелекту (Monitor AI) пізніше проводить аналіз відео, щоб виявити аномальну поведінку. Після завершення цих операцій система створює звіт, що зберігається на порталі для пріоритетного рецензування, який ранжує результати контролю відповідно до ризику порушення іспиту [18, 19].

3. ProctorExam — це хмарне рішення, що відповідає загальному регламенту захисту даних (GDPR) і може функціонувати як автономне програмне забезпечення або бути інтегрованим у системи керування навчанням (рис 1.3).

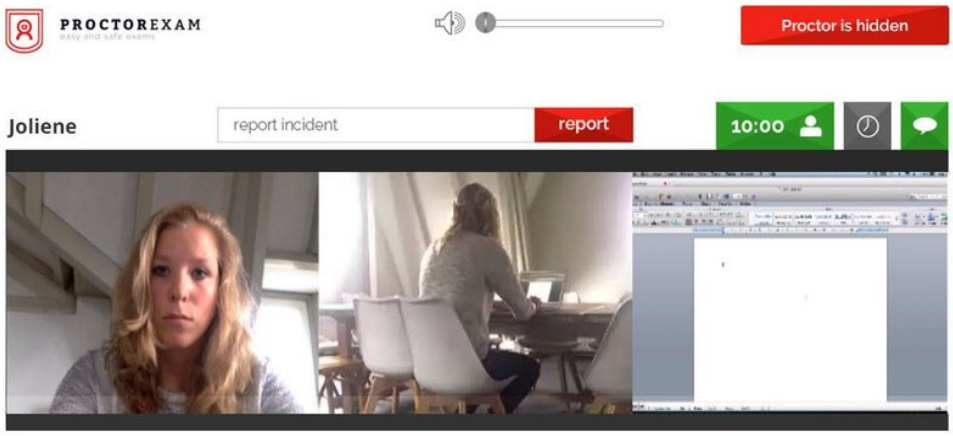


Рисунок 1.3 – Вікно роботи з системою ProctorExam

Веб-інфраструктура ProctorExam пропонує рішення для контролю в реальному часі, запису та перегляду, а також нагляду за тестувальниками за допомогою до трьох одночасних каналів: спільне використання екрана, аудіо/відео веб-камери та додаткова мобільна камера, з'єднана з мобільним додатком на Android або iOS на смартфонах або планшетах кандидатів. ProctorExam забезпечує 360° огляд робочого простору кандидата, дозволяючи записувати весь тестовий сеанс під різними кутами. Відсутність підтримки ШІ для виявлення неправомірної поведінки компенсується інтерфейсом моніторингу, який відображає всіх активних кандидатів на панелі. Ротація між кандидатами відбувається автоматично кожні 7 секунд, забезпечуючи рівний час перевірки для всіх учасників тесту. Наприкінці іспиту рецензент аналізує записані відеопотоки та дії студентів у веб-переглядачі, позначаючи конкретні сеанси, які вважає аномальними, і розміщуючи мітки часу на інцидентах [20, 21].

4. Examity інтегровано з різними системами управління навчанням (LMS), що дозволяє студентам створювати особисті профілі перед початком іспиту. Етап перед іспитом включає в себе комплексну автоматичну аутентифікацію: студенти надають офіційне посвідчення особи та зображення, зроблене в реальному часі за допомогою веб-камери. Крім того, система вимагає від студентів відповісти на

низку запитань для подальшої перевірки їх особи та надання цифрового підпису, що також аналізується для визначення частоти натискання клавіш. Використовуючи ці дані, система порівнює ідентифікатор, зображення в реальному часі та метрику натискання клавіш з інформацією, зареєстрованою перед іспитом.

Після аутентифікації студент завантажує та встановлює програмне забезпечення ExamiLock, яке дозволяє моніторити робоче місце та відслідковувати початок іспиту. Весь процес іспиту контролюється алгоритмами штучного інтелекту, які фіксують та мітять час можливих інцидентів. Відеозаписи зберігаються на інформаційній панелі Examity, де екзаменатори можуть їх переглядати та аналізувати [22, 23].

5. Служба захисту Honorlock функціонує як розширення для Google Chrome, яке вимикає функції копіювання та друку, а також запобігає використанню декількох моніторів і доступу до файлів, що зберігаються на комп'ютері. Перед початком іспиту система виконує запис обличчя та ідентифікатора студента для автентифікації, а також створює 360-градусне сканування кімнати. В ході іспиту ведеться запис екрану, аудіо, відео та веб-активності студента. Алгоритми штучного інтелекту виявляють присутність сторонніх голосів та використання вторинних пристроїв, як-от смартфони, планшети або ноутбуки. Також, у випадку виявлення аномальної діяльності студента під час іспиту, на екрані з'являється спливаюче повідомлення від проктора. Це дозволяє значно заощадити час на аналіз звітів та перегляд відеозаписів після іспиту [24, 25].

6. ExamSoft блокує пристрій студента та вимикає з’єднання Wi-Fi, щоб запобігти будь-яким спробам шахрайства. Студент розпочинає іспит, входячи в процедуру підтвердження особи, яка включає двоетапний процес автентифікації. Після початку іспиту, ExamMonitor починає записувати аудіо та відео, фіксуючи поведінку студента під час іспиту. Ці записи створюють детальний журнал усіх дій студента. На основі цих даних, програмне забезпечення з штучним інтелектом аналізує записи, визначаючи аномалії, такі як незвичайні рухи тіла чи очей. Воно також виявляє потенційні "червоні прапорці" на основі фонового шуму, що можуть вказувати на можливі порушення правил іспиту [26, 27].

7. Безпечний онлайн браузер (Safe Exam Browser, SEB) — це програма з відкритим вихідним кодом, розроблена для створення «безпечного середовища», що дозволяє студентам складати іспити вдома, використовуючи власні пристрої (рис. 1.4). Наразі SEB доступний для операційних систем Windows і Mac. SEB включає різноманітні технології, що запобігають доступу до заборонених ресурсів або програм, наприклад, шляхом блокування веб-сайтів, обмеження функцій копіювання й вставки та вимкнення доступу до інших програм [6].

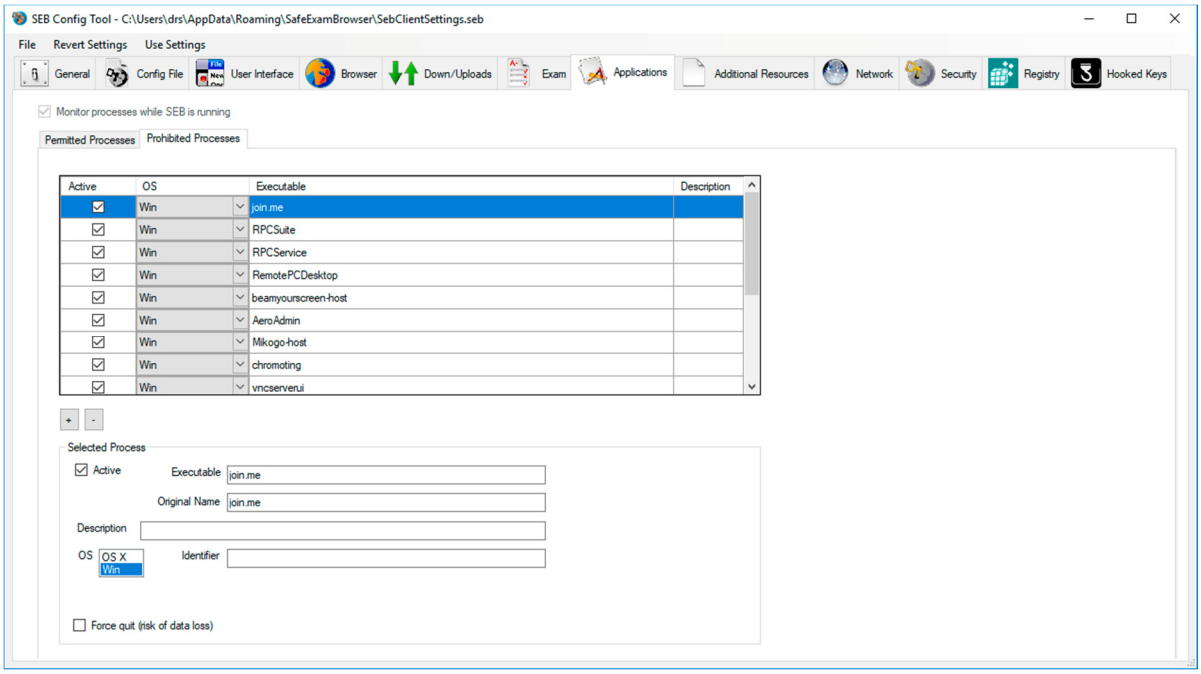


Рисунок 1.4 – Вікно конфігурації SEB

8. Proctortrack пропонує живу та повністю автоматизовану онлайн-службу віддаленого контролю, що постійно перевіряє особу учасників онлайн-тесту, одночасно виявляючи та запобігаючи будь-яким порушенням. Proctortrack забезпечує повну інтеграцію з усіма основними системами керування навчанням, включаючи Moodle, Canvas, Sakai, Blackboard, edX та Brightspace, і може функціонувати навіть під час перебоїв у роботі Інтернету, забезпечуючи неперервний контроль і збереження даних для подальшої обробки після відновлення з’єднання. За допомогою алгоритмів розпізнавання облич платформа контролює іспит студента, відстежуючи його поведінку, включаючи перебування

на робочому місці, пошуки в Інтернеті додаткових ресурсів або спілкування з іншою людиною. Система здійснює перевірку особи студента через багатофакторний біометричний аналіз, включаючи характеристики обличчя, ідентифікаційні картки та сканування пальців, які порівнюються з базовим біометричним профілем студента, збереженим у системі. Proctortrack також надає мобільну програму, яка використовує передню камеру комп'ютера для здійснення ідентифікації обличчя та підтвердження особи в реальному часі під час іспиту [28].

Слід зазначити, що більшість викладачів що використовували веб-браузер SEB додавали власну тактику для ефективного моніторингу поведінки студентів під час онлайн-іспитів і з моменту впровадження технологій (SEB із Zoom) жоден із викладачів, які брали участь у опитуваннях, не помітив кричущих випадків списування чи зіткнувся зі скаргами студентів на несправедливість іспиту .

1.3. Постановка задачі дослідження

Традиційні тести проводяться в аудиторіях у спеціально відведених місцях – екзаменаційних центрах. Проте з цим пов’язано ряд труднощів, оскільки екзаменаційні центри повинні бути доступні на момент для проведення іспитів, що включає витрати на визначення місць проведення тестування, створення необхідної інфраструктури, забезпечення відповідною технікою (комп’ютери, принтери) та наймання екзаменаторів, спостерігачів для нагляду за тестуванням. Ну і самі кандидати і езаменатори мають добиратись самі до місць тестування. В результаті витрачається час та кошти. Якщо тестування відбувається без комп’ютерів, то має бути доступна достатня кількість аркушів для відповідей і запитань, які необхідно роздрукувати, бланки фізично транспортуються до місць іспиту, що теж займає відповідні витрати. Тому традиційні іспити з використанням паперу є дорогими для проведення.

У зв'язку зі зростанням популярності онлайн-тестувань та необхідністю забезпечення чесності та надійності цих процесів, потрібно постійно досліджувати

можливості розробки системи виявлення зловживань під час онлайн-тестувань за допомогою методів машинного навчання та обробки зображень, відео та звуку з використанням відеокамери, мікрофону та копії екрану.

Розробка такого типу системи повинна забезпечувати постійний моніторинг дій студента під час тестування, аналізуючи рухи голови, очей, губ та інші фактори, які можуть свідчити про зловживання. Вона повинна виявляти підозрілі дії, такі як спроби перегляду заборонених матеріалів, спілкування з іншими особами або використання додаткових пристроїв для допомоги під час тестування.

Очікується, що розроблена система зможе ефективно виявляти зловживання під час онлайн-тестувань і допоможе покращити чесність та надійність процесу оцінювання студентів.

Розробка такої системи дозволить отримати практичний досвід у розробці та впровадженні інноваційних технологій у сфері онлайн-освіти. Така система буде відігравати важливе значення для академічного середовища, оскільки відкриває можливості для вдосконалення систем онлайн-тестувань та забезпечення їхньої ефективності та надійності.

2. АЛГОРИТМІЧНЕ ТА ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

2.1. Загальна структура проектованої системи

Загальна структура системи системи онлайн-прокторингу з використанням штучного інтелекту для забезпечення академічної доброчесності під час онлайн-іспитів складається з кількох взаємопов'язаних компонентів і користувачів, включаючи адміністратора, студентів, відповідальних співробітників та центральний сервер.

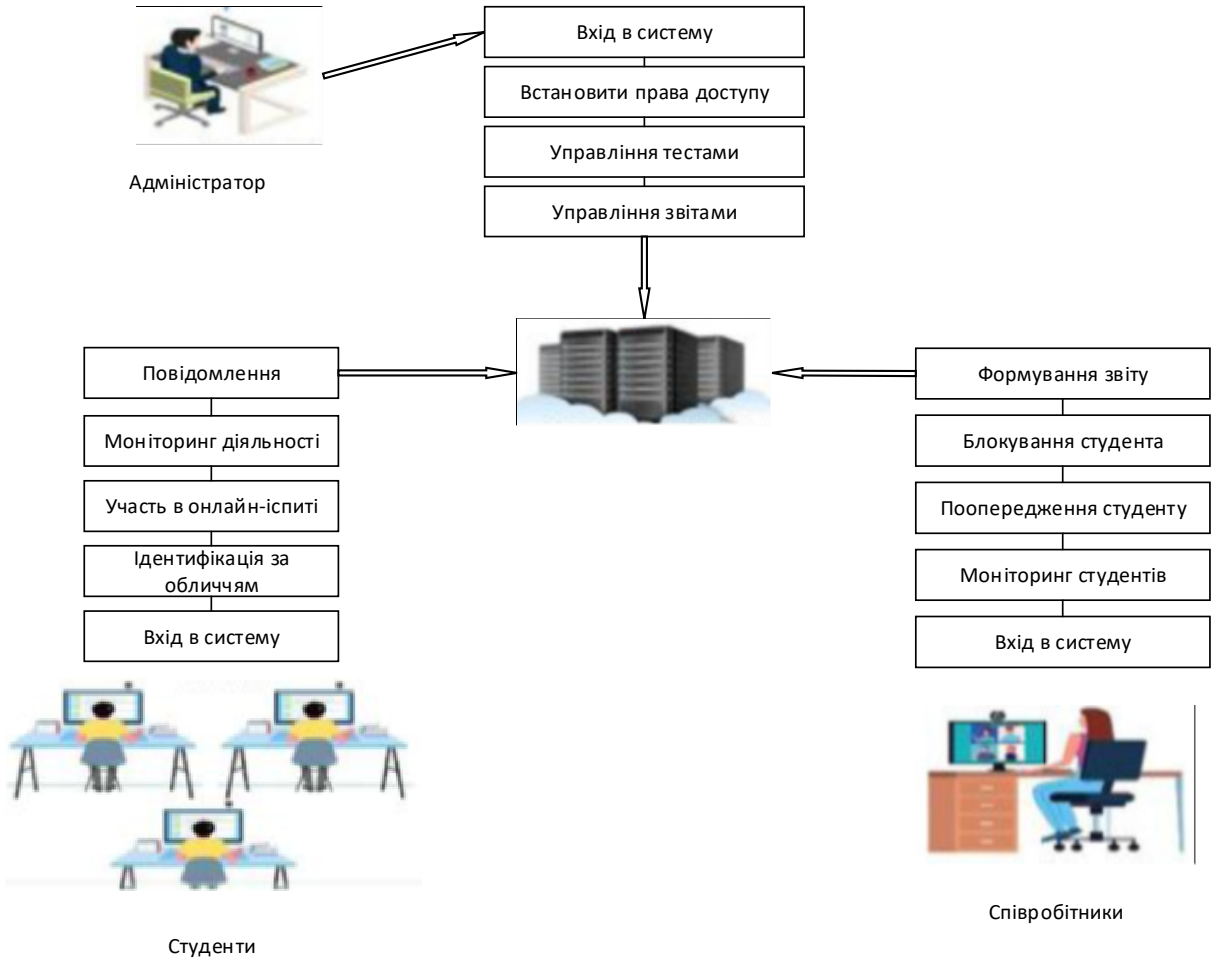


Рисунок 2.1 – Загальна структура системи

Адміністратор системи входить в систему, і назначає за потреби права доступу користувачам системи в базі даних, за потреби створює питання для іспитів, надає посилання на іспит в системі тестування (наприклад moodle), встановлює час початку та завершення іспитів та генерує різноманітні звіти,

пов'язані з іспитами і прокторингом. Центральний сервер є ключовим елементом системи, який взаємодіє з усіма користувачами та управляє основними функціями. Він зберігає попередження про підозрілі дії, виявлені випадки шахрайства, забезпечує процес онлайн-іспитів, аутентифікує користувачів та здійснює розпізнавання облич.

Студенти входять в систему, проходять ідентифікацію за обличчям і беруть участь в онлайн-іспитах. Система постійно моніторить їхні дії, виявляє шахрайство і відправляє попередження в разі необхідності. Співробітники, відповідальні за прокторинг, також входять в систему для спостереження за студентами в реальному часі, можуть видавати попередження та блокувати студентів у випадку виявлення шахрайства, а також мають можливість генерувати звіти.

Процес роботи системи починається з налаштування іспиту адміністратором, який додає питання і налаштовує час іспитів. Потім студенти проходять ідентифікацію і беруть участь в іспитах, під час яких система моніторить їхні дії. Після завершення іспитів адміністратори і відповідальні співробітники можуть переглядати звіти для аналізу проведення іспитів і виявлених випадків шахрайства. Система забезпечує безпечний, контрольований і ефективний процес онлайн-іспитів, підтримуючи академічну доброчесність за допомогою своїх комплексних функцій і взаємодії між різними ролями користувачів.

2.2. Алгоритмічне забезпечення

Будучи системою взаємодії між людиною та комп’ютером, система в основному фіксує відео та аудіосигнали онлайн-користувача, щоб визначити статус користувача під час іспиту. Клієнт-серверна архіектура забезпечує розподіл функцій і навантаження на клієнтський і серверний частини. На стороні сервера зберігаються ідентифікаційні дані всіх досліджуваних і важливих змін в поведінці, включаючи вираз обличчя, рух очей і рота та мову. На стороні клієнта відбувається аналіз мультимедійних даних, користувача який пише онлайн-тест локально, і

передає найважливіші моменти зміни поведінки особи що проходить тест на сервер.

Відео та аудіосигнали отримуються з апаратних пристроїв користувача (відеокамери та мікрофони), які можуть бути вбудовані в ноутбук або зовні підключені до настільного комп'ютера. Захоплення сигналу відстежується в режимі реального часу. Проте швидкість захоплення (кількість кадрів за секунду, FPS), залежить від продуктивності апаратного забезпечення. Зазвичай швидкість може становити 30 кадрів в секунду, а роздільна здатність кожного кадрового зображення теоретично може становити від 480p до 1080p.

Відеопотік і аудіопотік аналізуються відеомонітором і аудіомінітором відповідно. Відеомонітор розроблений на основі наступних трьох компонентів: бібліотеки dlib, face-api.js.

Система контролю потребує функції безпечної перевірки входу з ідентифікацією обличчя. Перед входом в систему учасник тестування вводить ім'я користувача, персональні дані та пароль для першої ідентифікації. Оскільки під час першого входу в систему немає еталонного обличчя, обстежуваного просять зробити один знімок у вікні захопленого відео обличчя. Цей знімок аналізується розпізнавачем обличчя на стороні клієнта, і його характеристики завантажуються в базу даних сервера як дескриптори обличчя обстежуваного. Звичайно, можна розширити функціональність системи де можна додати можливість зробити більше знімків, щоб більше варіантів обличчя зберігались на сервері. Середнє значення набору ознак одного обстежуваного представлятиме обстежуваного та використовуватиметься як еталон обстежуваного. Ідентифікація одного досліджуваного може бути точнішою за допомогою більшої кількості відновлених рис зі знімків обличчя.

Після ідентифікації імені користувача та пароля обличчя досліджуваного, дані зняті відеокамерою, постійно аналізуються та порівнюються з відповідними характеристиками що зберігаються в базі даних сервера. Якщо будь-яка функція збігається, випробуваний проходить перевірку обличчя, йому дозволяється перейти на головну сторінку іспиту та писати відповіді на запитання в іспитах.

Окрім ідентифікації орієнтирів, можна додати функцію виявлення підробки обличчя, щоб запобігти проходженню перевірки тим, хто використовує фото певної людини. Виявлення підробки обличчя в основному реалізується через аналіз дій рота обстежуваного, так що обстежуваного просять відкрити та закрити рот, щоб пройти перевірку. На рисунку представлено Алгоритм роботи системи.

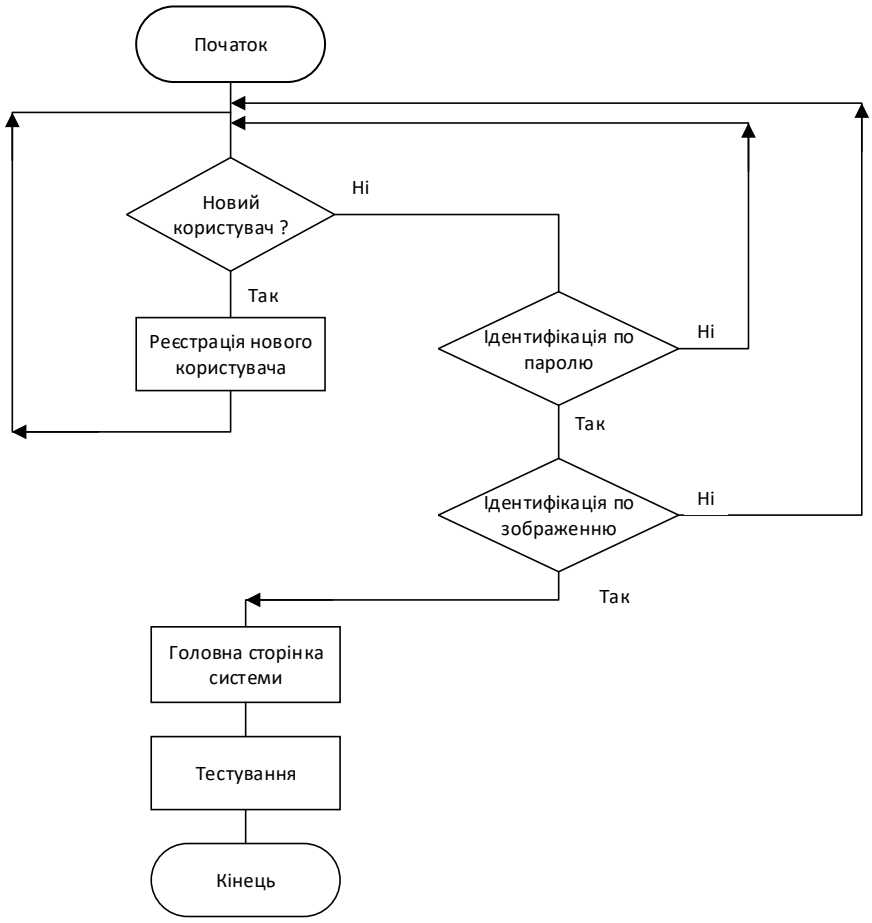


Рисунок 2.2 – Алгоритм входу в систему

Після перевірки безпечного входу з ідентифікацією обличчя система позначає статус випробуваного як «розпізнаний», і випробуваний заходить на веб-сторінку тесту та починає тестування. Коротку архітектуру контролю над процесом тестування представлено на рисунку 2.3, де метадані визначаються як сукупні дані, що надходять від голосу, обличчя, та інших дій. Під час іспиту відео- та аудіосигнал буде зафіксовано апаратними пристроями, зокрема камерами та мікрофонами на клієнтській машині. Відеомонітор безперервно аналізує захоплене

відеозображення та протоколює статус обстежуваного одним із трьох наступних критеріїв: відсутній, повторний і багаторазовий. Якщо людське обличчя не виявлено, тобто жоден обстежуваний не виявлений перед відеокамерою, система позначає цей статус як «відсутність» обстежуваного. Якщо наступного разу буде виявлено одне людське обличчя, і це обличчя буде розпізнано як обличчя обстежуваного шляхом порівняння ознак розпізнаного обличчя і отриманого з бази даних сервера, система позначає цей статус як «повторний». Якщо виявлено більше ніж одне людське обличчя, система позначає цей статус як «декілька».

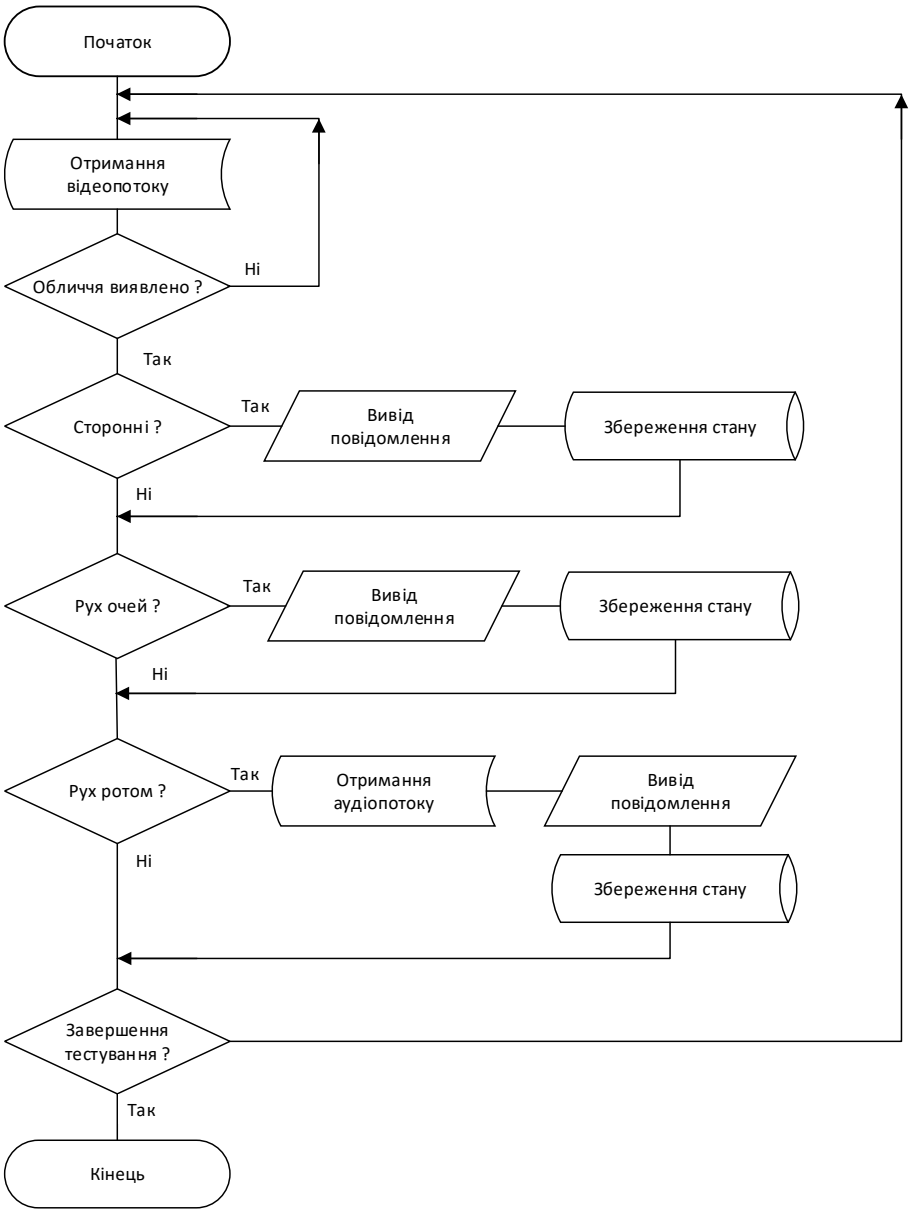


Рисунок 2.3 – Алгоритм участі в тестуванні

Додатковою функцією для перевірки чи не використовується фото опитуваного замість реального обличчя опитуваного може бути вимога відкривати і закривати рот. Розпізнавання емоцій використовується для перевірки динаміки виразу обличчя обстежуваного.

Для вирішення виявлення позиції зіниць та визначення напрямку погляду людини спочатку визначається координати області очей за допомогою бібліотеки face-api.js, яка використовується для виявлення облич та визначення ключових точок на обличчі. Це дозволяє виділити конкретні регіони очей для подальшої обробки. Після цього використовується метод методу Blob Detection з бібліотеки OpenCV для виділення зіниць. Спершу, зображення регіону ока перетворюється в градації сірого для спрощення подальшої обробки. Потім застосовується порогове значення для виділення найтемніших областей, які відповідають зіницям. Знайдені контури допомагають виділити зіниці серед інших елементів на зображенні. Наступним кроком є визначення точного положення зіниць. Для цього обчислюєьбся центр найбільшого контуру, який ідентифікується як зіниця. Це дозволяє точно визначити її координати. Останнім етапом є визначення напрямку погляду. Порівнюючи положення зіниці з центром ока, можна визначити, куди дивиться людина: вправо, вліво, вгору або вниз. На основі цього аналізу виводиться повідомлення чи студен відводить погляд від монітора чи ні.

Аудіомонітор може виявляти голосову активність, і з потреби - розпізнавати голосові повідомлення. Виявлення голосової активності може використовуватись з метою запобігання зачитуванню запитань опитуваним, чи прослузовування ним відповідей від сторонньої особи. Додатково систему можна буде розширити функцією розпізнавання голосу, за допомогою системи Julius.

Також можна представити Use Case діаграму, що де розписано процес онлайн іспиту з виявлення академічної недоброчесності. Головний актор – це користувач, який є учнем/студентом, що здає онлайн іспит. Перший крок для користувача – це вхід у систему, використовуючи засіб аутентифікації. Після успішного входу до системи, користувач може перейти до процесу іспиту де він входить на онлайн іспит.

Під час іспиту користувач може взаємодіяти з системою моніторингу, яка включає кілька модулів для забезпечення чесного проходження іспиту. Система моніторингу складається здійснює аутентифікацію користувача для підтвердження присутності зареєстрованої особи та перевірки наявності сторонніх осіб, здійснює виявлення обличчя, відслідковування рухів очей та рота, запис звуків та перевірки наявності сторонніх осіб.

Користувач також має можливість відправити відповіді на питання іспиту через систему, а в разі потреби – запитати допомогу під час іспиту. Усі ці дії користувача взаємодіють з інтерфейсом системи, яка управляє процесом іспиту та моніторингом. Якщо система виявляє академічну недоброчесність, вона повідомляє про це.

Таким чином, діаграма відображає інтеграцію різних модулів моніторингу для забезпечення академічної чесності під час онлайн іспиту. Система включає в себе модулі для виявлення облич, виявлення сторонніх осіб, відслідковування рухів очей, відслідковування рухів рота та виявлення звуків, що забезпечує комплексний підхід до виявлення академічної недоброчесності.

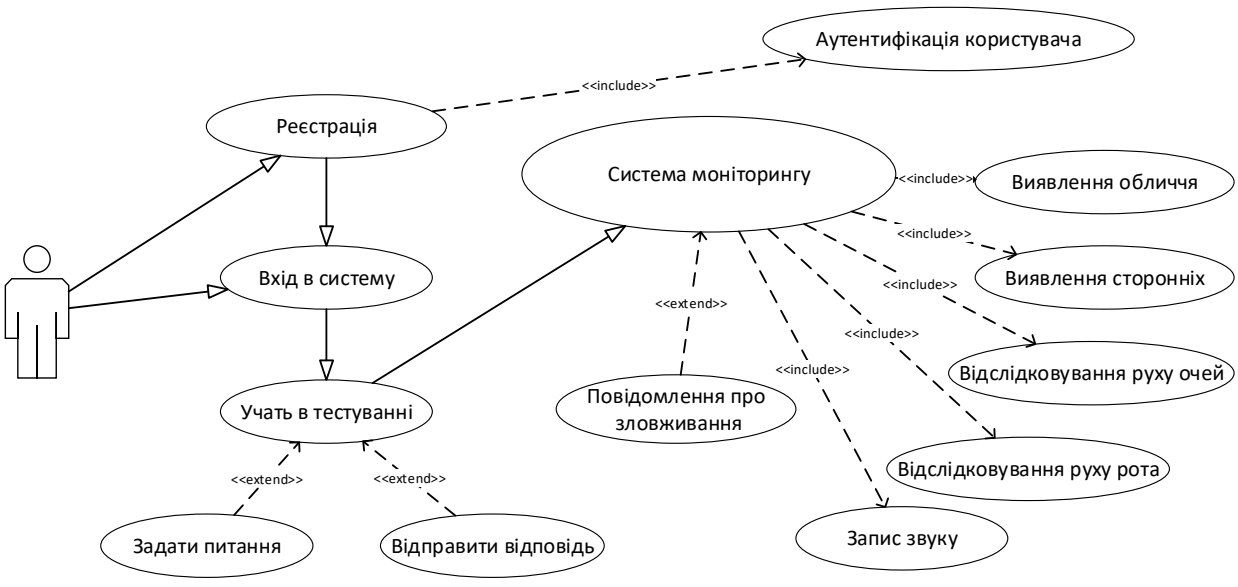


Рисунок 2.4 - Use Case діаграма

Використання засобів штучного інтелекту в онлайн системах складання іспитів з дасть змогу полегшити процес проведення та здачі іспитів як для

студентів, так і для вчителів/викладачів. Ця система дозволяє користувачам виконувати різні дії залежно від їх ролі в системі.

Діяльність студента зводиться до наступних дій, спочатку він здійснює реєстрацію у системі. Для цього він повинен заповнити онлайн форму, що дозволяє системі зібрати необхідну інформацію про користувача. Після успішної реєстрації студент може увійти в систему, пройшовши процес аутентифікації. Після входу студент має можливість відповідати на питання іспиту. Коли всі відповіді надані, студент може відправити і завершити свою роботу. Після цього він може переглянути результати свого іспиту, що додає прозорості та допомагає студентам оцінити свої знання.

Діяльність вчителя також починається із входу в систему, проходячи процес аутентифікації. Після входу вчитель може створити набір питань для іспиту, додаючи, оновлюючи або видаляючи питання з бази даних. Це дозволяє вчителю підтримувати актуальність і релевантність питань.

Крім того, вчитель може переглядати заявки студентів, що дозволяє йому оцінювати та перевіряти відповідність студентів до іспиту. Після завершення іспиту вчитель може оприлюднити результати, що робить їх доступними для студентів.

У системі також є можливість публікації результатів, що є продовженням процесу оголошення результатів. Це дозволяє вчителю легко розповсюджувати інформацію про результати іспитів серед студентів.

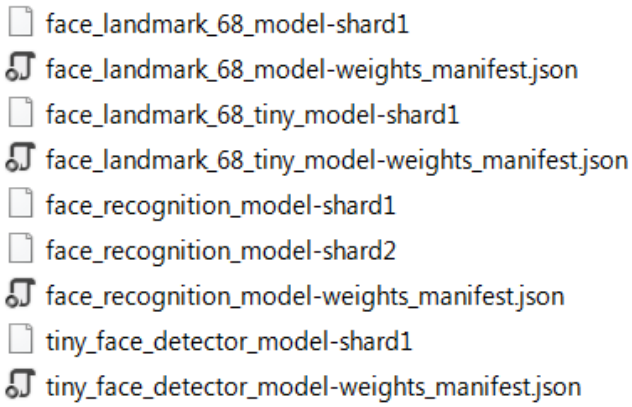
У системі використовується кілька ключових механізмів, які забезпечують її ефективну роботу. Заповнення онлайн форми є важливою частиною процесу реєстрації для студентів, а аутентифікація є необхідною для входу як студентів, так і вчителів. Крім того, система підтримує розширення дій, що дозволяє додавати додаткові кроки до основних процесів, такі як відповідь на питання та здача роботи.

Таким чином, ця онлайн система складання іспитів з використанням штучного інтелекту є комплексним рішенням, що забезпечує ефективне проведення іспитів, прозорість оцінювання та зручність для всіх користувачів.

2.3. Інформаційне забезпечення

2.3.1. Детектори облич

Для своєї роботи програмна система використовує Face-api.j, що являє собою бібліотеку API розпізнавання обличчя на JavaScript для браузера та node.js, яка працює на основі TensorFlow.js. Вона дозволяє легко інтегрувати функціональність розпізнавання облич у веб-додаток, надаючи можливість виявляти обличчя, визначати лендмарки, розпізнавати емоції та виконувати інші складні завдання з обробки облич у реальному часі. У даному проекті face-api.js використовується для виявлення облич у відеопотоці, визначення основного обличчя та виявлення рухів очей і рота. Існує ряд моделей котрі потрібно завантажити для роботи з face-api.js:



SSD Mobilenet V1 (Single Shot Multibox Detector) - детектор облич, де задіяні детектор SSD (Single Shot Multibox Detector) на основі моделі MobileNetV1. Нейронна мережа обчислює розташування кожного обличчя на зображенні та повертає обмежувальні прямокутники разом із ймовірністю для кожного обличчя. Цей детектор обличчя націлений на отримання високої точності виявлення обмежувальних контурів обличчя замість малого часу висновку. Розмір квантованої моделі становить приблизно 5,4 МБ (ssd_mobilenetv1_model). Модель розпізнавання обличчя навчена на наборі даних WIDER FACE, а ваги надаються користувачем. Детектор SSD Mobilenet V1 був обраний завдяки своїй здатності знаходити баланс між точністю та швидкістю. SSD є сучасною архітектурою для виявлення об'єктів, яка об'єднує кілька масштабів обчислень, що дозволяє обробляти об'єкти різних розмірів ефективно. Використання MobileNetV1,

легковагової архітектури глибоких нейронних мереж, додатково забезпечує зменшення розміру моделі без значної втрати точності. Дана модель навчена на наборі даних [29, 30].

Tiny Face Detector — це дуже продуктивний детектор обличчя в режимі реального часу, який набагато швидший, менший і споживає менше ресурсів порівняно з детектором обличчя SSD Mobilenet V1, натомість він дещо гірше розпізнає маленькі обличчя. Ця модель є надзвичайно мобільною та дружньою для використання в Інтернеті, тому вона є основним детектором обличчя на мобільних пристроях та клієнтах з обмеженими ресурсами. Розмір квантованої моделі становить лише 190 КБ (tiny_face_detector_model). Детектор обличчя було навчено на спеціальному наборі даних із ~14 тис. зображень, позначених обмежувальними рамками. Крім того, модель навчена передбачати обмежувальні прямокутники, які повністю охоплюють точки рис обличчя, тому вона загалом дає кращі результати в поєднанні з подальшим виявленням орієнтирів обличчя, ніж SSD Mobilenet V1. Ця модель, по суті, є ще меншою версією Tiny Yolo V2, яка замінює звичайні згортки Yolo на згортки, які можна розділити по глибині. Yolo повністю згортковий, тому може легко адаптуватися до різних розмірів вхідного зображення, щоб поступитися точністю продуктивності (час висновку).

Face Landmark 68 – модель для визначення 68 лендмарок (точок) на обличчі, таких як очі, ніс, рот тощо. МОдкль реалізує дуже легкий, швидкий і точний 68-точковий детектор орієнтирів обличчя. Модель за замовчуванням має розмір лише 350 Кб (face_landmark_68_model), а мініатюрна модель — лише 80 Кб (face_landmark_68_tiny_model). Обидві моделі використовують ідеї роздільних по глибині звивин, а також щільно зв’язаних блоків. Моделі були навчені на наборі даних із приблизно 35 тисяч зображень облич, позначених 68 орієнтирами.

Face Recognition – модель для створення дескрипторів облич, які дозволяють ідентифікувати та розпізнавати обличчя. МОдель подібна до ResNet-34, для обчислення дескриптора обличчя (вектор ознак із 128 значеннями) з будь-якого зображення обличчя, який використовується для опису характеристик обличчя людини. Модель не обмежується набором облич, які використовуються для

навчання, тобто ви можете використовувати її для розпізнавання обличчя будь-якої людини, наприклад, себе. Ви можете визначити подібність двох довільних облич, порівнюючи їхні дескриптори облич, наприклад, обчислюючи евклідову відстань або використовуючи будь-який інший класифікатор на ваш вибір. Нейронна мережа еквівалентна FaceRecognizerNet, що використовується в face-recognition.js і сітка, яка використовується в dlib приклад розпізнавання обличчя. Ваги були навчені davisking і модель досягає точності передбачення 99,38% за тестом LFW (Labeled Faces in the Wild) для розпізнавання облич. Розмір квантованої моделі становить приблизно 6,2 МБ (face_recognition_model) [30].

MTCNN (багатозадачна каскадна згорточна нейронна мережа) представляє собою альтернативний детектор обличчя SSD Mobilenet v1 і Tiny Yolo v2, який пропонує набагато більше можливостей для конфігурації. Налаштувавши вхідні параметри, MTCNN повинен мати можливість виявляти широкий діапазон розмірів обмежувальної рамки обличчя. MTCNN — це 3-етапний каскадний CNN, який одночасно повертає 5 орієнтирів обличчя разом із обмежувальними рамками та балами для кожного обличчя. Крім того, розмір моделі становить лише 2 МБ.

Моніторинг звуку є важливим компонентом даної системи, який дозволяє визначати наявність звуку у середовищі та реагувати на нього. Для реалізації цієї функціональності ми використано Web Audio API, що надає інструменти для захоплення звуку з мікрофона та аналізу аудіосигналів у реальному часі.

2.3.2. Бібліотеки

Виявлення облич є однією з найвивченіших тем у спільноті комп'ютерного зору. Багато досягнень у спільноті комп'ютерного зору стали можливими завдяки доступності еталонних наборів даних для виявлення облич. Завдяки їм можна оцінити розрив між поточною продуктивністю методів виявлення облич та вимогами реального світу. Великий набір даних WIDER FACE [29, 30], який приблизно в 10 разів більший за попередні набори. Набір даних містить багаті анотації, включаючи перекриття, пози, категорії подій та межі облич. Обличчя в

запропонованому наборі даних є надзвичайно складними через великі варіації в масштабі, позах і перекриттях, як показано на рисунку 2.5.

Бібліотека dlib містить модель орієнтира обличчя, навчена на наборі даних із 35 000 зображень облич, позначених 68 точками/орієнтирами (рис. 2.6) [31]. Обличчя одного користувача можна ідентифікувати за рисами 68 точок. Евклідова відстань між рисами двох граней використовується для розрізнення двох граней. Якщо евклідова відстань між двома гранями більша за порогове значення, наприклад 0,4, дві грані розпізнаються як різні. Для більшої точності поріг можна зменшити.

Рисунок 2.5 – Еталонний набір даних для виявлення облич

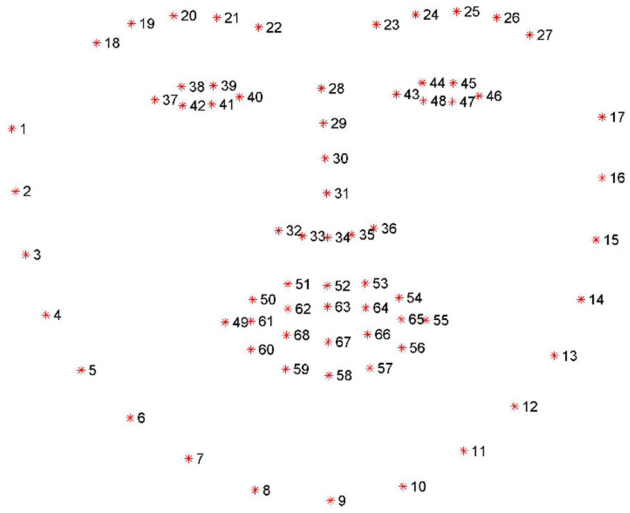


Рисунок 2.6 – 68 орієнтирів/точок обличчя бвбліоеки dlib.

Бібліотека OpenCV (Open Source Computer Vision Library) — це відкрита бібліотека, яка містить більше 2500 оптимізованих алгоритмів комп'ютерного зору та машинного навчання. Вона забезпечує інструменти для аналізу відео та зображень, розпізнавання об'єктів, виявлення руху, відстеження об'єктів та багато іншого. OpenCV активно використовується для створення систем автоматизації, робототехніки, медичної діагностики, обробки зображень та відео [32].

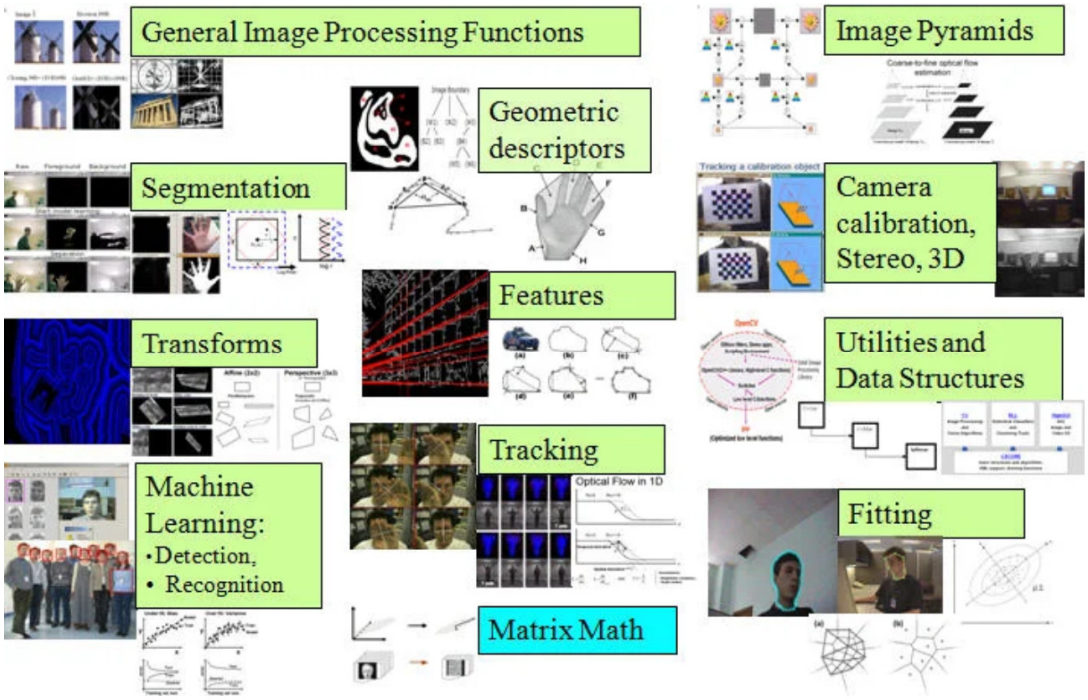


Рисунок 2.7 – Основні функції бібліотеки OpenCV

2.3.3. База даних

Для збереження інформації про користувачів системи та стан проходження тестувань кожним із учасників необхідно використати базу даних. Запропоновано використати реляційну модель даних, представлену у відношеннях, які реалізовуються у конкретних таблицях з даними.

Таблиця Roles призначена для зберігання інформації про ролі користувачів у системі. Вона складається з двох полів: id, який є унікальним ідентифікатором ролі і автоматично збільшується, та role_name, який містить назву ролі. Кожен запис у цій таблиці відповідає певній ролі користувача, наприклад "адміністратор",

"координатор", "користувач". Таблиця Roles пов'язана з таблицею Users, де кожен користувач має роль, вказану через поле `role_id`. Код створення таблиці:

```
CREATE TABLE Roles (
    id INT AUTO_INCREMENT PRIMARY KEY,
    role_name VARCHAR(255) NOT NULL
);
```

Таблиця Countries містить інформацію про країни. Вона має два поля: id, що є унікальним ідентифікатором країни і автоматично збільшується, та country_name, яке містить назву країни. Ця таблиця використовується для зберігання даних про різні країни, що можуть бути пов'язані з користувачами або іншими об'єктами системи. Код створення наблиці:

```
CREATE TABLE Countries (
    id INT AUTO_INCREMENT PRIMARY KEY,
    country_name VARCHAR(255) NOT NULL
);
```

Таблиця Cities зберігає інформацію про міста і їх зв'язок з країнами. Вона включає три поля: id, що є унікальним ідентифікатором міста і автоматично збільшується, city_name, яке містить назву міста, та country_id, яке посилається на id у таблиці Countries. Таким чином, кожне місто пов'язане з певною країною. Код створення таблиці:

```
CREATE TABLE Cities (
    id INT AUTO_INCREMENT PRIMARY KEY,
    city_name VARCHAR(255) NOT NULL,
    country_id INT NOT NULL,
    FOREIGN KEY (country_id) REFERENCES Countries(id)
);
```

Таблиця Users містить детальну інформацію про користувачів системи. Вона включає наступні поля: id (унікальний ідентифікатор користувача, що автоматично збільшується), first_name (ім'я користувача), last_name (прізвище користувача), middle_name (по-батькові користувача, яке є необов'язковим), birth_date (дата народження), email (електронна пошта, яка є унікальною для кожного користувача), address (адреса), city_id (ідентифікатор міста, що посилається на id у таблиці Cities), country_id (ідентифікатор країни, що посилається на id у таблиці

Countries), role_id (ідентифікатор ролі, що посилається на id у таблиці Roles), created_at (час створення запису, за замовчуванням встановлюється на поточний час), photo_path (шлях до фото користувача), descriptor (JSON-дані, що описують користувача), password_hash (хеш пароля користувача) та salt (сіль для хешування пароля). Таблиця Users пов'язана з таблицею Roles через поле `role_id`. Також вона може бути пов'язана з таблицями TestSessions і Results. Код створення таблиці:

```
CREATE TABLE Users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(255) NOT NULL,
    last_name VARCHAR(255) NOT NULL,
    middle_name VARCHAR(255),
    birth_date DATE,
    email VARCHAR(255) UNIQUE NOT NULL,
    address VARCHAR(255),
    city_id INT,
    country_id INT,
    role_id INT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    photo_path VARCHAR(255),
    descriptor JSON,
    password_hash VARCHAR(255) NOT NULL,
    salt VARCHAR(255) NOT NULL,
    FOREIGN KEY (city_id) REFERENCES Cities(id),
    FOREIGN KEY (country_id) REFERENCES Countries(id),
    FOREIGN KEY (role_id) REFERENCES Roles(id)
);
```

Таблиця Tests використовується для зберігання інформації про тести. Вона складається з таких полів: id (унікальний ідентифікатор тесту, що автоматично збільшується), test_name (назва тесту), test_url (URL тесту), access_key (ключ доступу до тесту) та created_at (час створення тесту, який за замовчуванням встановлюється на поточний час). Таблиця Tests пов'язана з таблицями TestSessions і Results, де зберігається інформація про проходження тестів користувачами. Код створення таблиці:

```
CREATE TABLE Tests (
    id INT AUTO_INCREMENT PRIMARY KEY,
    test_name VARCHAR(255) NOT NULL,
    test_url VARCHAR(255) NOT NULL,
    access_key VARCHAR(255) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Таблиця TestSessions зберігає дані про сесії тестування. Вона включає поля: id (унікальний ідентифікатор сесії, що автоматично збільшується), user_id (ідентифікатор користувача, що посилається на id у таблиці Users), test_id (ідентифікатор тесту, що посилається на id у таблиці Tests), start_time (час початку сесії) та end_time (час завершення сесії). Таблиця TestSessions пов'язана з таблицями Users і Tests через відповідні зовнішні ключі. Також вона пов'язана з таблицею Violations, де зберігається інформація про порушення під час тестування. Код створення таблиці:

```
CREATE TABLE TestSessions (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    test_id INT NOT NULL,
    start_time TIMESTAMP NOT NULL,
    end_time TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES Users(id),
    FOREIGN KEY (test_id) REFERENCES Tests(id)
);
```

Таблиця ViolationTypes призначена для зберігання типів порушень. Вона складається з полів: id (унікальний ідентифікатор типу порушення, що автоматично збільшується), type_name (назва типу порушення) та description (опис порушення). Таблиця ViolationTypes пов'язана з таблицею Violations, де зберігається інформація про зафіксовані порушення. Код створення таблиці:

```
CREATE TABLE ViolationTypes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    type_name VARCHAR(255) NOT NULL,
    description TEXT NOT NULL
);
```

Таблиця Violations містить інформацію про конкретні порушення, що були зафіксовані під час сесій тестування. Вона включає поля: id (унікальний ідентифікатор порушення, що автоматично збільшується), session_id (ідентифікатор сесії, що посилається на id у таблиці TestSessions), violation_type_id (ідентифікатор типу порушення, що посилається на id у таблиці ViolationTypes), media_path (шлях до медіафайлу, що зафіксував порушення) та timestamp (час створення запису про порушення, який за замовчуванням встановлюється на

поточний час). Таблиця Violations пов'язана з таблицями TestSessions і ViolationTypes через відповідні зовнішні ключі. Код створення таблиці:

```
CREATE TABLE Violations (
    id INT AUTO_INCREMENT PRIMARY KEY,
    session_id INT NOT NULL,
    violation_type_id INT NOT NULL,
    media_path VARCHAR(255),
    timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (session_id) REFERENCES TestSessions(id),
    FOREIGN KEY (violation_type_id) REFERENCES ViolationTypes(id)
);
```

Реалізовано фізичну модель бази даних в СУБД MySQL за допомогою phpMyAdmin, структура таблиць та зв’язки між ними представлені на рисунку 2.8.

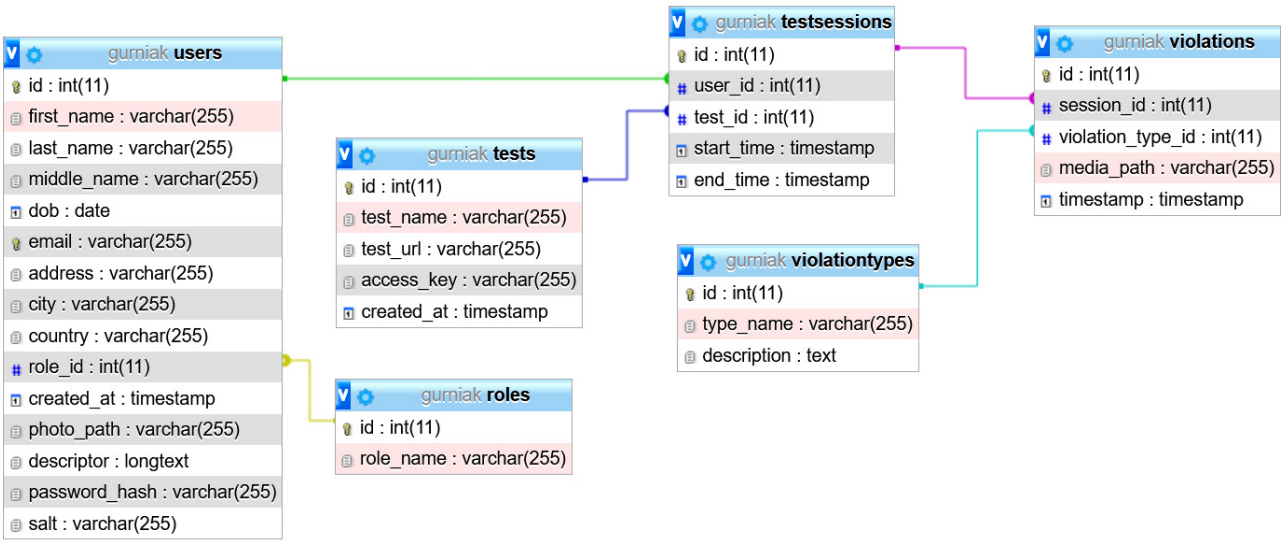


Рисунок 2.8 – Структура бази даних

Дана структура забезпечує цілісність даних, логічну узгодженість, функціональність та безпеку системи, що є критично важливим для ефективного управління даними користувачів та процесами тестування.

3. ПРОГРАМНО-ТЕХНОЛОГІЧНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1. Реалізація програмного забезпечення

3.1.1. Вибір програмних засобів для реалізації

Вибір програмних засобів для реалізації системи базувався на їхній здатності забезпечити високу продуктивність, гнучкість та зручність у використанні. HTML5 та CSS3, Bootstrap надають основу для створення інтуїтивно зрозумілого та привабливого інтерфейсу користувача. JavaScript та face-api.js додають динамічності та інтерактивності, дозволяючи реалізувати розпізнавання облич у реальному часі. TensorFlow.js забезпечує потужну основу для обробки даних та реалізації алгоритмів машинного навчання. Visual Studio Code значно покращує процес розробки завдяки своїм багатофункціональним можливостям, а Git забезпечує ефективне керування версіями та спільну роботу над проєктом. Комбінація цих технологій дозволяє створити ефективну та надійну систему для розпізнавання облич та виявлення рухів у реальному часі [33-36].

Бібліотека face-api.js є потужним інструментом для розпізнавання облич, який працює на основі TensorFlow.js. Вона дозволяє легко інтегрувати функціональність розпізнавання облич у веб-додаток, включаючи виявлення облич, визначення лендмарків та розпізнавання емоцій. Використання face-api.js значно спрощує процес розробки системи розпізнавання облич, забезпечуючи високу точність та продуктивність [37].

TensorFlow.js – це бібліотека для машинного навчання, що дозволяє навчати та виконувати моделі безпосередньо у браузері за допомогою JavaScript. Вона підтримує різноманітні алгоритми та архітектури для розпізнавання образів, обробки тексту та інших завдань. Завдяки TensorFlow.js можна реалізувати потужні функції машинного навчання у веб-додатках, забезпечуючи високу продуктивність та інтерактивність. У проекті TensorFlow.js використовується для роботи з моделями розпізнавання облич, що дозволяє виконувати ці обчислення безпосередньо на стороні клієнта, зменшуючи навантаження на сервер і покращуючи користувацький досвід.

Visual Studio Code (VSCode) – це безплатний редактор коду, розроблений компанією Microsoft, який став одним з найпопулярніших інструментів для розробників завдяки своїй гнучкості, продуктивності та багатофункціональності. VSCode підтримує безліч мов програмування та фреймворків завдяки розширенням, що дозволяє легко адаптувати його під будь-який проєкт.

MySQL – це популярна реляційна система управління базами даних (СУБД) з відкритим вихідним кодом. MySQL використовується для зберігання, організації та управління даними в різноманітних додатках, від невеликих проєктів до великих корпоративних систем. Ця СУБД забезпечує високу продуктивність, надійність і масштабованість, що робить її ідеальним вибором для багатьох веб-додатків.

phpMyAdmin – це веб-застосунок з відкритим вихідним кодом, який використовується для адміністрування MySQL через веб-інтерфейс. phpMyAdmin дозволяє виконувати різноманітні операції з базами даних. Використання phpMyAdmin значно спрощує процес управління базами даних MySQL і підвищує ефективність роботи адміністраторів.

XAMPP – це повний пакет для розробки, який включає веб-сервер Apache, СУБД MySQL (або MariaDB), інтерпретатори для мов PHP і Perl, а також інші корисні інструменти. XAMPP є крос-платформеним рішенням, що дозволяє розробникам швидко налаштовувати локальне середовище для розробки веб-додатків. Завдяки XAMPP, розробники можуть легко встановлювати та налаштовувати необхідні компоненти, щоб мати повноцінне середовище для розробки на своїх комп'ютерах. Пакет містить зручні панелі керування для запуску та зупинки серверів.

Вибір цих програмних засобів для реалізації системи базувався на їхній здатності забезпечити високу продуктивність, гнучкість та зручність у використанні. MySQL надає потужні інструменти для управління даними, phpMyAdmin спрощує процес адміністрування бази даних, а XAMPP забезпечує легке налаштування локального середовища для розробки. Разом ці технології створюють надійну та ефективну основу для розробки і підтримки сучасних веб-додатків.

3.1.2. Реалізація програмного забезпечення

Перед роботою з відеопотоком необхідно завантажити моделі для розпізнавання облич. Цю задачу вирішує наступний блок коду:

```
const video = document.getElementById('video');

Promise.all([
  faceapi.nets.tinyFaceDetector.loadFromUri('/models'),
  faceapi.nets.faceLandmark68Net.loadFromUri('/models'),
  faceapi.nets.faceRecognitionNet.loadFromUri('/models')
  ]).then(startVideo).catch(err => {
  console.error("Model loading failed:", err);
});
```

Даний код ініціалізує відеоелемент і завантажує моделі для розпізнавання облич. Promise.all використовується для одночасного завантаження кількох моделей з директорії /models. Коли всі моделі завантажено, викликається функція startVideo, яка починає відеопотік. Якщо виникає помилка під час завантаження моделей, вона виводиться в консоль.

Функція початку відеопотоку startVideo описана наступним кодом:

```
function startVideo() {
  navigator.mediaDevices.getUserMedia({ video: {} })
    .then(stream => {
      video.srcObject = stream;
      video.play();
    })
    .catch(err => {
      console.error("Error accessing the webcam: ", err);
      alert("Error accessing the webcam: " + err.message);
    });
}
```

Дана функція використовує метод navigator.mediaDevices.getUserMedia для доступу до веб-камери користувача. Якщо доступ до камери дозволено, відеопотік встановлюється як джерело для відеоелемента і починає відтворення. У разі помилки, повідомлення про помилку виводиться в консоль і показується повідомлення (алерт) з описом помилки.

Обробник подій завантаження даних з відео описаний наступним кодом, що додає обробник подій для події loadeddata відеоелемента. Коли відео завантажено, створюється область для виводу (канвас), яка додається до документа. Розміри області встановлюються відповідно до розмірів відео. Метод setInterval

використовується для періодичного виклику функції, яка здійснює детекцію облич, визначає їхні точки контурів (лендмарки) кожні 100 мс. DetectAllFaces повертає масив виявлених облич:

```
video.addEventListener('loadeddata', () => {
  const canvas = faceapi.createCanvasFromMedia(video);
  document.body.append(canvas);
  const displaySize = { width: video.width, height: video.height };
  faceapi.matchDimensions(canvas, displaySize);

  setInterval(async () => {
    const detections = await faceapi.detectAllFaces(video,
        new faceapi.TinyFaceDetectorOptions()).withFaceLandmarks().withFaceDescriptors();
    const resizedDetections = faceapi.resizeResults(detections, displaySize);
    const context = canvas.getContext('2d');
    context.clearRect(0, 0, canvas.width, canvas.height);
    faceapi.draw.drawDetections(canvas, resizedDetections);
    faceapi.draw.drawFaceLandmarks(canvas, resizedDetections);
```

Визначення та обробка обличчя що проходить тест описані наступним кодом:

```
if (resizedDetections.length > 0) {
  if (!registeredFaceDescriptor) {
    registeredFaceDescriptor = resizedDetections[0].descriptor;
    console.log('Registered face descriptor:', registeredFaceDescriptor);
  }

  const distances = resizedDetections.map(d => faceapi.euclideanDistance(
      d.descriptor, registeredFaceDescriptor));
  const mainFaceIndex = distances.indexOf(Math.min(...distances));
  const mainFace = resizedDetections[mainFaceIndex];
  const landmarks = mainFace.landmarks;
  const leftEye = landmarks.getLeftEye();
  const rightEye = landmarks.getRightEye();
  const mouth = landmarks.getMouth();
```

Цей блок коду перевіряє, чи було знайдено обличчя. Якщо зареєстроване обличчя ще не визначено, воно встановлюється як перше знайдене обличчя (зберігається його дескриптор: лендмарки (точки очей і рота). Дане обличчя, також відповідає збереженому в базі при реєстрації обличчю. Далі для кожного знайденого обличчя обчислюються відстані точок відповідно до зареєстрованого обличчя, щоб визначити де основне обличчя а де стороннє.

Код для виявлення руху очей і рота перевіряє наявність попередніх точок (лендмарки, для перевірки руху). Викликаються функції для виявлення руху зіниць та рота. Якщо рух виявлено, навколо очей або рота рисується червоний

прямокутник та відображається відповідне повідомлення. Поточні лендмарки зберігаються для порівняння з наступним кадром :

```
if (previousLandmarks) {
  const leftPupilMovement = detectPupilMovement(previousLandmarks.leftEye, leftEye);
  const rightPupilMovement = detectPupilMovement(previousLandmarks.rightEye, rightEye);
  const mouthMovement = detectMouthMovement(mouth);

  if (leftPupilMovement || rightPupilMovement) {
    drawRectangleAroundEyes(context, leftEye, rightEye);
    displayMessage(context, 'Eye Movement', displaySize.width / 2, 50);
  }

  if (mouthMovement) {
    drawRectangleAroundMouth(context, mouth);
    displayMessage(context, 'Lips Movement', displaySize.width / 2, 100);
  }
}

previousLandmarks = {
  leftEye,
  rightEye,
  mouth
};
```

Виявлення та обробка додаткових осіб здійснюється за допомогою наступного коду:

```
for (let i = 0; i < resizedDetections.length; i++) {
  if (i !== mainFaceIndex) {
    const additionalFace = resizedDetections[i].detection.box;
    drawRectangleAroundFace(context, additionalFace);
    displayMessage(context, 'Additional person', additionalFace.x +
       additionalFace.width / 2, additionalFace.y - 10);
  }
}
```

Цей блок коду обробляє додаткові обличчя, які з'являються в кадрі. Для кожного обличчя, яке не є основним, малюється червоний прямокутник навколо обличчя та відображається повідомлення "Additional person".

Функції для виявлення руху представлені за допомогою наступного коду:

```
function detectPupilMovement(previousEye, currentEye) {
  const previousCenter = getPupilCenter(previousEye);
  const currentCenter = getPupilCenter(currentEye);
  const dx = Math.abs(previousCenter.x - currentCenter.x);
  const movementThreshold = 3; // Threshold for significant movement

  return dx > movementThreshold;
}
```

```
function getPupilCenter(eyeLandmarks) {
  return {
    x: (eyeLandmarks[0].x + eyeLandmarks[3].x) / 2,
    y: (eyeLandmarks[1].y + eyeLandmarks[5].y) / 2
  };
}

function detectMouthMovement(mouth) {
  const upperLip = mouth[13]; // Середня точка верхньої губи
  const lowerLip = mouth[14]; // Середня точка нижньої губи
  const currentDistance = Math.abs(upperLip.y - lowerLip.y);

  if (currentDistance < minMouthDistance) {
    minMouthDistance = currentDistance;
  }

  const movementThreshold = 1.5 * minMouthDistance;
  // 50% збільшення від мінімальної відстані

  return currentDistance > movementThreshold;
}
```

Ці функції визначають рухи зіниць і рота. detectPupilMovement обчислює рух зіниць, визначаючи відстань між поточним і попереднім положенням центру ока. detectMouthMovement визначає рух рота, вимірюючи відстань між верхньою та нижньою губами і порівнюючи її з мінімальною зареєстрованою відстанню.

Функції для виводу прямокутників і повідомлень представлені за допомогою наступного коду:

```
function drawRectangleAroundEyes(context, leftEye, rightEye) {
  const allEyePoints = leftEye.concat(rightEye);
  const eyeBox = getBoundingBox(allEyePoints);

  const x = eyeBox.x;
  const y = eyeBox.y - eyeBox.height / 2; // Центрування прямокутника вертикальнo
  const width = eyeBox.width;
  const height = eyeBox.height * 2; // Подвоєння висоти очей

  context.strokeStyle = 'red';
  context.lineWidth = 2;
  context.strokeRect(x, y, width, height);
}
```

```
function drawRectangleAroundMouth(context, mouth) {
  const mouthBox = getBoundingBox(mouth);

  const x = mouthBox.x;
  const y = mouthBox.y - mouthBox.height / 2;
                    // Центрування прямокутника вертикально
  const width = mouthBox.width;
  const height = mouthBox.height * 2; // Подвоєння висоти рота

  context.strokeStyle = 'red';
  context.lineWidth = 2;
  context.strokeRect(x, y, width, height);
}

function drawRectangleAroundFace(context, box) {
  context.strokeStyle = 'red';
  context.lineWidth = 2;
  context.strokeRect(box.x, box.y, box.width, box.height);
}
```

Ці функції зобржають червоні прямокутники навколо очей, рота та обличчя. drawRectangleAroundEyes і drawRectangleAroundMouth визначають прямокутники навколо відповідних частин обличчя, а drawRectangleAroundFace рисує прямокутник навколо всього обличчя.

Допоміжні функції обчислення розмірів та виводу тексту описані наступним кодом:

```
function getBoundingBox(points) {
  const x = Math.min(...points.map(point => point.x));
  const y = Math.min(...points.map(point => point.y));
  const width = Math.max(...points.map(point => point.x)) - x;
  const height = Math.max(...points.map(point => point.y)) - y;
  return { x, y, width, height };
}

function displayMessage(context, message, x, y) {
  context.fillStyle = 'red';
  context.font = '20px Arial';
  context.textAlign = 'center';
  context.fillText(message, x, y);
}
```

Ці допоміжні функції допомагають обчислювати габаритний прямокутник для набору точок і відображати повідомлення на канвасі. getBoundingBox обчислює

координати і розміри прямокутника, що охоплює всі точки. displayMessage відображає повідомлення на канвасі в заданих координатах.

Моніторинг звуку забезпечується за допомогою наступного коду

```
startSoundMonitorButton.addEventListener('click', () => {
  navigator.mediaDevices.getUserMedia({ audio: true })
    .then(stream => {
      const audioContext = new (window.AudioContext || window.webkitAudioContext)();
      const mediaStreamSource = audioContext.createMediaStreamSource(stream);
      const analyser = audioContext.createAnalyser();
      mediaStreamSource.connect(analyser);
      analyser.fftSize = 256;
      const dataArray = new Uint8Array(analyser.frequencyBinCount);

      function monitorSound() {
        analyser.getByteFrequencyData(dataArray);
        const average = dataArray.reduce((a, b) => a + b) / dataArray.length;

        if (average > 50) { // Threshold for detecting sound
          displayMessageOnCanvas('Sound Detected', canvas, context, displaySize.width / 2, 150);
        }

        requestAnimationFrame(monitorSound);
      }

      monitorSound();
    })
    .catch(err => {
      console.error("Error accessing the microphone: ", err);
      alert("Error accessing the microphone: " + err.message);
    });
});
```

Даний код отримує доступ до мікрофона користувача. Це здійснюється за допомогою navigator.mediaDevices.getUserMedia, який запитує у користувача дозвіл на доступ до аудіопристроїв. Якщо дозвіл отримано, створюється аудіопотік. Після отримання аудіопотоку, створюється аудіоконтекст (AudioContext), який є основою для роботи з аудіосигналами. Використовуючи аудіоконтекст, ми створюємо джерело медіапотоку (MediaStreamSource) з нашого аудіопотоку. Далі, для аналізу аудіосигналів у реальному часі, створюється аналізатор (AnalyserNode).

Для виявлення звуку аналізуються частотні дані у реальному часі. за допомогою методу fftSize, який встановлює розмір FFT (Fast Fourier Transform) для аналізу частотного спектру. Значення FFT визначає точність аналізу частотного спектру. Використовуючи аналізатор, ми можемо отримати частотні дані аудіосигналу за допомогою методу getByteFrequencyData, який заповнює масив Uint8Array частотними даними. Відбирається середнє значення частотних даних і порівнюється з визначеним пороговим значенням. Якщо середнє значення перевищує поріг, це означає, що був виявлений звук, і система реагує відповідним

чином. Коли звук виявлено, на канвасі відображається повідомлення "Sound Detected". Це робиться за допомогою функції displayMessageOnCanvas, яка виводить текстове повідомлення на канвас в заданій позиції.

3.2. Інтерфейс користувача

На рисунку 3.1 показано головну сторінку з назвою проекту. Оскільки в проекті використовуються різні моделі для виявлення і розпізнавання, тому браузер має мати доступ до файлів на локальному комп'ютері користувача, що забезпечується використанням локального сервера:

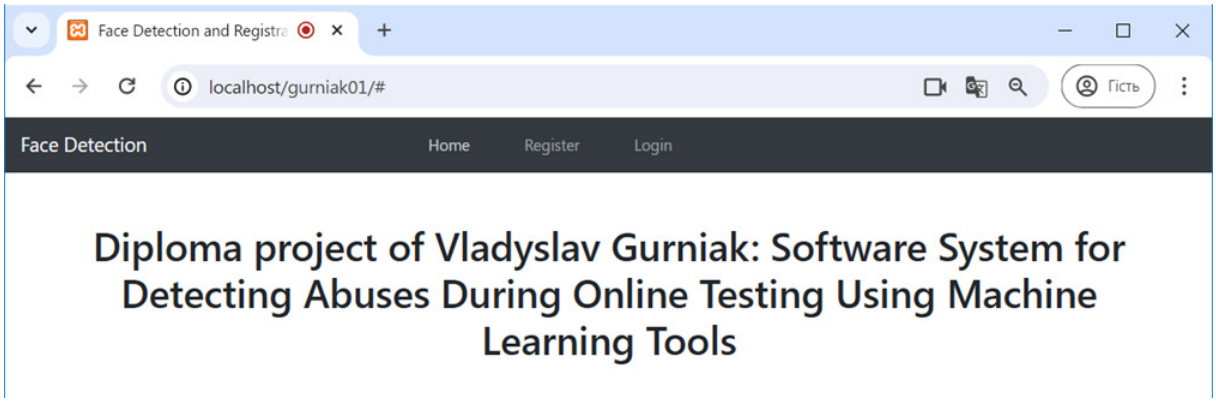


Рисунок 3.1 – Головна сторінка з назвою проекту

На рисунку 3.2 показана сторінка реєстрації користувача, де можна ввести персональні дані та зробити знімок обличчя для ідентифікації. Робота даної сторінки забезпечується наступним кодом:

```
function loadPage(page) {
  const content = document.getElementById('content');

  switch (page) {
    case 'home':
      content.innerHTML = '<h1>Welcome to the Home Page</h1>';
      break;
    case 'login':
      content.innerHTML = `
        <h1>Login</h1>
        <form id="loginForm">
          <label for="email">Email:</label>
          <input type="email" id="email" name="email"><br>
          <label for="password">Password:</label>
          <input type="password" id="password" name="password"><br>
          <button type="button" onclick="login()">Login</button>
        </form>`;
      break;
```

```
    case 'register':
      content.innerHTML = `
        <h1>Register</h1>
        <form id="registerForm">
          <label for="firstName">First Name:</label>
          <input type="text" id="firstName" name="firstName"><br>
          <label for="lastName">Last Name:</label>
          <input type="text" id="lastName" name="lastName"><br>
          <label for="email">Email:</label>
          <input type="email" id="email" name="email"><br>
          <label for="password">Password:</label>
          <input type="password" id="password" name="password"><br>
          <label for="photo">Photo:</label>
          <input type="file" id="photo" name="photo" accept="image/*" capture="user"><br>
          <button type="button" onclick="register()">Register</button>
        </form>
        <video id="video" width="720" height="560" autoplay muted></video>`;
      setupCamera();
      break;
    case 'test':
      content.innerHTML = '<h1>Take Test</h1>';
      // Include test logic here
      break;
  }
}
```

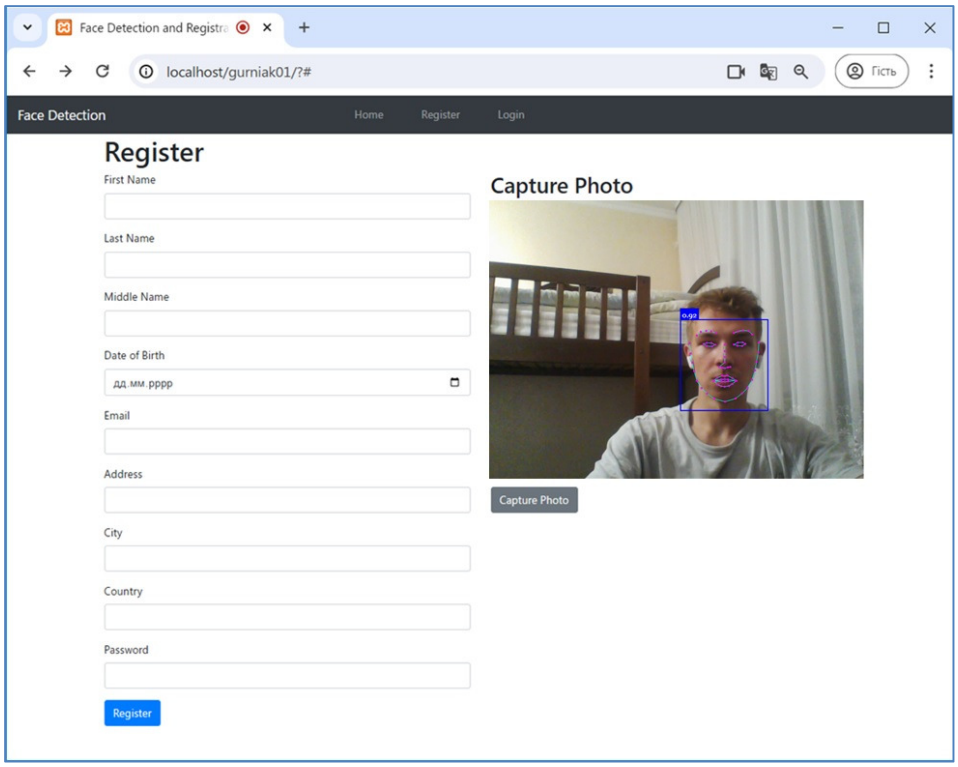


Рисунок 3.2 – сторінка реєстрації нового користувача

Функція setupCamera налаштовує відеопотік з веб-камери та інтегрує його з бібліотекою face-api.js для виявлення обличчя та визначення лендмарків. Спочатку вона отримує відеоелемент з HTML-документа. Потім, за допомогою navigator.mediaDevices.getUserMedia, запитує доступ до відеопотоку з веб-камери. Якщо доступ отримано, відеопотік встановлюється як джерело для відеоелемента.

```
function setupCamera() {
  const video = document.getElementById('video');

  navigator.mediaDevices.getUserMedia({ video: {} })
    .then(stream => {
      video.srcObject = stream;
    })
    .catch(err => console.error(err));
```

Після початку відтворення відео завантажуються моделі face-api.js для виявлення облич та лендмарків з URL /models. Встановлюються розміри відображення відео, і кожні 100 мілісекунд проводиться аналіз кадру. Виявляються обличчя та лендмарки, результати масштабуються відповідно до розмірів відео, і створюється канвас для відображення результатів. Якщо обличчя виявлено, малюються рамки навколо обличчя та лендмарки.

```
 video.addEventListener('play', async () => {
   await faceapi.nets.tinyFaceDetector.loadFromUri('/models');
   await faceapi.nets.faceLandmark68Net.loadFromUri('/models');

   const displaySize = { width: video.width, height: video.height };
   faceapi.matchDimensions(video, displaySize);

   setInterval(async () => {
     const detections = await faceapi.detectAllFaces(video, new faceapi.TinyFaceDetectorOptions())
       .withFaceLandmarks();
     const resizedDetections = faceapi.resizeResults(detections, displaySize);


       if (resizedDetections.length > 0) {
         const canvas = document.createElement('canvas');
         canvas.width = video.width;
         canvas.height = video.height;
         document.body.append(canvas);

         faceapi.draw.drawDetections(canvas, resizedDetections);
         faceapi.draw.drawFaceLandmarks(canvas, resizedDetections);

         const landmarks = resizedDetections[0].landmarks;
         const points = landmarks.positions.map(p => ({ x: p.x, y: p.y }));
         savePhotoWithLandmarks(video, points);
       }
     }, 100);
   });
}
```

Функція savePhotoWithLandmarks створює канвас, на якому відображається поточне зображення з відео. Використання const dataUrl = canvas.toDataURL('image/png'); дозволяє перетворити зображення, що зберігається на канвасі, у формат Data URL.

```
function savePhotoWithLandmarks(video, landmarks) {
  const canvas = document.createElement('canvas');
  canvas.width = video.videoWidth;
  canvas.height = video.videoHeight;
  const context = canvas.getContext('2d');
  context.drawImage(video, 0, 0, canvas.width, canvas.height);

  const dataUrl = canvas.toDataURL('image/png');


// Send dataUrl and landmarks to the server
fetch('/save_photo', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
  },
  body: JSON.stringify({ photo: dataUrl, landmarks }),
}).then(response => response.json())
  .then(data => console.log(data))
  .catch(err => console.error('Error:', err));
}
```

Data URL є текстовим представленням зображення у форматі PNG, закодованим у Base64. Це дозволяє легко передавати зображення у текстовому форматі через мережу. Для відправки цього зображення разом з лендмарками на сервер використовується функція fetch. Вона відправляє POST-запит на URL /save_photo. У тілі запиту міститься JSON-об'єкт, який складається з двох ключів: photo, що містить закодоване зображення, та landmarks, що містить координати лендмарків обличчя. На рисунку 3.3 представлено вікно входу в систему для зареєстрованих користувачів, де користувач вводить електронну пошту та пароль.

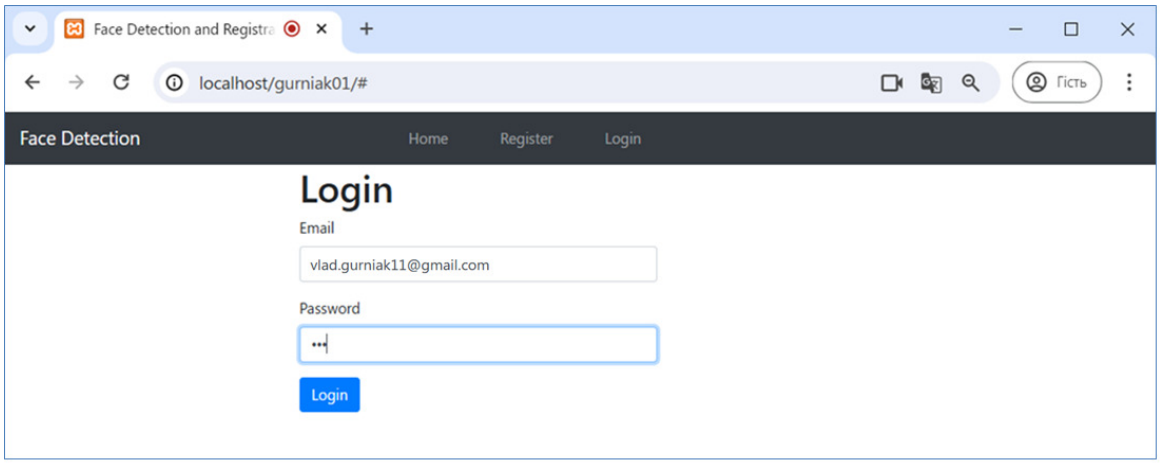


Рисунок 3.3 – Вікно входу в систему

Ініціалізуюче перед тестуванням вікно (до нтиснення кнопки «Start test», Рис. 3.4) відображає зареєстровану особу з його лендмарк точками.

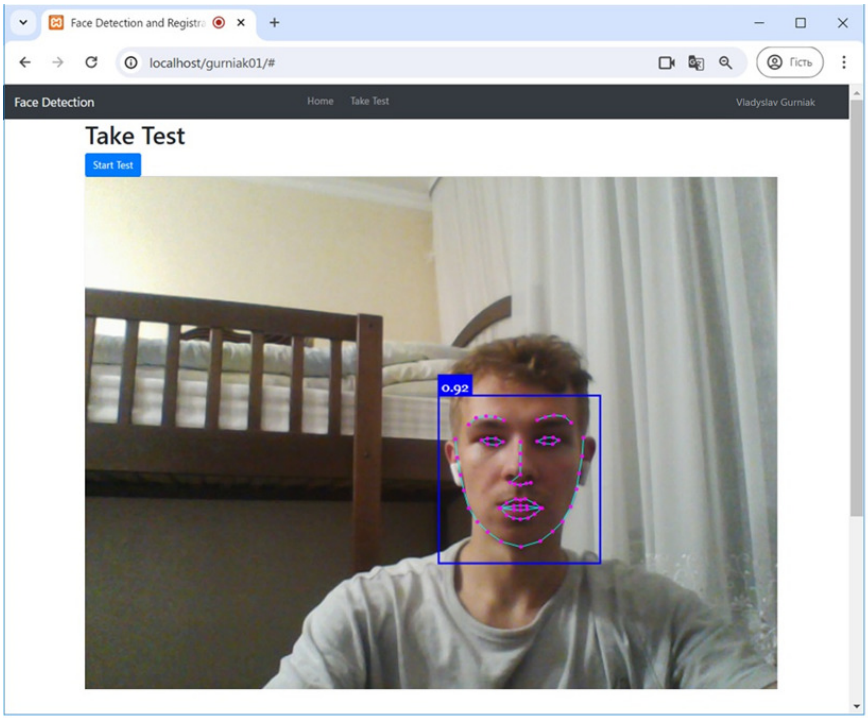


Рисунок 3.4 – Відображення зареєстрованої особи

Після натиснення кнопки «Start test» відбувається отримання відеопотоку з веб-камери і завантажуються модулі бібліотеки face-api.js для виявлення облич і визначення лендмарків у режимі реального часу. Спочатку отримується відеоелемент з HTML-документа і завантажуються моделі для виявлення облич і лендмарків їз зазначеного шляху. Після отримання доступу до веб-камери, відеопотік встановлюється як джерело для відеоелемента. Коли відео починає відтворюватися, створюється область для виводу (канвас) для відображення результатів і кожні 100 мілісекунд аналізуються кадри відео для виявлення облич. Якщо обличчя виявлено, його межі і відображаються на канвасі. Також визначається рух очей і рота, що відображається відповідними повідомленнями на екрані. Крім того, якщо в кадрі з'являється додаткова особа, або інша ніж зареєстрована, її обличчя окреслюється червоним прямокутником і відображається повідомлення "Additional person".

Також в даному вікні визначаєтья рух очей і рота. Якщо рух виявлено, відповідні області окреслюються червоними прямокутниками і відображаються повідомлення на екрані. Відповідні області обличчя виділяються червоними

прямокутниками. Визначаються межі обличчя за координатами лендмарків та відображає повідомлення на екрані.

3.3. Тестування програмного забезпечення

В таблиці 1 представлено сценарії перевірки основних функцій системи. Такий підхід дозволяє детально оцінити відповідність системи її функціональним вимогам [38-40].

Таблиця 1. Функціональне тестування системи

№	Функція	Сценарії виконання	Очікуваний результат	Фактичний результат
1	Ідентифікація користувача	Завантажити відеопотік, ввести правильні ім'я користувача та пароль, показати обличчя	Користувач має бути розпізнаний та допущений до тесту	Користувач розпізнаний та допущений до тесту
2	Виявлення обличчя	Завантажити відеопотік, звернути увагу на обличчя	Обличчя має бути виділене прямокутником	Обличчя виділене прямокутником
3	Виявлення додаткових осіб	Завантажити відеопотік, додати іншу особу в кадр	Додаткова особа має бути виділена червоним прямокутником з повідомленням "Additional person"	Додаткова особа виділена червоним прямокутником з повідомленням "Additional person"
4	Виявлення руху очей	Звернути увагу на обличчя, рухати очима вліво/вправо	Область очей має бути виділена червоним прямокутником з повідомленням "Eye Movement"	Область очей виділена червоним прямокутником з повідомленням "Eye Movement"
5	Виявлення руху рота	Звернути увагу на обличчя, відкрити/закрити рот	Область рота має бути виділена червоним прямокутником з повідомленням "Lips Movement"	Область рота виділена червоним прямокутником з повідомленням "Lips Movement"

На рисунку 3.5 зображено вікно роботи системи з ідентифікацією користувача, на ньому показано відеопотік, де користувач натискає «Start Test» і

система перевіряє його зображення з наявним у базі. Цей рисунок демонструє успішну ідентифікацію користувача та надання доступу до подальших дій.

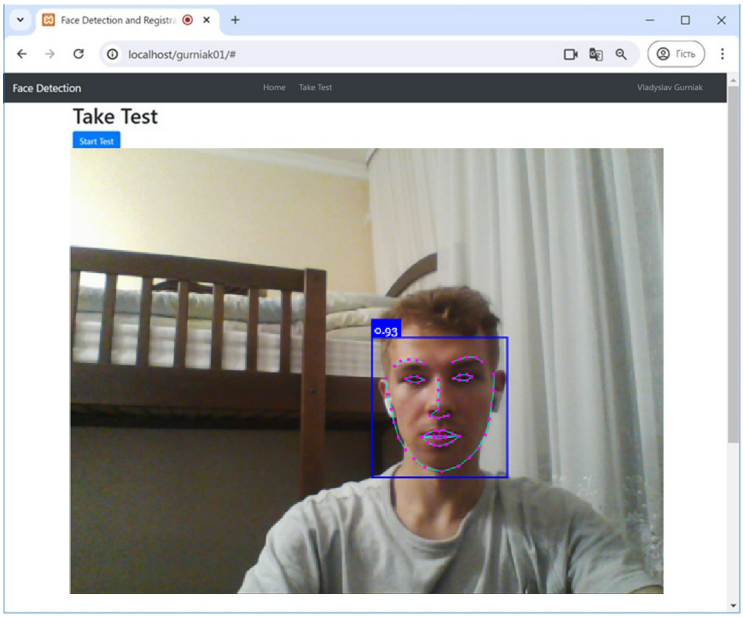


Рисунок 3.5 – Ідентифікація користувача

На рисунку 3.6 зображено вікно роботи системи з виявленням обличчя, на відеопотоці автоматично виділяється обличчя користувача прямокутником. Цей прямокутник вказує на те, що система успішно розпізнала обличчя і відслідковує його положення в кадрі.

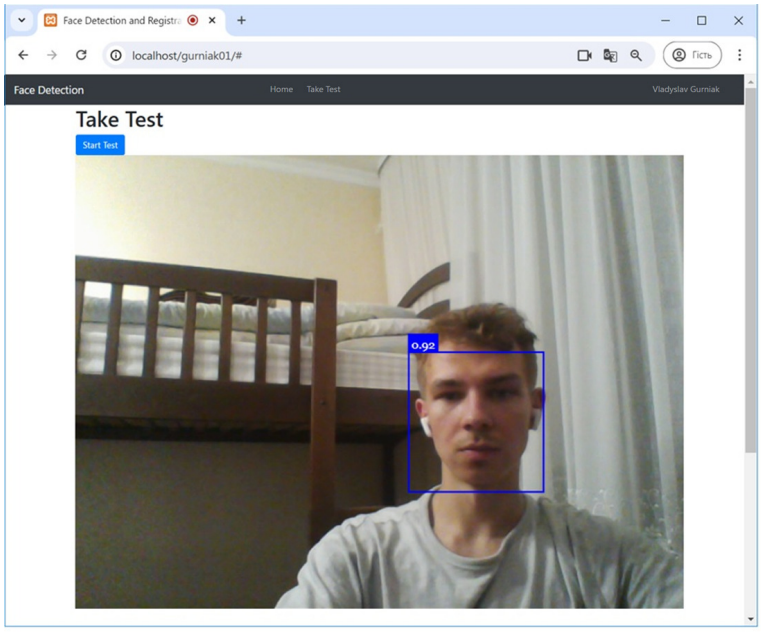


Рисунок 3.6 – Виявлення обличчя

На рисунку 3.7 зображено вікно роботи системи з виявленням додаткових осіб, на ньому зображено відеопотік з двома особами в кадрі. Обличчя основного користувача виділено стандартним прямокутником, а обличчя додаткової особи окреслене червоним прямокутником з повідомленням "Additional person". Це вказує на те, що система успішно виявила присутність іншої особи.

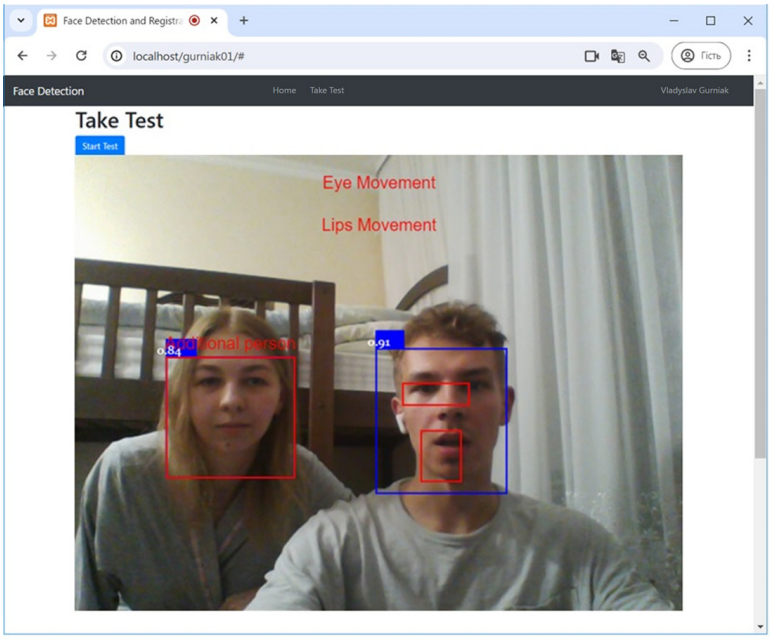


Рисунок 3.7 – Виявлення облич додаткових осіб

На рисунку 3.8 зображено вікно роботи системи з виявленням руху очей, на ньому показано відеопотік, в якому система відстежує рух очей користувача. Коли користувач рухає очима вліво або вправо, область очей виділяється червоним прямокутником з повідомленням "Eye Movement". Це демонструє успішне виявлення руху очей.

На рисунку 3.9 зображено вікно роботи системи з виявленням руху рота, а На рисунку 3.10 зображено вікно роботи системи з виявленням руху рота і очей одночасно, де система фіксує рухи рота і очей користувача. Коли користувач відкриває рот тавідводиьт погляд від екрану, область рота виділяється червоним прямокутником з повідомленням "Lips Movement", область очей виділяється червоним прямокутником з повідомленням "Eye Movement". Це показує, що система успішно розпізнає рухи рота та очей.

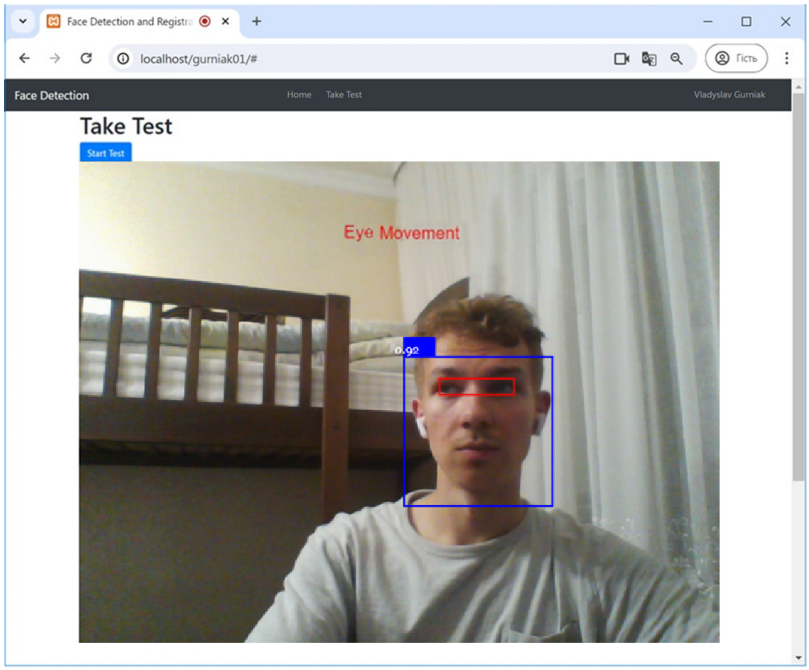


Рисунок 3.8 – Виявлення руху очей

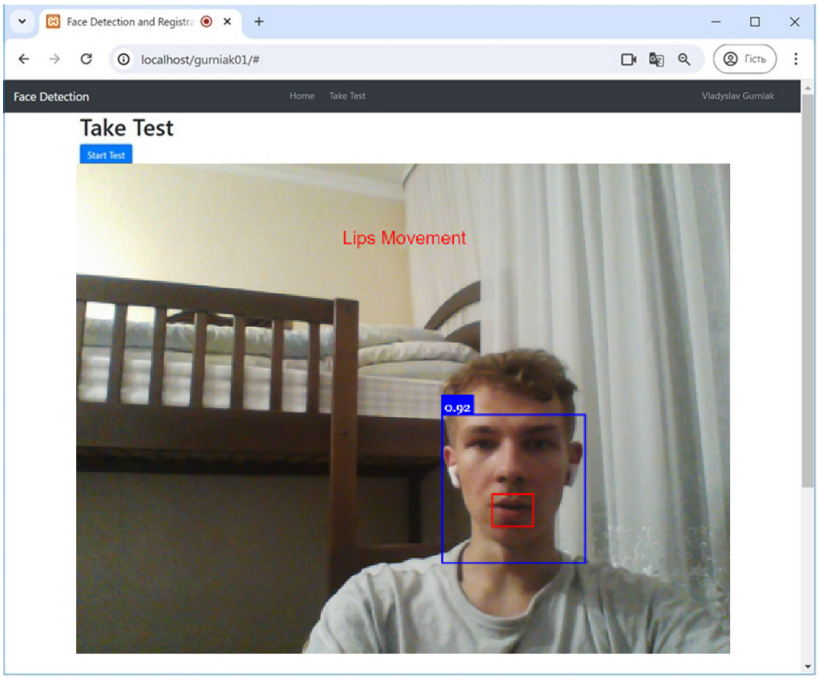


Рисунок 3.9 – Виявлення руху рота

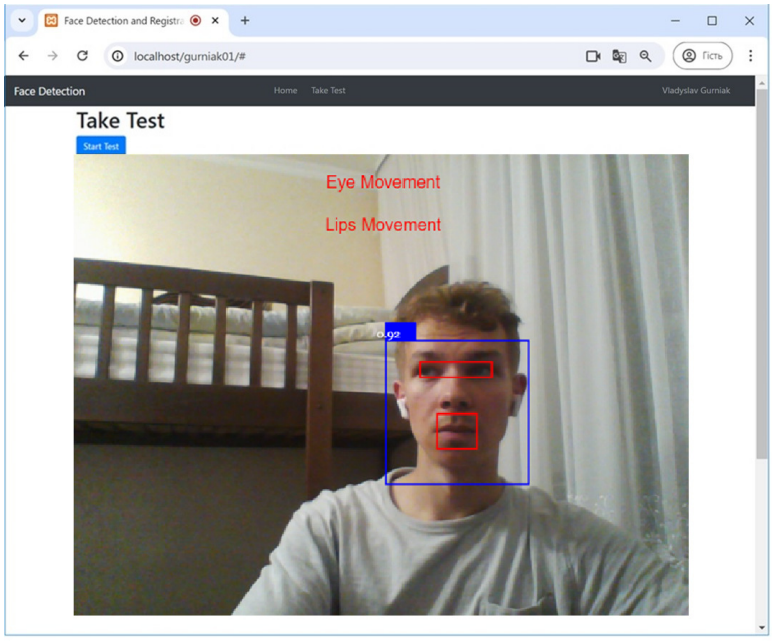


Рисунок 3.10 – Виявлення руху рота і очей

ВИСНОВКИ

Метою даної кваліфікаційної роботи була розробка програмної системи для виявлення зловживань під час онлайн-тестувань за допомогою методів машинного навчання. Система підвищує рівень академічної чесності та забезпечує справедливість оцінювання знань студентів.

Під час виконання кваліфікаційної роботи було вирішено наступні завдання:

1. Проведено аналіз предметної області, що дозволило зрозуміти ключові аспекти та вимоги до системи.
2. Здійснено огляд та аналіз існуючих аналогів, що допомогло визначити сильні та слабкі сторони поточних рішень.
3. Визначено основні функціональні вимоги до системи, що забезпечило чітке уявлення про необхідну функціональність.
4. Розроблено основні сценарії роботи системи, що сприяло створенню інтуїтивно зрозумілого користувацького досвіду.
5. Створено унікальний та зрозумілий дизайн інтерфейсу, який забезпечує зручність та ефективність користування системою.
6. Розроблено функціональну веб-базовану систему, яка ефективно виявляє зловживання під час онлайн-тестувань, забезпечуючи академічну чесність та справедливість оцінювання.

Запропонована програмна система виявлення зловживань під час онлайн-тестувань засобами машинного навчання дозволяє покращити процес оцінки знань, забезпечуючи справедливість і чесність оцінювання студентів. Система використовує методи обробки зображень, відео та звуку з відеокамери, мікрофону та копії екрану, аналізуючи рухи голови, очей, губ та інші фактори, що можуть свідчити про зловживання.

Запропонована система демонструє стабільну роботу, високу продуктивність та надійність, значно покращуючи процес оцінки знань, забезпечуючи справедливість і чесність оцінювання студентів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ЮНЕСКО. Повідомлення Генерального директора ЮНЕСКО про глобальну коаліцію в галузі освіти [Електронний ресурс] / ЮНЕСКО. – Режим доступу: https://www.unesco.org/

2. Всесвітній економічний форум. Зростання онлайн-навчання під час пандемії COVID-19 [Електронний ресурс] / Всесвітній економічний форум. – Режим доступу: https://www.weforum.org/agenda/2020/04/coronavirus-education-global-covid19-online-digital-learning/

3. Lee, K., Fanguy, M., Bligh, B., & Lu, S. (2021). Adoption of online teaching during the COVID- 19 Pandemic: A systematic analysis of changes in university teaching activity. Educational Review. https://doi.org/10.1080/00131 911.2021.1978401

4. Lee, K., Fanguy, M., Lu, X. S., & Bligh, B. (2021). Student learning during COVID- 19: It was not as bad as we feared. Distance Education, 42(1), 1–8. https://doi.org/10.1080/01587919.2020.1869529

5. Kyungmee Lee, Mik Fanguy. Online exam proctoring technologies: Educational innovation or deterioration? D OI : 10.1111/bjet.1318 2

6. Безпечний іспитовий браузер, 2020 p. Safe Exam browser https://safee xambr owser.org/about_overv iew_en.html

7. Yang, S. C., Huang, C. L., & Chen, A. S. (2013). An investigation of college students' perceptions of academic dishonesty, reasons for dishonesty, achievement goals, and willingness to report dishonest behavior. Ethics & Behavior, 23(6), 501–522. https://doi.org/10.1080/10508422.2013.802651

8. Becker, D., Connolly, J., Lentz, P., & Morrison, J. (2006). Using the business fraud triangle to predict academic dishonesty among business students. Academy of Educational Leadership Journal, 10(1), 37– 54. https:// w ww.proqu est.com/schol arly-journ als/using - busin ess- fraud - trian gle- predi ct- acade mic/docvi ew/21423 2023/se-2?accountid=27828

9. Finchilescu, G., & Cooper, A. (2018). Perceptions of academic dishonesty in a South African univer¬sity: A Q- methodology approach. Ethics & Behavior, 28(4), 284–301. https://doi.org/10.1080/10508 422.2017.1279972

10. Day, N. E., Hudson, D., Dobies, P. R., & Waris, R. (2011). Student or situation? Personality and classroom context as predictors of attitudes about business school cheating. Social Psychology of Education, 14(2), 261–282. https://doi.org/10.1007/s11218-010-9145-8

11. Pulfrey, C. J., Vansteenkiste, M., & Michou, A. (2019). Under pressure to achieve? The impact of type and style of task instructions on student cheating. Frontiers in Psychology, 10, 1– 18. https://doi.org/10.3389/ fpsyg.2019.01624

12. Finchilescu, G., & Cooper, A. (2018). Perceptions of academic dishonesty in a South African univer¬sity: A Q- methodology approach. Ethics & Behavior, 28(4), 284– 301. https://doi.org/10.1080/10508 422.2017.1279972

13. Peled, Y., Eshet, Y., Barczyk, C., & Grinautski, K. (2019). Predictors of Academic Dishonesty among under-graduate students in online and face- to- face courses. Computers & Education, 131, 49– 59. https://doi. org/10.1016/j.compedu.2018.05.012

14. Chudzicka- Czupała, A., Grabowski, D., Mello, A. L., Kuntz, J., Zaharia, D. V., Hapon, N., Lupina- Wegener, A., & Börü, D. (2016). Application of the theory of planned behavior in academic cheating research– cross- cultural comparison. Ethics & Behavior, 26(8), 638–659. https://doi.org/10.1080/10508422.2015.1112745Lonsdale, 2017

15. O'Reilly, A., & Creagh, T. Developing an online examination system using rule-based and algorithmic processing. In Proceedings of the 2015 International Conference on Learning and Teaching in Computing and Engineering (pp. 134-139). DOI: 10.1145/2839509.2839526

16. Hylton K. Online Remote Proctoring: A New Era in Testing // Journal of Higher Education Theory and Practice. 2016. Т. 16. № 3. С. 40-46.

17. Офіційний веб-сайт ProctorU. URL: https://www.proctoru.com .

18. Watson G., Sottile J. Cheating in the Digital Age: Do Students Cheat More in Online Courses? // Online Journal of Distance Learning Administration. 2010. Т. 13. № 1. С. 1-10.

19. Respondus. Офіційний веб-сайт Respondus. URL: https://www.respondus.com/products/monitor .

20. Brouwer, N. & Heck, Andre & Smit, Guusje. (2017). Proctoring to improve teaching practice. MSOR Connections. 15. 25. 10.21100/msor.v15i2.414.

21. ProctorExam. Офіційний веб-сайт ProctorExam. URL: https://www.proctorexam.com

22. O'Reilly M., Creagh R. The Use of Online Proctoring in High-Stakes Exams // Journal of Educational Technology Systems. 2015. Т. 44. № 1. С. 21-39.

23. Examity. Офіційний веб-сайт Examity. URL: https://www.examity.com .

24. Rowe N. Cheating in Online Student Assessment: Beyond Plagiarism // Online Journal of Distance Learning Administration. 2004. Т. 7. № 2. С. 1-10.

25. HonorLock. Офіційний веб-сайт HonorLock. URL: https://honorlock.com

26. Lang J. M. Cheating Lessons: Learning from Academic Dishonesty. Cambridge, MA: Harvard University Press, 2013. 304 с.

27. ExamSoft. Офіційний веб-сайт ExamSoft. URL: https://examsoft.com .

28. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. [Електронний ресурс]. Режим доступу: https://arxiv.org/abs/1512.02325 (дата звернення: 06.06.2024). - С. 1-14.

29. Набір даних WIDERFACE (http://mmlab.ie.cuhk.edu.hk/projects/WIDERFace/)

30. Yang, S., Luo, P., Loy, C. C., & Tang, X. (2016). WIDER FACE: A Face Detection Benchmark. [Електронний ресурс]. Режим доступу: https://arxiv.org/abs/1511.06523 (дата звернення: 06.06.2024). - С. 1-9.

31. Бібліотека dlib [Електронний ресурс] – Режим доступу: http://dlib.net/

32. Бібліотека OpenCV. Open Source Computer Vision Library [Електронний ресурс]. Режим доступу: https://opencv.org/

33. Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. [Електронний ресурс]. Режим доступу: https://arxiv.org/abs/1704.04861 (дата звернення: 06.06.2024). - С. 1-15.

34. HTML Introduction [Електронний ресурс]. – URL: https://www.w3schools.com/html/html_intro.asp (дата звернення: 20.05.2024)

35. Visual Studio Code Documentation [Електронний ресурс]. – URL: https://code.visualstudio.com/docs (дата звернення: 20.05.2024)

36. Web JavaScript Documentation [Електронний ресурс]. – URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення:

37. face-api.js - API розпізнавання облич на JavaScript для браузера та node.js, реалізованих поверх ядра tensorflow.js» (https://github.com/justadudewhohacks/face-api.js)

38. Software Testing: Functional Testing [Електронний ресурс]. – URL: https://www.geeksforgeeks.org/software-testing-functional-testing/ (дата звернення: 20.05.2024)

39. Гурняк В. І., Гриньків А. М. Програмна система виявлення зловживань під час онлайн-тестувань засобами машинного навчання // Перспективні напрямки розвитку економіки, обліку, управління та права: теорія і практика: матеріали міжнар. наук.-практ. конф., м. Ізмаїл, 18 травня 2024 р. Секція: Математичні методи, моделі та інформаційні технології в економіці. Ізмаїл: ЦФЕНД, 2024. С. 27-28.

40. Методичні рекомендації до виконання кваліфікаційної роботи / [Комар М. П., Саченко А. О., Васильків Н. М. та ін.]. – Тернопіль: ЗУНУ, 2024. – 52 с.

ДОДАТОК А

Програмний код

Модуль головної сторінки

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1,
shrink-to-fit=no">
    <title>Face Detection and Registration</title>
    <link
href="https://stackpath.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.
css" rel="stylesheet">
    <link href="styles.css" rel="stylesheet">
</head>
<body>

<nav class="navbar navbar-expand-md navbar-dark bg-dark fixed-top">
    <a class="navbar-brand" href="#">Face Detection</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse"
data-target="#navbarCollapse" aria-controls="navbarCollapse" aria-
expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarCollapse">
        <ul class="navbar-nav mr-auto">
            <li class="nav-item">
                <a class="nav-link" href="#"
onclick="loadPage('main')">Home</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="#"
onclick="loadPage('register')">Register</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="#"
onclick="loadPage('login')">Login</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="#"
onclick="loadPage('test')">Take Test</a>
            </li>

        </ul>
    </div>
</nav>

<main role="main" class="container">
```

```
    <div id="content">
        <div class="text-center">
            <img src="your_image_url.jpg" alt="Face Detection" class="img-
fluid" style="max-width: 100%; height: auto;">
            <h1 class="mt-4">Diploma project of Vladyslav Gurniak: Software
System for Detecting Abuses During Online Testing Using Machine Learning
Tools</h1>
        </div>
    </div>
</main>

<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.11.0/umd/popper.min
.js"></script>
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"><
/script>
<script src="scripts/face-api.min.js"></script>
<script src="scripts/main.js"></script>
</body>
</html>
```

Модуль сторінки реєстрації в системі

```
function loadPage(page) {
  const content = document.getElementById('content');

  switch (page) {
    case 'home':
      content.innerHTML = '<h1>Welcome to the Home Page</h1>';
      break;
    case 'login':
      content.innerHTML = `
        <h1>Login</h1>
        <form id="loginForm">
          <label for="email">Email:</label>
          <input type="email" id="email" name="email"><br>
          <label for="password">Password:</label>
          <input type="password" id="password" name="password"><br>
          <button type="button" onclick="login()">Login</button>
        </form>`;
      break;
    case 'register':
      content.innerHTML = `
        <h1>Register</h1>
        <form id="registerForm">
          <label for="firstName">First Name:</label>
          <input type="text" id="firstName" name="firstName"><br>
          <label for="lastName">Last Name:</label>
          <input type="text" id="lastName" name="lastName"><br>
```

```
          <label for="email">Email:</label>
          <input type="email" id="email" name="email"><br>
          <label for="password">Password:</label>
          <input type="password" id="password" name="password"><br>
          <label for="photo">Photo:</label>
          <input type="file" id="photo" name="photo" accept="image/*"
capture="user"><br>
          <button type="button" onclick="register()">Register</button>
        </form>
        <video id="video" width="720" height="560" autoplay
muted></video>`;
      setupCamera();
      break;
    case 'test':
      content.innerHTML = '<h1>Take Test</h1>';
      // Include test logic here
      break;
  }
}

function setupCamera() {
  const video = document.getElementById('video');

  navigator.mediaDevices.getUserMedia({ video: {} })
    .then(stream => {
      video.srcObject = stream;
    })
    .catch(err => console.error(err));

  video.addEventListener('play', async () => {
    await faceapi.nets.tinyFaceDetector.loadFromUri('/models');
    await faceapi.nets.faceLandmark68Net.loadFromUri('/models');

    const displaySize = { width: video.width, height: video.height };
    faceapi.matchDimensions(video, displaySize);

    setInterval(async () => {
      const detections = await faceapi.detectAllFaces(video, new
faceapi.TinyFaceDetectorOptions())
        .withFaceLandmarks();
      const resizedDetections = faceapi.resizeResults(detections,
displaySize);

      if (resizedDetections.length > 0) {
        const canvas = document.createElement('canvas');
        canvas.width = video.width;
        canvas.height = video.height;
        document.body.append(canvas);

        faceapi.draw.drawDetections(canvas, resizedDetections);
        faceapi.draw.drawFaceLandmarks(canvas, resizedDetections);
```

```
        const landmarks = resizedDetections[0].landmarks;
        const points = landmarks.positions.map(p => ({ x: p.x, y: p.y }));
        savePhotoWithLandmarks(video, points);
      }
    }, 100);
  });
}

function savePhotoWithLandmarks(video, landmarks) {
  const canvas = document.createElement('canvas');
  canvas.width = video.videoWidth;
  canvas.height = video.videoHeight;
  const context = canvas.getContext('2d');
  context.drawImage(video, 0, 0, canvas.width, canvas.height);

  const dataUrl = canvas.toDataURL('image/png');

  // Send dataUrl and landmarks to the server
  fetch('/save_photo', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
    },
    body: JSON.stringify({ photo: dataUrl, landmarks }),
  }).then(response => response.json())
    .then(data => console.log(data))
    .catch(err => console.error('Error:', err));
}
```

Модуль входу в систему

```
const express = require('express');
const bcrypt = require('bcrypt');
const jwt = require('jsonwebtoken');
const { Pool } = require('pg');

const app = express();
const port = 3000;

const pool = new Pool({
    user: 'admin',
    host: 'localhost',
    database: 'gurniak',
    password: '',
    port: 3306,
});

const jwtSecret = 'your_jwt_secret';
```

```
app.use(express.json());

app.post('/login', async (req, res) => {
    const { email, password } = req.body;

    try {
        const userResult = await pool.query('SELECT * FROM Users WHERE
email = $1', [email]);

        if (userResult.rows.length === 0) {
            return res.status(400).json({ error: 'Invalid email or
password' });
        }

        const user = userResult.rows[0];

        const isMatch = await bcrypt.compare(password, user.password_hash);

        if (!isMatch) {
            return res.status(400).json({ error: 'Invalid email or
password' });
        }

        const token = jwt.sign({ userId: user.id, role: user.role_id },
jwtSecret, { expiresIn: '1h' });

        res.json({
            token,
            user: {
                firstName: user.first_name,
                lastName: user.last_name
            }
        });
    } catch (error) {
        res.status(500).json({ error: error.message });
    }
});

app.listen(port, () => {
    console.log(`Server running on port ${port}`);
});
```

Модуль збереження даних

```
const express = require('express');
const bodyParser = require('body-parser');
const fs = require('fs');
const path = require('path');
```

```
const app = express();

app.use(bodyParser.json({ limit: '50mb' }));
app.use(express.static('public'));

app.post('/register', (req, res) => {
    const { firstName, lastName, middleName, birthDate, email, address,
city, country, password, photo, landmarks } = req.body;
    const photoBuffer =
Buffer.from(photo.replace(/^data:image\/\w+;base64,/, ""), 'base64');

    const photoPath = path.join(__dirname, 'uploads', `${Date.now()}.png`);
    fs.writeFileSync(photoPath, photoBuffer);

    // Save the user data and landmarks to the database
    const db = require('./db');
    db.query('INSERT INTO Users (first_name, last_name, middle_name,
birth_date, email, address, city, country, role_id, created_at, photo_path,
descriptor, password_hash, salt) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, NOW(),
?, ?, ?, ?)', [firstName, lastName, middleName, birthDate, email, address,
city, country, 2, photoPath, JSON.stringify(landmarks), 'hashed_password',
'salt'], (err, result) => {
        if (err) {
            console.error(err);
            res.status(500).send('Error saving user data');
        } else {
            res.status(200).send('User registered successfully');
        }
    });
});

app.listen(3000, () => {
    console.log('Server running on port 3000');
});
```

Модуль збереження даних нового користувача

```
const express = require('express');
const bcrypt = require('bcrypt');
const { Pool } = require('pg');
const multer = require('multer');
const path = require('path');

const app = express();
const port = 3000;

const pool = new Pool({
    host: 'localhost',
    user: 'admin',
```

```
    password: '',
    database: 'gurniak'
});

app.use(express.json());

const storage = multer.diskStorage({
    destination: (req, file, cb) => {
        cb(null, 'uploads/');
    },
    filename: (req, file, cb) => {
        cb(null, Date.now() + path.extname(file.originalname));
    },
});

const upload = multer({ storage });

app.post('/register', upload.single('photo'), async (req, res) => {
    const { first_name, last_name, middle_name, birth_date, email, address,
city, country, role_id, password, landmarks } = req.body;
    const photo_path = req.file.path;

    try {
        const salt = await bcrypt.genSalt(10);
        const password_hash = await bcrypt.hash(password, salt);

        const result = await pool.query(
            `INSERT INTO Users (first_name, last_name, middle_name,
birth_date, email, address, city, country, role_id, created_at, photo_path,
descriptor, password_hash, salt)
            VALUES ($1, $2, $3, $4, $5, $6, $7, $8, $9, NOW(), $10, $11,
$12, $13) RETURNING id`,
            [first_name, last_name, middle_name, birth_date, email,
address, city, country, role_id, photo_path, landmarks, password_hash,
salt]
        );

        res.status(201).json({ user_id: result.rows[0].id });
    } catch (error) {
        res.status(500).json({ error: error.message });
    }
});

app.listen(port, () => {
    console.log(`Server running on port ${port}`);
});
```

Модуль спостереження під час тестування

```
const video = document.getElementById('video');
```

```
Promise.all([
  faceapi.nets.tinyFaceDetector.loadFromUri('/models'),
  faceapi.nets.faceLandmark68Net.loadFromUri('/models'),
  faceapi.nets.faceRecognitionNet.loadFromUri('/models'),
  faceapi.nets.faceExpressionNet.loadFromUri('/models')
]).then(startVideo).catch(err => {
  console.error("Model loading failed:", err);
});

let registeredFaceDescriptor = null;
let previousLandmarks = null;
let minMouthDistance = Infinity;

function startVideo() {
  navigator.mediaDevices.getUserMedia({ video: {} })
    .then(stream => {
      video.srcObject = stream;
      video.play();
    })
    .catch(err => {
      console.error("Error accessing the webcam: ", err);
      alert("Error accessing the webcam: " + err.message);
    });
}

video.addEventListener('loadeddata', () => {
  const canvas = faceapi.createCanvasFromMedia(video);
  document.body.append(canvas);
  const displaySize = { width: video.width, height: video.height };
  faceapi.matchDimensions(canvas, displaySize);

  setInterval(async () => {
    const detections = await faceapi.detectAllFaces(video, new
faceapi.TinyFaceDetectorOptions()).withFaceLandmarks().withFaceDescriptors(
);
    const resizedDetections = faceapi.resizeResults(detections,
displaySize);
    const context = canvas.getContext('2d');
    context.clearRect(0, 0, canvas.width, canvas.height);

    faceapi.draw.drawDetections(canvas, resizedDetections);
   faceapi.draw.drawFaceLandmarks(canvas, resizedDetections);

    if (resizedDetections.length > 0) {
      if (!registeredFaceDescriptor) {
        registeredFaceDescriptor = resizedDetections[0].descriptor;
        console.log('Registered face descriptor:',
registeredFaceDescriptor);
      }
```

```
      const distances = resizedDetections.map(d =>
faceapi.euclideanDistance(d.descriptor, registeredFaceDescriptor));
      const mainFaceIndex = distances.indexOf(Math.min(...distances));
      const mainFace = resizedDetections[mainFaceIndex];
      const landmarks = mainFace.landmarks;
      const leftEye = landmarks.getLeftEye();
      const rightEye = landmarks.getRightEye();
      const mouth = landmarks.getMouth();

      if (previousLandmarks) {
        const leftPupilMovement =
detectPupilMovement(previousLandmarks.leftEye, leftEye);
        const rightPupilMovement =
detectPupilMovement(previousLandmarks.rightEye, rightEye);
        const mouthMovement = detectMouthMovement(mouth);

        if (leftPupilMovement || rightPupilMovement) {
          drawRectangleAroundEyes(context, leftEye, rightEye);
          displayMessage(context, 'Eye Movement', displaySize.width / 2,
50);
        }

        if (mouthMovement) {
          drawRectangleAroundMouth(context, mouth);
          displayMessage(context, 'Lips Movement', displaySize.width / 2,
100);
        }
      }

      previousLandmarks = {
        leftEye,
        rightEye,
        mouth
      };

      for (let i = 0; i < resizedDetections.length; i++) {
        if (i !== mainFaceIndex) {
          const additionalFace = resizedDetections[i].detection.box;
          drawRectangleAroundFace(context, additionalFace);
          displayMessage(context, 'Additional person', additionalFace.x +
                additionalFace.width / 2, additionalFace.y - 10);
        }
      }
    }
  }, 100);
});

function detectPupilMovement(previousEye, currentEye) {
  const previousCenter = getPupilCenter(previousEye);
  const currentCenter = getPupilCenter(currentEye);
  const dx = Math.abs(previousCenter.x - currentCenter.x);
```

```
  const movementThreshold = 3; // Threshold for significant movement


  return dx > movementThreshold;
}

function getPupilCenter(eyeLandmarks) {
  return {
    x: (eyeLandmarks[0].x + eyeLandmarks[3].x) / 2,
    y: (eyeLandmarks[1].y + eyeLandmarks[5].y) / 2
  };
}

function detectMouthMovement(mouth) {
  const upperLip = mouth[13]; // Середня точка верхньої губи
  const lowerLip = mouth[14]; // Середня точка нижньої губи
  const currentDistance = Math.abs(upperLip.y - lowerLip.y);

  if (currentDistance < minMouthDistance) {
    minMouthDistance = currentDistance;
  }

  const movementThreshold = 7 * minMouthDistance;
  // 50% збільшення від мінімальної відстані

  return currentDistance > movementThreshold;
}

function drawRectangleAroundEyes(context, leftEye, rightEye) {
  const allEyePoints = leftEye.concat(rightEye);
  const eyeBox = getBoundingBox(allEyePoints);

  const x = eyeBox.x;
  const y = eyeBox.y - eyeBox.height / 2; // Центрування прямокутника
вертикально
  const width = eyeBox.width;
  const height = eyeBox.height * 2; // Подвоєння висоти очей

  context.strokeStyle = 'red';
  context.lineWidth = 2;
  context.strokeRect(x, y, width, height);
}

function drawRectangleAroundMouth(context, mouth) {
  const mouthBox = getBoundingBox(mouth);

  const x = mouthBox.x;
  const y = mouthBox.y - mouthBox.height / 2; // Центрування прямокутника
вертикально
  const width = mouthBox.width;
  const height = mouthBox.height * 2; // Подвоєння висоти рота
```

```
  context.strokeStyle = 'red';
  context.lineWidth = 2;
  context.strokeRect(x, y, width, height);
}

function drawRectangleAroundFace(context, box) {
  context.strokeStyle = 'red';
  context.lineWidth = 2;
  context.strokeRect(box.x, box.y, box.width, box.height);
}

function getBoundingBox(points) {
  const x = Math.min(...points.map(point => point.x));
  const y = Math.min(...points.map(point => point.y));
  const width = Math.max(...points.map(point => point.x)) - x;
  const height = Math.max(...points.map(point => point.y)) - y;
  return { x, y, width, height };
}

function displayMessage(context, message, x, y) {
  context.fillStyle = 'red';
  context.font = '20px Arial';
  context.textAlign = 'center';
  context.fillText(message, x, y);
}
```

ДОДАТОК Б

Копія публікації

ЦФЕНД ЦЕНТР ФІНАНСОВО-ЕКОНОМІЧНИХ НАУКОВИХ ДОСЛІДЖЕНЬ

МІЖНАРОДНА НАУКОВО-ПРАКТИЧНА КОНФЕРЕНЦІЯ
INTERNATIONAL SCIENTIFIC-PRACTICAL CONFERENCE

ПЕРСПЕКТИВНІ НАПРЯМКИ РОЗВИТКУ ЕКОНОМІКИ, ОБЛІКУ, УПРАВЛІННЯ ТА ПРАВА: ТЕОРІЯ І ПРАКТИКА

PERSPECTIVE DIRECTIONS FOR THE DEVELOPMENT OF ECONOMICS, ACCOUNTING, MANAGEMENT AND LAW: THEORY AND PRACTICE

Збірник тез доповідей
Book of abstracts

18 травня 2024 р.
May 18, 2024

м. Ізмаїл, Україна
Izmail, Ukraine

УДК 004.8

Гурняк В. І.
студент IV курсу
Західноукраїнський національний університет
Гриньків А. М.
аспірант
Західноукраїнський національний університет

ПРОГРАМНА СИСТЕМА ВИЯВЛЕННЯ ЗЛОВЖИВАНЬ ПІД ЧАС ОНЛАЙН-ТЕСТУВАНЬ ЗАСОБАМИ МАШИННОГО НАВЧАННЯ

Сучасні технології значно сприяли розвитку дистанційної освіти, яка стала важливою складовою освітнього процесу під час пандемії COVID-19 та військових дій в Україні, також завдяки своїй зручності, доступності та можливості навчатися у найкращих спеціалістів світу. Проте такий підхід до навчання підвищує ризики академічної недоброчесності, оскільки студенти мають більше можливостей для зловживань, використовуючи конспекти, книжки, онлайн-ресурси та навіть інших сторонніх осіб для складання тестів чи іспитів [1]. Тому необхідність розробки програмної системи виявлення зловживань під час онлайн-тестувань є актуальною задачею.

Для вирішення проблеми зловживань під час онлайн-тестувань пропонується використовувати методи обробки зображень, відео та звуку з відеокамери, мікрофону та копії екрану, що об'єднані в єдину програмну систему моніторингу процесу онлайн-тестувань [2].

Аналіз зображень і відео включатиме відстеження очей і пози голови учасника тестування. Система аналізуватиме рухи голови та позу очей учасника, відстежуючи, наскільки він спостерігає за екраном. Якщо учасник відвертає погляд від екрану, система надсилатиме попередження. Для реалізації даного функціоналу будуть використані бібліотеки OpenCV, детектор облич dlib [3].

Якщо на зображенні з'являється декілька осіб, система автоматично реагуватиме, роблячи знімок і відправляючи попередження. Також система виявлятиме мобільні телефони, автоматично реагуючи, роблячи знімок та відправляючи попередження. Якщо учасника не буде в полі зору камери, система реагуватиме, роблячи знімок та засікаючи час його відсутності.

Виявлення руху губ і контроль на основі звуку дозволять ідентифікувати шахрайство, коли учасник тестування відкриває рот, щоб задати запитання або отримати допомогу. У такій ситуації система починатиме запис, ідентифікуючи зловживання.

Отже, запропонована програмна система виявлення зловживань під час онлайн-тестувань засобами машинного навчання дасть змогу покращити процес оцінки знань, забезпечуючи справедливість і чесність оцінювання студентів.

Список літератури

1. Distance Learning Advantages and Disadvantages: Teaching Experience Analysis at the University with the Basis on Different Informational-Communicative Technologies. / Lobanova, Y.I., Silhavy, R. (eds) // Artificial Intelligence in Intelligent Systems., CSOC 2021. Lecture Notes in Networks and Systems, vol 229. Springer, Cham. https://doi.org/10.1007/978-3-030-77445-5_46

2. Aiman A Turani; Jawad H Alkhateeb; AbdulRahman A. Alsewari.: "Students Online Exam Proctoring: A Case Study Using 360 Degree Security Cameras", 2020 Emerging Technology in Computing, Communication and Electronics (ETCCE), 2020.

3. Hadian S. G. Asep,Yoanes Bandung.:"A Design of Continuous User Verification for Online Exam Proctoring on MLearning",In.: 2019 International Conference on Electrical Engineering and Informatics (ICEEI), 9-10 July 2019.

УДК 004.67

Завойовська О. М.
студентка 4 курсу факультету комп'ютерних
інформаційних технологій
Західноукраїнського національного університету

ПРОГРАМНИЙ МОДУЛЬ ДЛЯ ВИЯВЛЕННЯ ОБРАЗЛИВИХ СЛІВ ТА НЕНАВИСТІ В ТЕКСТОВИХ ПОВІДОМЛЕННЯХ

У сучасному інформаційному середовищі, що характеризується стрімким зростанням обсягів текстової інформації, питання виявлення та запобігання образливих слів та проявів ненависті в текстових повідомленнях стає все більш актуальним. Соціальні мережі, форуми, коментарі на сайтах новин, електронна пошта та інші платформи для обміну повідомленнями часто стають середовищем для поширення ненависницьких висловлювань, які можуть призвести до серйозних соціальних наслідків. Розробка програмного модуля для автоматичного виявлення таких проявів є важливим кроком у боротьбі з онлайн-насильством та кібербулінгом. [1]

Проблема мови ненависті та образ у мережі є глобальною і торкається всіх аспектів суспільного життя. Соціальні платформи, форуми та чати часто стають майданчиками для розповсюдження ненависті, що може призвести до дискримінації, цькування та інших негативних явищ. Тому розробка ефективних інструментів для виявлення та блокування таких повідомлень є критично важливою. [2]

Метою даної роботи є розробка та впровадження програмного модуля, який буде здатний ефективно ідентифікувати образливі слова та висловлювання ненависті у текстових повідомленнях. Зокрема, дослідження зосереджується на застосуванні сучасних методів обробки природної мови (NLP) та машинного навчання для створення точного і надійного інструменту, який зможе автоматично аналізувати великі обсяги текстових даних.