Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему *Метод спостереження за очима (eye-tracking) для вебсистеми тестування знань*

Галузь знань *12 – Інформаційні технології*
Шифр і назва галузі знань

Спеціальність *122 – Комп'ютерні науки*
Шифр і назва спеціальності

Освітня програма *Комп'ютерні науки*
Назва освітньої програми

Виконала: *студент групи КН-20-2* (Група виконавця) _____ (Підпис) *Богдан РОМАНОВ* (Ім'я, ПРІЗВИЩЕ)

Керівник: *зав.каф. КН,д.т.н.,проф.* (Науковий ступінь, посада) _____ (Підпис) *Олександр БАРМАК* (Ім'я, ПРІЗВИЩЕ)

Нормоконтроль: *к.т.н., доц. каф. КН* (Науковий ступінь, посада) _____ (Підпис) *Руслан БАГРІЙ* (Ім'я, ПРІЗВИЩЕ)

До захисту допускаю:
зав. кафедри КН, д.т.н., професор _____ (Підпис) *Олександр БАРМАК* (Ім'я, ПРІЗВИЩЕ)

12 06 2024 р.

Хмельницький 2024

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
Факультет *інформаційних технологій*
Кафедра *комп'ютерних наук*
Освітній ступінь *бакалавр*
Галузь знань *12 – Інформаційні технології*
Спеціальність *122 – Комп'ютерні науки*

ЗАТВЕРДЖУЮ
Завідувач кафедри *комп'ютерних наук*
_____
(підпис)
д.т.н., професор Олександр БАРМАК
«*16*» *02* *2024* року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

1. Тема кваліфікаційної роботи бакалавра: *«Метод спостереження за очима (eye-tracking) для вебсистеми тестування знань»*

2. Завдання видано студенту *Богдану РОМАНОВУ*
(Ім'я, прізвище)

3. Керівник роботи *зав. кафедри КН, д.т.н., професор Олександр БАРМАК*
(посада, ім'я, прізвище)

4. *Затверджено наказом університету від «15» 02 2024 р. №8*

5. *Дата видачі завдання студенту: «16» 02 2024 р.*

6. *Зміст пояснювальної записки (перелік задач) та вихідні дані:*

Мета роботи – підвищення академічної доброчесності під час проходження тестувань за аналізом погляду шляхом підрахунку кількості часу, коли погляд знаходиться поза екраном. Для реалізації поставленої мети виконанні та описані завдання: аналіз існуючих підходів які реалізують метод спостереження за очима під час тестування; запропоновано метод спостереження за очима; реалізовано аналіз відео потоку на стороні сервера під час проходження тестувань; реалізована вебсистема тестування знань з вбудованим методом спостереження за очима.

Анотація

Тема кваліфікаційної роботи бакалавра: *«Метод спостереження за очима (eye-tracking) для вебсистеми тестування знань»*

Виконавець кваліфікаційної роботи бакалавра: *студент групи КН-20-2 Богдан РОМАНОВ*

Керівник кваліфікаційної роботи бакалавра: *зав. кафедри КН, д.т.н., професор Олександр БАРМАК*

Кваліфікаційна робота бакалавра містить:

Пояснювальна записка				Кількість додатків
Сторінок	Рисунків	Таблиць	Джерел інформації	
72	54	0	27	2

Метою кваліфікаційної роботи бакалавра є підвищення академічної доброчесності під час проходження тестувань за аналізом погляду шляхом підрахунку кількості часу, коли погляд знаходиться поза екраном. Для розробки зазначеного методу та інформаційної системи було використано методи обробки зображень у відеопотоці, методи виявлення елементів обличчя на зображенні, методи проходження тестувань знань, технології створення вебзастосувань.

Розроблена система призначена для вчителів та учнів, що мають потребу у ефективному тестуванні знань. Реалізована система спостереження за очима (eye-tracking) для тестування знань дозволяє підвищити академічну доброчесність під час проходження тестувань.

Напрямками практичного використання розробленого методу є використання його у інформаційних системах тестування знань.

Ключові слова: академічна доброчесність, eye-tracking, інформаційна система, освіта, тестування знань, вебсистема.

Виконавець: *студент групи КН-20-2* (Група виконавця) _____ (Підпис) *Богдан РОМАНОВ* (Ім'я, ПРІЗВИЩЕ)

7. Календарний план виконання кваліфікаційної роботи бакалавра:

№	Назва етапів (розділів) кваліфікаційної роботи бакалавра	Термін виконання	Приміт
1	*Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи бакалавра з керівником, складання календарного графіка виконання роботи*	*січень 2024*	*виконан*
2	*Ознайомлення з предметною областю, формулювання мети та задач дослідження, визначення об'єкта та предмета дослідження*	*січень 2024*	*виконан*
3	*Робота над розділом 1 – Аналіз систем та методів спостереження за очима для задачі тестування знань у онлайн режимі*	*лютий 2024*	*виконан*
4	*Робота над розділом 2 – Метод спостереження за очима для систем тестування знань*	*березень 2024*	*виконан*
5	*Робота над розділом 3 – Програмна реалізація вебсистеми тестування знань з вбудованим методом спостереження за очима*	*квітень 2024*	*виконан*
6	*Оформлення пояснювальної записки згідно вимог*	*травень 2024*	*виконано*
7	*Попередній захист кваліфікаційної роботи бакалавра*	*травень 2024*	*виконано*
8	*Захист кваліфікаційної роботи бакалавра*	*червень 2024*	*виконано*

Виконавець: *студент групи КН-20-2* (Група виконавця) _____ (Підпис) *Богдан РОМАНОВ* (Ім'я, ПРІЗВИЩЕ)

Керівник: *зав. каф. КН д.т.н.,проф.* (Науковий ступінь, посада) _____ (Підпис) *Олександр БАРМАК* (Ім'я, ПРІЗВИЩЕ)

Зміст

Перелік скорочень

Скорочення, термін, позначення	Пояснення
СКБД	Система керування базами даних
ЕОГ	Електроокулографія
UI	User Interface (Інтерфейс користувача)
UX	User Experience (Досвід користувача)
CTR	Click Through Rate (Рейтинг кліків)
VR	Virtual Reality (Віртуальна реальність)
БД	База даних
КРБ	Кваліфікаційна робота бакалавра
URL	Uniform Resource Locator (Адреса ресурсу)
MVC	Model View Controller (Модель, Вид, Контролер)
FPS	Frames per second (Кадри за секунду)
SQL	Structured query language (мова структурованих запитів)
HTML	*HyperText Markup Language* (мова розмітки гіпертексту)
MS	Microsoft
VS Code	Visual studio code

Вступ

Одним з актуальних, на сьогоднішній день, напрямків комп’ютерних технологій є комп’ютерний зір: розпізнавання об'єктів, облич, спостереження за очима, на зображеннях та відео. Такі технології можуть бути потужним інструментом у сферах, де є необхідність аналізувати зображення та відео потоки, зокрема системи безпеки, автоматизовані системи навігації тощо. Ці технології також можуть бути застосовані для аналізу поведінки людини, її емоцій, напрямку погляду тощо.

Система спостереження за очима (eye-tracking) може стати одним з потенційних рішень для вирішення проблем академічної доброчесності під час проходження тестування знань.

Тестування та перевірка засвоєних знань завжди була та залишається одною з основних частин освітнього процесу. Тестування дозволяє перевірити рівень розуміння та засвоєння навчального матеріалу і допомагає об’єктивно оцінити знання.

Отже, одним з важливих сучасних інструментів освітнього процесу може стати вебсистема для тестування знань. Така система дозволить автоматизувати певні процеси навчання, мінімізувати людську помилку при оцінюванні рівня знань та забезпечити доступність навчання для широких груп населення.

Але, варто зазначити, що перехід до використання сучасних технологій не є завжди ідеальним вибором та може мати як позитивні, так і негативні наслідки. Зокрема, полегшення доступу до будь якої інформації за допомогою мережі інтернет є потужним інструментом для процесів освіти, але при цьому легкий доступ до інформації може призвести до зменшення загальної якості освіти. Наявність всієї інформації завжди доступною через один пошуковий запит може створити ситуацію, коли вивчення освітньої програми стає менш активним, замість вивчення та розуміння навчального матеріалу учні надаватимуть перевагу простому пошуку відповідей на задане питання у мережі, при цьому не

намагаючись зрозуміти навчальний матеріал, що є проблемою якості освіти та академічної доброчесності.

Системи спостереження за очима та розпізнавання облич можуть бути ефективно застосовані в цілях безпеки та забезпечення якості освіти. Їх застосування дозволить автоматизувати процеси забезпечення академічної доброчесності під час навчання, що також значно зменшує кількість роботи для вчителів.

Об’єкт дослідження – процес проходження тестування знань за аналізом погляду у відеопотоці.

Предмет дослідження – методи обробки зображень у відеопотоці, методи виявлення елементів обличчя на зображенні, методи проходження тестувань знань.

Мета кваліфікаційної роботи бакалавра – підвищення академічної доброчесності під час проходження тестувань за аналізом погляду шляхом підрахунку кількості часу, коли погляд знаходиться поза екраном.

Завдання кваліфікаційної роботи бакалавра – провести аналіз існуючих підходів які реалізують метод спостереження за очима під час тестування; розробити метод спостереження за очима; реалізувати аналіз відео потоку на стороні сервера під час проходження тестувань; розробити веб систему тестування знань з вбудованим методом спостереження за очима з наступною функціональністю: реєстрація та авторизація користувачів: взаємодія з сутностями БД для авторизованих користувачів; відображення списків предметів та тестувань, можливість проходження тестувань для авторизованих користувачів з аналізом за методом спостереження за очима; збереження та відображення даних про результати проходження тестування.

Розділ 1. Аналіз систем та методів спостереження за очима для задачі тестування знань у онлайн режимі

1.1 Огляд систем спостереження за очима (eye-tracking)

Спостереження за очима (eye-tracking) полягає у визначенні точки, в бік якої направлений погляд, або відносного напрямку погляду [1]. Ця технологія може мати велику кількість практичних застосувань, зокрема: спостереження за поведінкою людини, маркетинг, системи що керуються напрямком погляду [2], симуляції реалістичних подій, комп'ютерні ігри [3], системи безпеки тощо.

1.1.1 Огляд теоретичних методів спостереження за очима

Методи спостереження за очима можна класифікувати за декількома властивостями: тип пристроїв для спостереження за очима, типи випромінювання що вони спостерігають та способом отримання кінцевих даних.

За типами пристроїв для спостереження за очима можна виділити [4]:

1. Пристрої, що розміщені у статичній позиції [5] (Рис. 1.1), наприклад закріплені на робочому столі, або спеціально призначеній для цього поверхні [2]. Такими пристроями зазвичай є камери, розміщені таким чином, щоб мати прямий вид на обличчя людини, що спостерігається. За умови мінімальних рухів голови людини, що спостерігається, такі пристрої матимуть найбільшу точність у порівнянні з іншими.

Рисунок 1.1 – Приклад використання статичної камери [5]

2. Пристрої, що розміщуються на голові людини [5], очі якої спостерігаються. Яскравим прикладом такого пристрою є спеціальні окуляри (Рис. 1.2), що можуть візуально зчитувати позицію очей. Такі пристрої зазвичай є простішими у користуванні ніж статичні, оскільки не є надто залежними від положення та рухів голови, але, зазвичай мають меншу точність визначення напрямку погляду через надмірне наближення до очей.

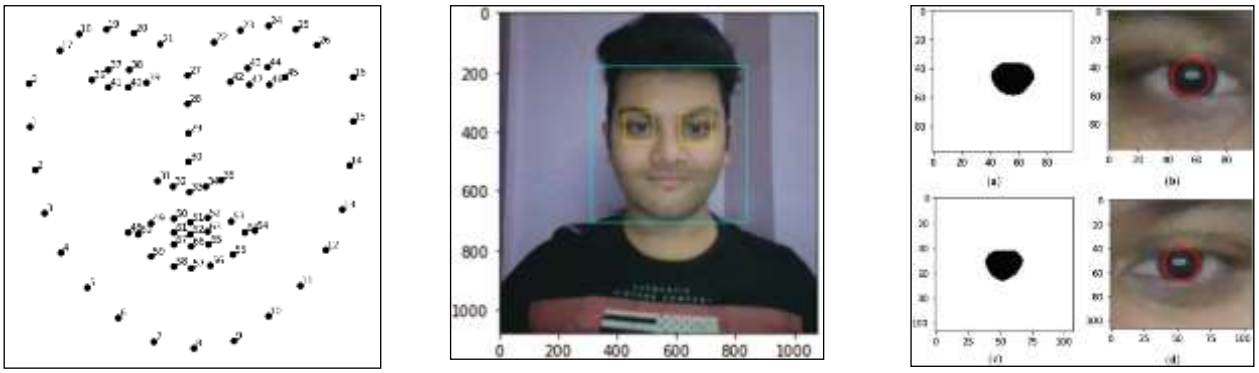
Рисунок 1.2 – Приклад використання окулярів для відстеження положення очей [5]

3. Електроокулографія (ЕОГ) використовує електроди для вимірювання електричних зарядів, що створюються м’язами очей. Цей спосіб не надає будь-яких точних даних про положення очей, але є одним з інструментів у медичній сфері, що може бути використаний під час обстеження пацієнта.

Існує декілька способів автоматизованого спостереження за очима [6]:

1. Розпізнавання зображень [7]. Для цього використовується камера, або декілька камер, дані з яких аналізуються спеціалізованою програмою, що розпізнає положення обличчя та очей на зображенні. В залежності від використаної моделі розпізнавання облич алгоритм розпізнавання напрямку погляду може різнитись. Існують моделі машинного зору, які одразу виділяють позицію зіниці ока на зображенні людського обличчя, але у випадку коли модель лише здатна розпізнавати загальні риси обличчя (Рис. 1.3, а) б)) [8], може знадобитися окремий аналіз розпізнаних елементів [9] (Рис. 1.3 в)). Так на рис. 1.3 б) можна побачити розпізнані позиції та приблизні розміри обличчя та очей на зображенні. Використовуючи ці дані можна виділити частини зображення з

очима та аналізувати їх окремим алгоритмом. На рис 1.3 в) зображено виділення позиції зіниці ока за допомогою окремого алгоритму, цей метод застосовує виділення найтемнішої частини зображення, де найтемнішим елементом залишається зіниця ока, після чого оцінюється її центр та розмір. Маючи позицію зіниць на окремих зображеннях очей, можна порахувати відношення позиції зіниць до позиції центру очей, що дає можливість оцінити напрямок погляду на зображенні з відносною точністю.



а) б) в)

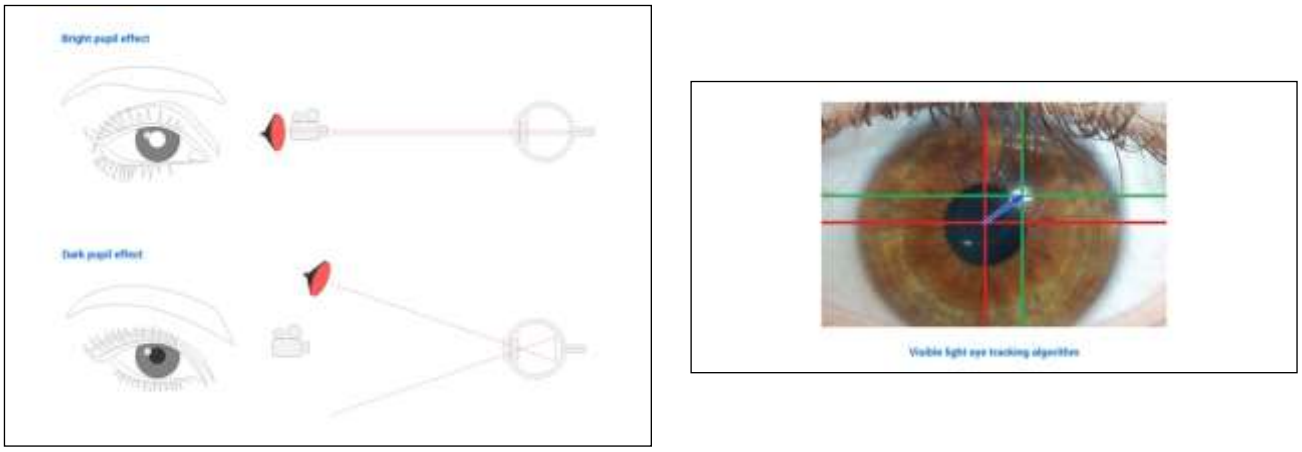
Рисунок 1.3 – а) проста модель для розпізнавання обличчя [10], б) результат розпізнавання обличчя, в) положення очей та положення зіниць у оці на зображенні [7]

2. Інфрачервоне зображення [11]. При застосуванні цього методу використовується випромінювач інфрачервоного світла та камери для спостереження за інфрачервоним випромінюванням, що відбивається від поверхні ока. При застосуванні цього способу можуть бути використані різні конфігурації положення камери та випромінювача (Рис. 1.4 а)). При використанні цього методу необхідно виділити окремо зображення кожного ока та позиції зіниць на ньому, що може бути забезпечено методами розпізнавання зображень за допомогою машинного зору (Рис 1.3 а),б)), або використання камер, розміщених близько до кожного спостережуваного ока. Вважаючи позиції камери та випромінювача статичними, спостережуване відбите інфрачервоне світло слугує орієнтиром для обрахування напрямку погляду [12]. У випадку,

коли випромінювач та камера знаходяться поряд або в межах одного пристрою, випромінення здійснюється по тій же самій осі, що і запис зображення, таким чином коли погляд спрямований в камеру, то слід відбитого випромінювання знаходитиметься в центрі зіниці ока (Рис 1.4 а)), а при зміщенні погляду очей в сторону інфрачервоний слід випромінювача залишиться у відносно статичній позиції на зображенні [12]. Таким чином різниця позиції зіниці ока та позиції відбитого інфрачервоного світла представляє собою вектор напрямку погляду у даний момент часу. При конфігурації випромінювача та камери, де камера спрямована на очі людини а випромінювач знаходиться під кутом точка відбиття інфрачервоного світла також вважається статичною аналогічно до першої конфігурації, але у цьому випадку при погляді в центр камери інфрачервоний слід знаходиться не в центрі зіниці а є дещо зміщеним в залежності від кута між камерою та випромінювачем (Рис 1.4 а)). Через цю особливість зміщеного інфрачервоного світла є необхідність проведення калібрації, тобто запису відносної позиції ока до відбитого світла при погляді в камеру, після чого алгоритм зможе порівнювати позицію зіниці ока з відносною «нульовою» позицією, при чому відкалібрована «нульова» позиція повинна обраховуватись під час кожної ітерації процесу відстеження напрямку погляду очей [12]. Однією з особливостей використання випромінювача під кутом до ока є можливість точного вимірювання кута між випромінювачем та камерою, що в комбінації з позицією відбитого світла на поверхні ока дає можливість дати відносно точну оцінку напрямку погляду у тривимірному просторі.

3. спостереження за відбитим світлом. Використання відбитого світла видимого спектру є аналогічним до інфрачервоного випромінювання [6], але немає необхідності у використанні спеціалізованих пристроїв для випромінювання та спостереження за інфрачервоним світлом, замість чого достатньо використати камеру загального призначення та будь-яке джерело видимого світла направленого в око. На відміну від використання інфрачервоного світла варто зазначити, що направлене видиме світло в центр зіниці може призвести до подразнення очей людини, що спостерігається, тому

найоптимальнішою конфігурацією позицій пристроїв є встановлення джерела світла під кутом до камери та об’єкту, що спостерігається очима. Обрахунок напрямку погляду відбувається аналогічним чином до використання інфрачервоного світла (Рис. 1.4 б)).



а) б)

Рисунок 1.4 – Різні види конфігурацій положень камери та очей та спостереження за відбитим променем видимого світла (справа) [6]

Отже, існує велика кількість теоретичних способів спостереження за очима, що відрізняються використовуваними пристроями, способами їх використання та точністю відстежування напрямку погляду. Кожен з цих способів має свої переваги та недоліки для використання в конкретних сферах людської діяльності. З огляду на особливості існуючих рішень при розробці методу спостереження за очима (eye-tracking) для вебсистеми тестування знань найбільш доцільним є використання алгоритмів розпізнавання зображень, оскільки такий підхід не вимагає значних витрат на встановлення та налаштування спеціальних пристроїв для клієнта та є сумісним з більшою частиною мобільних пристроїв або комп’ютерів клієнта за умови наявності камери.

1.1.2 Огляд існуючих рішень з функціями спостереження за очима

На сьогоднішній день існує багато готових рішень з функціями спостереження за очима [13], але більшість з них мають на меті комерційний аналіз поведінки потенційних клієнтів.

Lumen використовують спостереження за очима та інші види спостереження за поведінкою людини з ціллю аналізу того, що саме привертає увагу до реклами, або здійснення покупки тощо [14]. Lumen надає інструменти для тестування та оптимізації реклами, UX та UI дизайну для цілей збільшення продажів, або CTR (Рис. 1.5).

Рисунок 1.5 – Вигляд інтерфейсу головної сторінки аналізу інтернет реклами Lumen [13]

Tobii є одним з постачальників рішень для спостереження за очима. Ця компанія надає як апаратну так і програмну частину для спостереження за очима, включаючи камери та окуляри для відстеження очей (Рис. 1.6 а)) [15]. Здебільшого ці технології фокусуються на дослідженнях у сфері маркетингу та поведінки людей.

Eyezag надає рішення для спостереження за очима з використанням звичайних камер як на ПК так і мобільних пристроях [16]. ЇЇ послуги включають створення опитувань, інструменти для аналітики, проведення тестувань для створюваного контенту тощо (Рис. 1.6 б)).

а) б)

Рисунок 1.6 – а) VR окуляри з функцією спостереження за очима Tobii, б) вигляд інтерфейсу інструментів аналітики eyezag [13]

Element Human є простою у вивченні та використанні платформою для аналізу користувачів, для чого використовуються різні методики, одною з яких є спостереження за очима [17]. Для визначення того, як люди реагують на різні середовища Element Human застосовує такі методики як опитування, спостереження за очима, розпізнавання емоцій, що надає можливість швидко збирати дані про те, як користувачі себе поводять при використанні деякого вебсервісу (Рис. 1.7 а)).

RealEye надає функціонал спостереження за очима з використанням камери та відстежування положення миші тощо [18]. Ця платформа підтримує 10 мов та може бути інтегрована з вебсервісами (Рис. 1.7 б)) для тестування поведінки користувачів. Її функціонал надає дані аналітики при кожному тестуванні, при цьому одною з її особливих функцій є можливість передати результати за допомогою посилання або завантаження файлів.

а) б)

Рисунок 1.7 – а) вигляд інтерфейсу Element Human при аналізі тестового відео, б) вигляд інтерфейсу RealEye при тестуванні вебсторінки [13]

Окрім здебільшого комерційного використання технології спостереження за очима також починають активно використовуватись і в галузі освіти, наприклад EyePass.

Під час онлайн-презентації на платформі Українського кризового медіа-центру компанія Beehiveor презентувала технологію EyePass, надає можливість забезпечувати якість дистанційного навчання. Технологія EyePass, яка працює з використанням спостереження за очима використовує web-камеру для відстежування відстежування погляду та фокусування уваги [20]. Система EyePass створена з можливістю інтеграції з системами освіти та тестування знань, наприклад Moodle [21].

Отже, існує досить велика кількість рішень з функціями спостереження за очима, що можуть бути використані у різноманітних галузях людською діяльності. Варто зазначити, що найбільш успішними та популярними є системи для аналізу поведінки потенційних клієнтів в цілях оптимізації маркетингу, UI та UX дизайну вебресурсів, а існуючі системи призначені для використання в галузі освіти не є надто розповсюдженими та досконалими.

1.2 Огляд існуючих систем тестування знань

Одним з прикладів вебсайту для тестування знань є «На Урок» [22]. Даний сайт надає навчальні онлайн курси з шкільної програми, функціонал

тестування знань з різних предметів, інформацію про вебінари, олімпіади, конкурси тощо. (Рис. 1.8)

Рисунок 1.8 – Вигляд навігаційної панелі web-сайту «На Урок» [22]

При переході по навігаційній панелі до тестів (Рис 1.8) вебсайт пропонує нам список тестів, які можна пройти та можливість приєднатись до конкретного тесту використовуючи форму «Приєднатись до тесту», що дозволяє учневі знайти та пройти необхідний тест, за умови, що вчитель надав код тестування, з яким асоційоване потрібне тестування (Рис. 1.9).

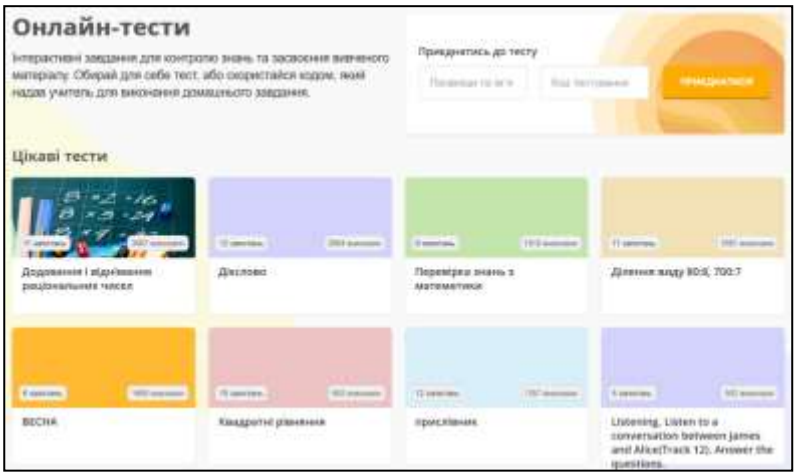


Рисунок 1.9 – Вигляд сторінки тестів «На Урок» [22]

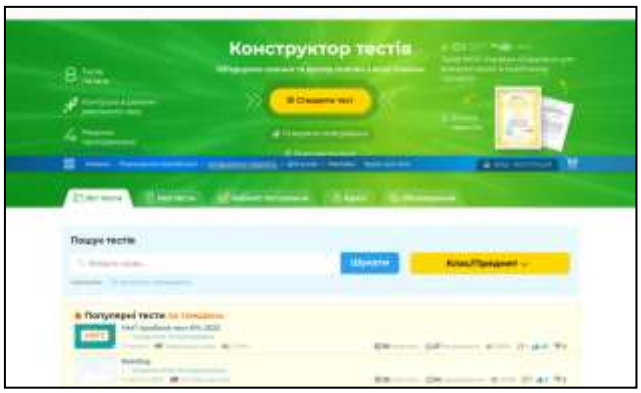
Даний web-сайт може бути практично застосований для організації тестування у шкільних закладах, завдяки можливості створення індивідуалізованих тестів вчителями для кожної окремої групи учнів.

Ще одним прикладом web-сайту з функціоналом тестування є «Освіта.ua» [23]. Даний ресурс надає можливість пройти тестування для підготовки, та оцінювання знань, необхідних для успішного складання ЗНО з різних предметів (Рис. 1.10).

Рисунок 1.10 – Вигляд сторінки тестів «Освіта.ua» [23]

Завдяки цьому ресурсу учень може переконатися у своїй підготовці до ЗНО, або розглянути надані завдання для подальшої підготовки.

Web портал «vseosvita.ua» [24] також надає можливість створення та проходження тестів безпосередньо користувачами (Рис. 1.11 а)), але при цьому для доступу до тестів необхідно зареєструватись. Однією з особливих функцій цього ресурсу є можливість створення нових тестів на основі вже існуючих (Рис 1.11 б)).



а) б)

Рисунок 1.11 – а) вигляд списку тестів «vseosvita.ua» [24], б)вигляд сторінки окремо взятого тесту на «vseosvita.ua» [24]

Іншим прикладом ресурсу, що надає функціонал тестування знань є «Moodle» [21]. На відміну від попередніх прикладів, «Moodle» представляє собою повноцінну платформу управління навчанням з відкритим кодом, що може бути локально розгорнута та налаштована під потреби конкретного

навчального закладу чи установи [21] (рис. 1.12). Таким чином, кожен Web-сервіс на основі «Moodle» є незалежним від інших, оскільки серверна частина середовища зберігається на локальному сервері, або в хмарному середовищі, що збільшує її надійність, завдяки загальній децентралізованості серверної частини.

Платформа «Moodle» надає функціонал створення курсів, розміщення навчального матеріалу, тестування, комунікації тощо [21]. Одною з основних особливостей «Moodle» є сумісність з технологією «EyePass», що надає можливість забезпечення якості освіти за допомогою спостереження за очима.

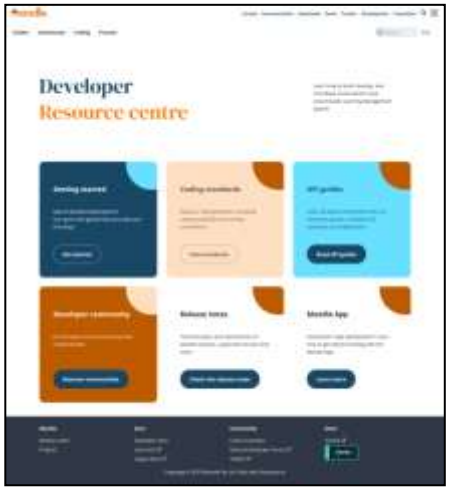


Рисунок 1.12 – Вигляд сторінки документації для розробників «Moodle» [21]

Отже, усі розглянуті сервіси в певній мірі мають реалізований функціонал проведення тестувань, перевірки та оцінювання результату. Також, однією із спільних рис можна назвати найімовірніше використання БД для збереження як самих завдань тестування та їх відповідей, так і результатів та оцінок при їх проходженні учнями. Як виявляється, більшість відомих систем тестування знань не використовують технологію спостереження за очима, їх використання може значно поліпшити освітній процес в більшості навчальних закладів, але використання технології спостереження за очима надає перевагу у забезпеченні якості тестування знань. У порівнянні з сервісами які не використовують спостереження за очима, використання «Moodle» з технологією «EyePass» може

мати значний позитивний вплив на академічну доброчесність учнів під час тестування знань.

1.3 Мета та завдання кваліфікаційної роботи

Отже, метою роботи є підвищення академічної доброчесності під час проходження тестувань за аналізом погляду шляхом обрахунку кількості часу, коли погляд знаходиться поза екраном.

Для реалізації поставленої мети, потрібно виконати наступні завдання:

– провести аналіз існуючих підходів які реалізують метод спостереження за очима під час тестування;

– запропонувати та реалізувати метод спостереження за очима;

– реалізувати аналіз відео потоку на стороні сервера під час проходження тестувань;

– реалізувати вебсистему тестування знань з вбудованим методом спостереження за очима з наступною функціональністю:

- реєстрація та авторизація користувачів:
- взаємодія з сутностями БД для авторизованих користувачів;
- відображення списків предметів та тестувань, можливість проходження тестувань для авторизованих користувачів з аналізом за методом спостереження за очима;
- збереження та відображення даних про результати проходження тестування.

Розділ 2 Метод спостереження за очима для систем тестування знань

2.1 Критерії поведінки при тестуванні за аналізом погляду

Для початку необхідно визначитися з критеріями поведінки, які випливають з аналізу погляду і є показниками для прийняття рішень при тестуванні знань.

При спостереженні за очима можна виділити такі види рухів очей як фіксації та саккади [13], де фіксації передбачають фокусування погляду у певному напрямку, а саккади є рухами погляду між точками фокусування. Використання різних метрик, що стосуються даних типових станів погляду дозволяє проаналізувати поведінку людини.

При аналізі погляду під час тестування можна застосовувати наступні критерії:

– *напрямок\координати фіксацій зору* дозволяють визначити використання сторонніх ресурсів під час проходження тестувань – під час тестування погляд повинен бути спрямований на частину екрану з тестуванням;

– *тривалість фіксацій* у деякому напрямку в сукупності з координатами фіксацій дозволяє з кращою точністю визначити ймовірність академічної недоброчесності – тривалі фіксації поза екраном свідчать про недотримання академічної доброчесності; також тривалі фіксації можуть говорити про підвищену розумову активність;

– *частота фіксацій* може бути використана для аналізу поведінки та академічної доброчесності під час тестування – велике значення частоти фіксацій може свідчити про підвищений рівень стресу, а велике значення частоти фіксацій поза екраном є ознакою академічної не доброчесності;

– *частота кліпання очей* може бути використана для аналізу рівня втоми та розумового навантаження людини – підвищена частота кліпання є ознакою підвищених втоми та розумового навантаження;

– *швидкість саккадів* знижується від втоми та втрати уважності, підвищена швидкість саккадів може вказувати на підвищену складність тестування.

Отже, під час аналізу погляду для аналізу поведінки під час тестування можна використовувати такі критерії, як частота саккадів, фіксацій та частота кліпання очей, а для визначення рівня академічної доброчесності достатньо напрямку\координат фіксацій, їх частоту та довжину.

2.2 Засоби аналізу зображення для спостереження за поглядом

При розробці методу спостереження за очима для вебсистеми тестування знань найкращим способом є розпізнавання зображень з камери, оскільки такий підхід дозволяє використання системи майже будь-яким пристроям, які оснащені камерою та мають доступ у мережу.

Виходячи зі особливостей вебзастосунку, засоби спостереження за поглядом варто розділити на клієнтську та серверну частини.

Для отримання зображення з клієнтської частини, на стороні клієнта повинна бути наявні камера для запису зображення та доступ до мережі інтернет і будь-який браузер для передачі його на серверну частину.

На серверній частині розміщуватиметься інфраструктура вебзастосунку та відкритий інтерфейс взаємодії з клієнтами. Для аналізу зображення для спостереження за поглядом необхідно інтегрувати метод аналізу погляду в існуючу вебсистему.

Для інтеграція методу спостереження за очима у вебсистему, найдоцільнішим на рівні основної бізнес-логіки, за шаблоном MVC є використання для реалізації відповідних контролерів.

Сам метод спостереження за очима найдоцільніше будувати на основі моделі машинного навчання для розпізнавання елементів обличчя. Така модель надасть координати очей та інших елементів, необхідних для обрахунку напрямку погляду та інших його параметрів.

Отже, для аналізу зображення нам необхідно висунути вимоги щодо наявності камери у клієнтської частини та інтеграції методу аналізу зображення на серверній частині. Такий підхід не створить значного навантаження на клієнтську частину та дозволить проводити аналіз зображення з камери клієнта для спостереження за поглядом.

2.3 Метод спостереження за поглядом у відео потоці

У роботі запропоновано метод спостереження за поглядом очей у відео потоці.

Для наглядного розуміння алгоритм було розділено на 4 основних блоки. Загальна схема методу наведена на рисунку 2.1.

Рисунок 2.1 – Загальна схема методу спостереження за поглядом очей у відео потоці

Далі розглянемо основну логіку методу. Блок 1 передбачає собою запис початкових даних на стороні клієнта (рис. 2.2).

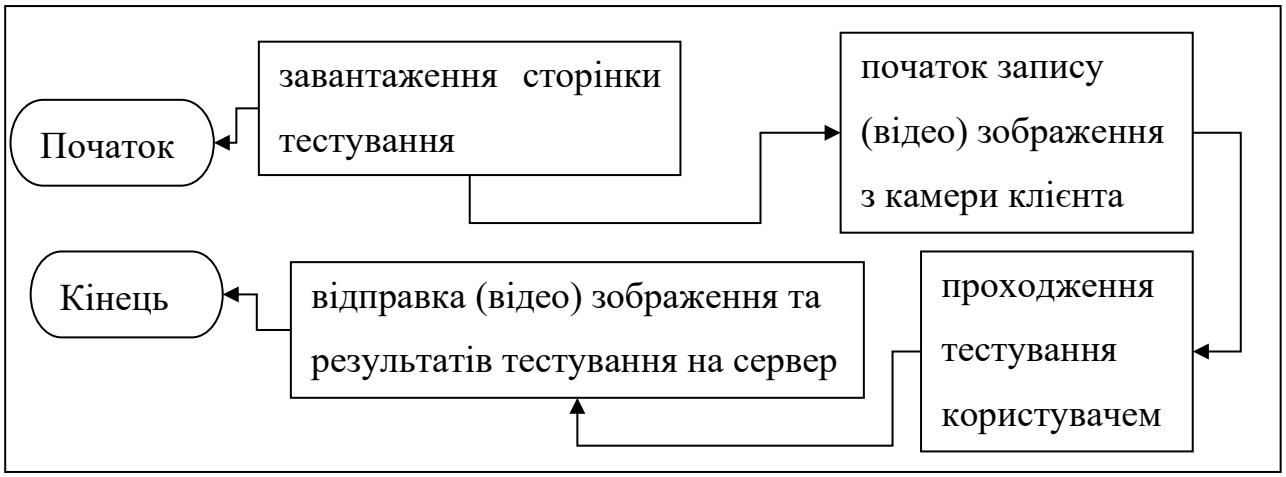


Рисунок 2.2 – Схема роботи клієнтської частини вебзастосунку при проходженні тестування

Послідовність роботи у першому блоці складається з наступних кроків.

1. Завантаження сторінки тестування на стороні клієнта. На цьому етапі користувач розпочинає тестування натиснувши на відповідне посилання. При цьому за шаблоном сторінка збирається на сервері та відправляється клієнтові. Сторінка тестування складається з статичного інтерфейсу проходження тестування, де розміщені завдання, варіанти відповіді та інші елементи керування і скриптів, призначених для відправки даних від клієнта на сервер та запису відео. При формуванні завдань на сторінці кожне завдання перевіряється на наявність правильної відповіді, якщо завдання не має правильної відповіді воно є неправильно сформульованим та не додається до сторінки тестування.

2. Початок запису (відео) зображення з камери клієнта. При завантаженні сторінки скрипт блокує елементи керування та запитує дозвіл на запис відео зображення, якщо запис відео вдалось успішно розпочати елементи керування розблоковуються.

3. Проходження тестування користувачем. Користувач обирає відповіді на завдання тестувань. У нашому випадку завдання можуть бути трьох типів: одна відповідь на вибір, декілька відповідей на вибір або введення відповіді у текстовому полі.

4. Відправка (відео) зображення та результатів на сервер. Процес зупинки запису та відправки відбувається по натисканні за кнопку завершення тесту. При

зупинці запису викликається скрипт, що розбиває відео файл на частини та відправляє кожну частину на сервер в рамках окремого запиту. Результати тестування передаються в тілі запитів з відео зображенням.

У блоці 2 відбувається початкова обробка даних, отриманих від клієнта на стороні серверу (рис. 2.3).

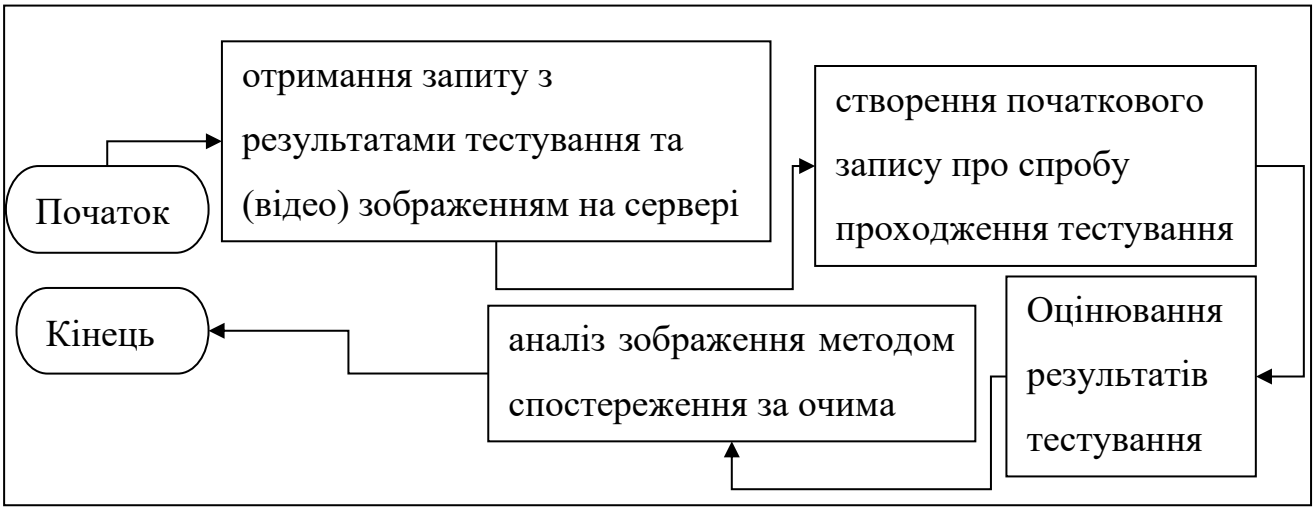


Рисунок 2.3 – Схема початкової обробки даних сервером

Послідовність кроків початкової обробки даних має наступний вигляд.

1. Отримання (відео) зображення та результатів тестування сервером. Сервер приймає запит, в якому містяться результати тестування та файл (відео) зображення. Обробка даного запиту направляється в контролер обробки тестувань. Першим кроком обробки запиту є перевірка наявності файлу, якщо файл знайдено викликається функція збереження у контролері обробки завантаження файлів, що відповідає за збереження файлів на сервері та збірку файлів у випадку завантаження по частинах. Після того як файл було повністю завантажено разом з результатами тестування наступним кроком є обробка результатів тестування та зображення відповідно.

2. Створення початкового запису про спробу проходження тестування. Після того, як (відео) зображення було успішно завантажено на сервер, на основі результатів тестування створюється екземпляр моделей спроби тестування та відповідей для кожного завдання. Модель спроби тестування має такі атрибути,

як тест що був пройдений, користувач що пройшов тест, посилання на файл збереженого (відео) зображення для спостереження за очима та оцінка тестування. Модель відповідей тестування вміщує посилання на спробу проходження тестування, посилання на завдання на яке дається відповідь та обрана відповідь у завданні. Після створення відповідних запитів у БД викликається метод оцінювання конкретної спроби тестування.

3. Оцінювання результатів тестування. Для оцінки тестування використовується окремий метод, вміщений в контролері обробки тестувань, що приймає ID спроби тестування на вході. Першим кроком є запитування моделі нашої спроби тестування, моделей усіх завдань у тестуванні, що проходиться, моделей відповідей у тестуванні та моделей усіх обраних відповідей у спробі тестувань, що асоційовані з нею. Після запиту усіх необхідних моделей необхідно вибрати лише правильно сформульовані завдання, це відбувається за рахунок запиту усіх завдань, що мають хоча б одну відповідь асоційовану з ними, яка поміщена як «правильна» та створити змінну оцінки, що спочатку дорівнює 0. Це відбувається з ціллю відсіювання непідготовлених завдань. Алгоритм проходить по циклу для кожного правильно сформульованого завдання у тестуванні, спочатку перевіряючи чи було дано будь-яку відповідь на нього. Якщо відповідь на існуюче завдання відсутня у спробі тестування здійснюється перехід до наступного завдання у тестуванні. Якщо відповідь існує відбувається перевірка обраної відповіді у спробі, при чому кожен тип завдань (одна відповідь на вибір, декілька відповідей на вибір, текстова відповідь) має дещо різну логіку обробки цієї перевірки. Для одної відповіді на вибір відбувається проста перевірка, чи обрана відповідь збігається з одною з опцій, що позначена як «правильна», якщо відповідь збігається з правильною ми додаємо 1 до нашої оцінки. Для декількох відповідей на вибір ми обчислюємо дробову різницю між набором правильних та обраних відповідей, обраховуючи загальну кількість правильних відповідей, кількість правильних відповідей що були обрані, та кількість неправильних відповідей, що були обрані за формулою 2.1.

$$1 - \frac{\left(N_{\text{правильних}}\right) - \left(N_{\text{правильних обраних}}\right)}{\left(N_{\text{обраних}}\right)} \quad (2.1)$$

Результат обрахунку формули 2.1 дорівнює 1 при виборі усіх правильних відповідей і жодної неправильної, дорівнює 0 при виборі без жодної правильної відповіді та змінюється в залежності від кількості правильних та неправильних відповідей, після чого результат додається до загальної оцінки. Для обрахунку текстової відповіді текст введений користувачем та текст правильної відповіді приводяться до однакової форми піднесенням до верхнього регістру, таким чином при перевірці збігу текстових даних нам не важливо чи щось написано малими літерами чи великими, після чого, якщо відповідь є правильною ми додаємо 1 до значення оцінки. Після завершення оцінювання кожного завдання ми отримуємо деяку оцінку, пропорційну до кількості перевірених завдань, де найвищим значенням є кількість перевірених завдань, ця пропорція потім використовується для обрахунку оцінки у стандартно прийнятих форматах оцінювання, наприклад у відсотковому відношенні, оцінка по п’ятибальній шкалі, дванадцятибальній шкалі, тощо.

4. Аналіз зображення методом спостереження за очима. Після завершення обробки та оцінювання результатів тестування викликається окремий метод для обробки спостереження за очима. Даний метод викликає алгоритм спостереження за очима, що представляє собою Python [25] скрипт, передаючи шлях до файлу як параметр командного рядка та обробляє результати цього аналізу.

Блок 3 є безпосередньо методом для аналізу зображення для спостереження за очима. Аналіз зображення можна поділити на 5 основних кроків (рис. 2.4)

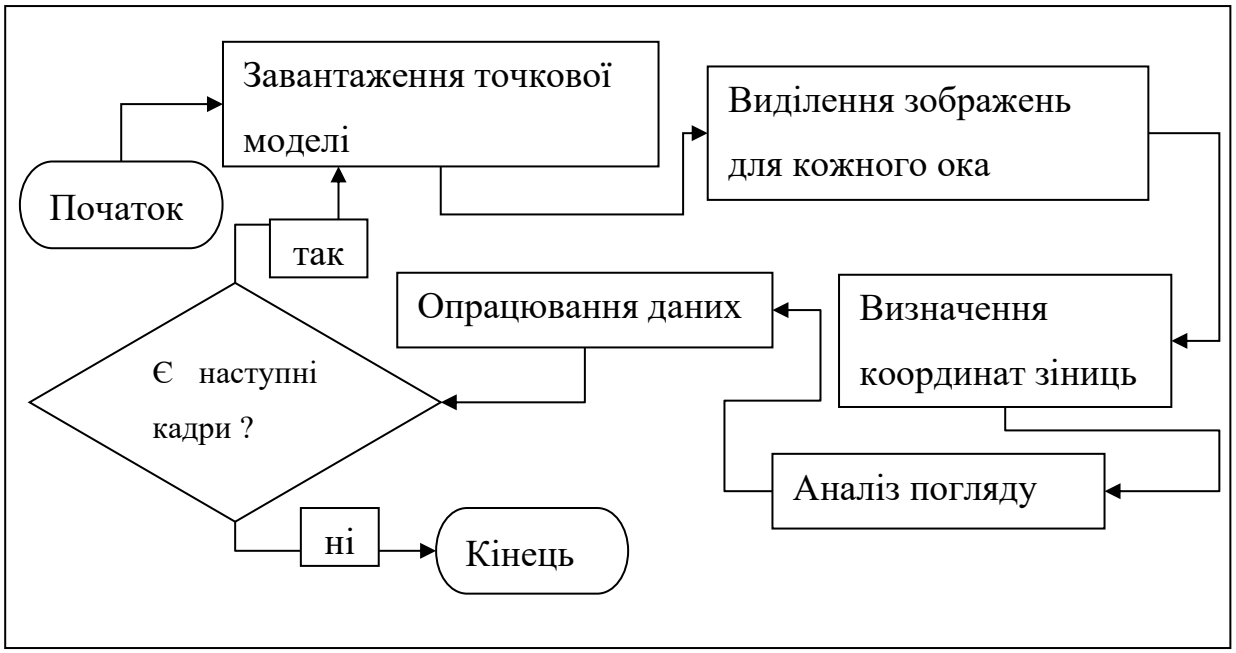


Рисунок 2.4 – Схема для виявлення напрямку погляду на відео зображенні

Наведемо основні кроки для виявлення напрямку погляду на відео зображенні.

1. Першим кроком методу є завантаження точкової моделі. На цьому етапі ми завантажуємо навчену нейронну мережу та подаємо на вхід наше зображення з обличчям, при аналізі нейромережа повертає 68 точкову модель обличчя на зображенні, де кожна точка відповідає деякому елементу обличчя та має визначені координати.

2. Маючи координати елементів обличчя ми можемо виділити частини зображення, що вміщують необхідні точки. В нашому випадку нам необхідно виділити всі точки, що є частиною очей та деякі наближені точки для порівняння позиції зіниць. У нашому випадку нам необхідні точки, що описують контур очей та брів. Для виділення окремих зображень для кожного ока нам необхідно побудувати два прямокутники, що вміщуватимуть усі точки контуру ока та відповідної брови, для кожного ока. Це можна зробити простим порівнянням координат усіх точок для визначення мінімальних та максимальних координат, після чого ми можемо побудувати прямокутник навколо кожного ока. Маючи координати кутів прямокутника ми можемо застосувати функцію обрізання,

отримуючи два прямокутних зображення, що містять усі необхідні нам частини ока для подальшого аналізу.

3. Оскільки точкова модель обличчя не включає в себе зіниці, нам необхідно виділити їх окремим способом. Маючи на початку окремі зображення кожного ока ми можемо знайти позицію зіниці відносно цих зображень за допомогою методу пошуку найтемнішої плями. Для цього нам необхідно підвищувати яскравість кожного зображення з оком, допоки ми не зможемо виділити темну пляму, контури якщо задовольняють умови кругоподібної форми та розміру всередині ока. Маючи контур об’єкта ми можемо обрахувати його центр, знайшовши середнє значення мінімальних та максимальних координат точок контуру. Дану пляму вважатимемо зіницею, а її центр як центр зіниці відповідно.

4. Маючи зображення очей та координати зіниці ми можемо провести аналіз погляду на зображенні. При аналізі конкретного зображення ми можемо виділити відносний напрямок погляду, побудувавши вектор, де центр ока є початковою точкою та центр зіниці задає довжину і напрямку погляду. Окрім напрямку погляду ми також можемо перевіряти, чи є око закритим, тощо.

5. Проаналізувавши одне зображення та отримавши координати очей, зіниць та напрямок погляду ми можемо перевірити, чи дотримується максимальне відхилення очей. Якщо очі відхиляються поза допустимих меж рахуємо довжину не доброчесних фіксацій. Для даної операції першим кроком є визначення середнього значення напрямку погляду для зіниць обох очей, після чого ми перевіряємо, чи ці координати не перевищують максимальне допустиме відхилення. Якщо максимальне допустиме відхилення перевищене, рахуємо часовий проміжок відхилень як час поза екраном. Обрахунок часу відбувається у декілька кроків. На початку, при обробці відео файлу ми зчитуємо кількість кадрів в секунду (FPS). Після цього ми рахуємо кількість кадрів, де наявне відхилення понад норми. Маючи на руках FPS відео зображення та кількість кадрів поза екраном, ми можемо отримати час не доброчесних фіксацій,

поділивши кількість не доброчесних кадрів на FPS, отримуючи час у секундах. Аналогічним методом можна порахувати кількість часу, де очі були закритими.

6. Повернутись до кроку 1 з наступним зображенням, допоки всі кадри в відео не будуть оброблені. Оскільки відео файл складається з великої кількості зображень, що постійно змінюють одне одного створюючи ілюзію руху, доцільним є обробка кожного кадру на зображенні для повноцінної обробки запису з тестування.

7. Python скрипт повертає дані про довжину погляду поза екраном. Після того як кожен кадр у відео файлі було оброблено, у вигляді текстового виходу повертаються такі атрибути, як час поза екраном (у секундах), загальна довжина відео (у секундах) тощо. Маючи ці дані ми можемо у відсотковому співвідношенні порахувати шанс академічної не доброчесності, приймаючи його як частку часу поза екраном до загальної кількості часу.

Блок 4 містить логіку для отримання результату (рис. 2.5).

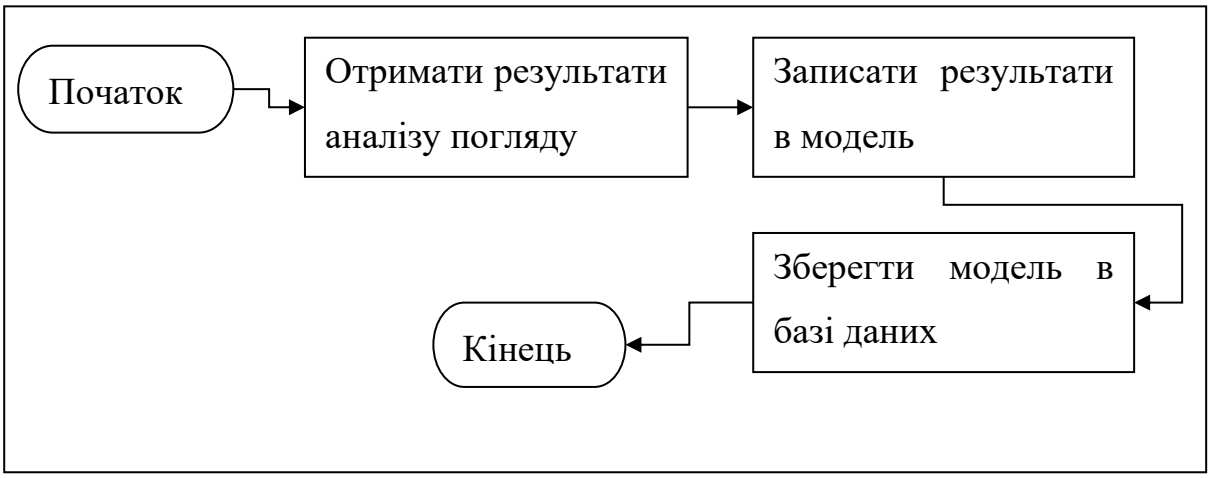


Рисунок 2.5 –Схема для отримання результату за аналізом погляду

Далі наведена послідовність кроків для отримання результатів за аналізом погляду.

1. Дані, отримані аналізом (відео) зображення записуються в контролері для попередньо створеного об’єкту результатів тестування. Після завершення аналізу погляду зображенні дані Python скрипту виводяться у формі текстових рядків у командній консолі, де кожен рядок відповідає за окрему змінну.

Приймаючи ці дані ми можемо обрахувати та записати у моделі спроби тестування час поза екраном, час відео, тощо.

2. Об’єкт, що вміщує результати тестування та результати роботи методу спостереження за очима зберігається в БД для подальшого відображення. Це відбувається за допомогою виклику функції збереження змін до моделі бази даних.

3. При запиті результатів тестування, дані, отримані методом аналізу погляду та результати тестування, відображаються на сторінці результатів. Відповідні дані про оцінку та результати аналізу погляду запитуються у базі даних, після чого відбувається збірка елементів вебсторінки, де відображається таблиця, що описує кожне завдання у спробі, чи була відповідь на завдання правильною, кінцева оцінка у відсотковому форматі та час поза екраном при проходженні тестування.

Отже, було запропоновано метод спостереження за поглядом у відео потоці, результати роботи якого будуть використання у системі тестування знань.

2.4 Особливості використання спеціалізованих програмних компонентів

Для забезпечення відповідного функціоналу системи тестування знань з функціоналом спостереження за очима нам необхідно застосувати програмні компоненти, що нададуть можливість створити його реалізацію.

Маючи вебсистему побудовану з використанням фреймворку Laravel [26], забезпечення процесу запису відео на стороні клієнта та його відправлення на сервер може бути реалізовано за допомогою локального JavaScript.

Конкретно для запису відео ми можемо застосувати об’єкт *MediaRecorder*, що записуватиме відео потік у пам’яті клієнта у формі частинок (chunks) допоки тестування не буде завершене, для чого реєструється подія завершення запису, що автоматично викликатиме відправлення відео файлу на

сервер. При натисканні на кнопку завершення тестування, викликається метод, що перериває запис відео. У свою чергу подія переривання запису викликає метод відправлення відео файлу на сервер.

Для відправлення великого файлу на сервер ми спочатку формуємо *Blob* даних, що представляє собою файл, зібраний з усіх чанків, який ми будемо відправляти, після чого можемо сформувати запит до серверу з даними форми для тестування. Для відправки великого файлу по частинам застосуємо об'єкт *Resumable*, що дозволяє додавати файл до запиту після чого він буде відправлений по частинам на сервер *resumable.addFile(file)*. Об'єкт Resumable підтримує реєстрацію подій, завдяки чому ми можемо відслідковувати прогрес завантаження та створити смугу прогресу *resumable.on('fileprogress'…*

Для отримання даних на стороні серверу нам необхідно використати Laravel контролери, що обумовлено архітектурою MVC даного фреймворку. При початковій обробці запиту на збереження результатів тестування нам необхідно пересвідчитися у тому, що цей запит був надісланий авторизованим користувачем, отже нам необхідно застосувати перевірку авторизації за допомогою бібліотеки *Auth,* де нам необхідний метод *Auth::check*, що повертає булеве значення *True* при успішній авторизації та *False* при помилці авторизації.

Для обробки завантаження файлу в окремому контролері для початку нам необхідно застосувати функцію *App::call*, що дозволяє викликати метод у деякому іншому контролері з будь якої точки в контролері. Для обробки завантаження файлу по частинах нам необхідно використати зовнішню бібліотеку *pion/laravel-chunk-upload*, що надає об'єкт *FileReceiver*, який дозволяє завантажувати файл по частинах викликаючи метод *receive()* для конкретного об'єкту *FileReceiver*. При збірці файлу ми можемо перевірити, чи його було повністю завантажено після завантаження кожної частинки використовуючи метод *isFinished()*, після чого ми можемо зберегти зібраний файл у деякому місці на диску серверу використовуючи метод *move(),* що приймає шлях у файловій системі та назву файлу, що зберігається як параметри.

Після завершення завантаження та збереження файлу ми можемо повернути шлях до нього у файловій системі. Для повернення структурованого набору даних ми можемо застосувати метод *response()->json*, параметрами якого буде набір деяких текстових параметрів.

Отриману відповідь у форматі json ми можемо перетворити на список змінних використати вбудований в мові програмування PHP метод *json_decode*. Після цього ми можемо виділити будь який параметр за його назвою для подальшого опрацювання, наприклад *array['param_name']*.

Для збереження моделей нам необхідно застосувати метод *save()* для об'єкта моделі. При перевірці завдань нам необхідно завантажити модель із бази даних, для цього необхідно використати метод *model::find()*, що приймає ID запису як параметр.

При пошуку запису у БД можна сформувати запит з конкретною умовою за допомогою методу *where()*, що дозволяє повернути лише ті запити, що задовольняють умову на вході даного методу.

У фреймворку Laravel є можливість задання глобальних налаштувань у файлі *.env* , доступ до даних налаштувань можна отримати відповідним методом *env()*, що повертає значення конкретної змінної в файлі. Цей файл використовується для налаштування серверу вебсистеми та задання посилань на сторонні ресурси, що можуть бути використані сервером.

Оскільки алгоритм спостереження за очима є окремим Python скриптом, нам необхідно запустити його в середовищі контролера Laravel. Для цього ми можемо викликати скрипт за допомогою командного рядка, передаючи посилання на відео файл, що обробляється як додатковий параметр. Для цього в середовищі контролера Laravel нам необхідно використати метод shell_exec(), що приймає команду як вхідний параметр та повертає текстовий вивід консолі.

Для розбору кожної окремої стрічки виводу консолі при застосуванні shell_exec() ми можемо застосувати метод explode(), що розділяє текстові дані за деяким символом. У нашому випадку нам необхідно розділити вивід консолі на кожну окрему стрічку, отже ми будемо розділяти текст символом нового рядка

explode("\n", $result). Таким чином ми зможемо проаналізувати кожен вихід нашого скрипту як окрему величину.

Переходячи до самого Python скрипту, що викликатиметься з консолі найпершою бібліотекою яка на потрібна являється sys. Ця бібліотека надає нам можливість зчитувати параметри командного рядку sys.argv[], де в квадратних дужках вказується індекс параметру та завершити виконання скрипту використовуючи sys.exit().

Для зчитування відео потоку та його обробки ми застосуємо бібліотеку OpenCV. Ця бібліотека надає функціонал зчитування кадрів відео потоку, зміну кольорів, обрізання та об'єднання зображень, тощо. Для отримання відео потоку ми можемо застосувати метод *cv2.VideoCapture()*, де параметром методу є посилання на відео файл у файловій системі, при цьому результат виконання методу повинен бути збережений у деяку змінну для подальшої обробки відео потоку. Після цього для зчитування кадру з потоку ми застосовуємо *videoStream.Read()*, де *videoStream* представляє собою деяку змінну якій було присвоєно результат *cv2.VideoCapture()*. При успіху зчитування кадру цей метод повертає булеву змінну що сигналізує про успіх та конкретне зображення (кадр) у відео потоці.

Для роботи з файловою системою нам необхідно використати бібліотеку *os*, а саме *os.abspath()* для пошуку глобальної адреси файлу у файловій системі. Це потрібно для отримання посилання на використовуваний файл натренованої 68 точкової моделі розпізнавання облич.

Для роботи з алгоритмами машинного навчання нам знадобиться бібліотека *dlib*. Основним її застосуванням буде аналіз зображення існуючою моделлю розпізнавання облич. Для початку роботи з розпізнаванням облич застосуємо вбудовану модель розпізнавання облич *face_detector=dlib.get_frontal_face_detector()*. Ця модель лише виділяє позицію обличчя на зображенні та не розпізнає конкретних його елементів. Для завантаження точкової моделі бібліотекою ми інстанціюємо

predictor=dlib.shape_predictor(model), де параметр моделі представляє собою шлях до файлу навченої моделі у файловій системі.

При аналізі зображення спочатку переведемо його в чорно-білий формат за допомогою методу *frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)* де *frame* представляє собою кадр, що обробляється, а *cv2.COLOR_BGR2GRAY* являється назвою кольорової схеми до якої ми приводимо зображення.

Використовуючи вбудовану модель розпізнавання облич бібліотеки *dlib* ми можемо використати попередньо названий метод *faces=face_detector(frame)*, таким чином отримуючи усі обличчя наявні на зображенні. Використовуючи ту ж бібліотеку ми можемо застосувати попередньо завантажену точкову модель *landmarks = predictor(frame, faces[0])*, де параметрами ми передаємо кадр на якому застосовується точкова модель frame та деяке конкретне обличчя на зображенні, оскільки обличчя у нас повинно бути тільки одно тому обираємо *faces[0]*, що відповідає найпершому обличчю у списку розпізнаних облич. На виході метод точкової моделі повертає список розпізнаних на обличчі точок, що можуть бути використані у подальшому аналізі зображення.

Оскільки обрана точкова модель при розпізнаванні елементів обличчя має статичне нумерування, тобто кожен розпізнаний елемент обличчя має конкретний номер точки, таким чином для виділення лише тих точок, що відповідають очам на зображенні достатньо мати список ID точок, що відповідають кожному оку як константу. Для цих цілей було застосовано метод *list()*, що перетворює параметри на вході у список та метод *range()*, що створює послідовність чисел між мінімальними та максимальними значенням що задаються у параметрах методу при його виклику. Разом це матиме вигляд *LEFT_EYE_POINTS = list(range(36, 42))+list(range(17,22))*. Маючи два набори точок, що належать кожному оку ми можемо виділити частину зображення, що вміщує лише наявні точки.

Для роботи з масивами та матрицями в Python використовується бібліотека numpy. Маючи деякий список id точок, що нам необхідно виділити, ми можемо перетворити їх у масив координат кожної точки за допомогою

numpy.array(), де вхідними параметрами є пари координат для кожної точки, отримані використовуючи доступ до атрибуту *part* у об’єкті розпізнаних точок *landmarks*, наприклад *landmarks.part(point).x*, при чому значення point представляє собою id точки, яка нам потрібна.

Після виділення усіх необхідних точок ми можемо ізолювати зображення для кожного окремого ока. Для цього необхідним створення маски, що ефективно видалить з зображення усі елементи, окрім тих, що нам потрібні. Першим кроком для цього є створення матриці зображення відповідного розміру, де кожен пік сель має чорний колір, використовуючи метод numpy.zeros((height, width)), де height та width представляють собою висоту та довжину зображення у пік селях відповідно. Після цього ми можемо створити нашу маску, що виділятиме контур необхідних нам елементів. Для початку створимо окремо матрицю білих пікселів використавши метод numpy.full((height, width), 255). Після цього ми можемо ми можемо застосувати побудову полігону cv2.fillPoly(mask, [region], (0, 0, 0)), де mask – це матриця білого зображення, а region – масив точок, що нам необхідні для спостереження за очима. Як результат отримуємо зображення з деякою чорною фігурою, що містить усі виділені елементи необхідні для подальшого аналізу зображення, у нашому випадки ми виділяємо око та брову над ним. Після цього ми можемо застосувати метод *cv2.bitwise_not(black_frame, frame.copy(), mask=mask)*, що перетворює все зображення копії нашого кадру на чорне, залишаючи лише фігуру, що окреслює конкретне око, де серед вхідних параметрів *black_frame* представляє собою матрицю чорних пікселів, *frame.copy()* є копією поточного кадру у відео та mask представляє нашу фігуру, що окреслює контур ока.

Для нарізки зображень на частини, що містять лише одне око ми можемо застосувати пошук максимальних та мінімальних координат точок таким чином, щоб кожна точка не виходила за межі зображень. Для цього застосуємо функціонал бібліотеки numpy використавши методи *numpy.min()* та *numpy.max()* відповідно. Але, обрізання по граничній точці може призвести до втрати необхідних для роботи моделі точкового розпізнавання даних, отже нам

необхідно необхідно додати деякі межі, наприклад *numpy.min(region[:, 0]) – margin,* де region це набір точок нашого контуру, а margin – деяке значення відступу від граничних значень в пікселях.

Маючи мінімальні та максимальні координати, ми можемо виділити зображення, що містить лише очі використовуючи побудову прямокутника за двома точками. Маючи наше зображення з виділеними очима *eye* ми можемо скоротити його розмір використовуючи координати мінімумів та максимумів *eye[min_y:max_y, min_x:max_x]*, отримавши на виході зображення, що містить лише конкретно обране око. Процес виділення другого ока відбувається аналогічним чином.

Після виділення частини зображення з очима, ми можемо ізолювати положення зіниці у оці. Для цього нам необхідно застосувати метод cv2.bilateralFilter(), що на вході приймає деяке зображення та параметри чутливості фільтру, а на виході повертає зображення, що складається лише з двох кольорів. Для підвищення якості роботи подальшого алгоритму використаємо функцію cv2.erode(), що зменшує різницю кольорів між сусідніми пікселями, що в результаті дає більш рівномірну фігуру. На кінець ми виділяємо зображення зіниці використовуючи відсікання кольорів методом cv2.threshold(), основними параметрами цього методу є зображення, що обробляється, межі відсікання кольорів та тип відсікання (наприклад cv2.THRESH_BINARY). На виході ми отримуємо зображення, що вміщує лише ту частину, яка вважається зіницею. Для оцінки правильності визначення координат зіниці ми можемо порівняти кількість пікселів що мають колір з загальною кількістю пікселів, для цього ми можемо застосувати метод cv2.countNonZero(frame), де frame – це наше зображення що повинно містити зіницю. Якщо зіницю не знайдено, тоді повторюємо процес її виділення змінивши параметри відсікання кольорів.

Отже, було описано особливості використання різних бібліотек та методів для середовищ фреймворку Laravel та мови програмування Python для реалізації методу спостереження за очима для вебсистеми тестування знань.

2.5 Архітектура вебсистеми тестування знань

Вебсистема для тестування знань (рис. 2.6) побудована за принципами MVC з використанням фреймворку Laravel.

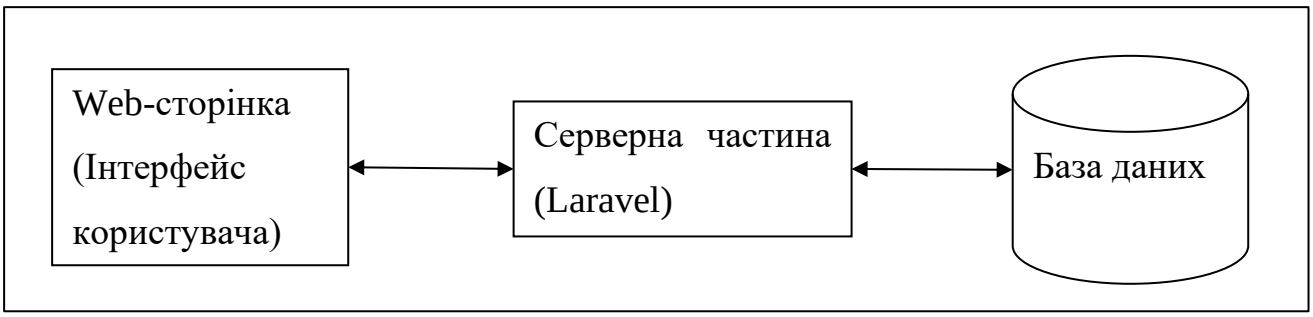


Рисунок 2.6 – Структура компонентів вебсистеми

Шаблон MVC подає застосунок у вигляді трьох основних компонент: моделі, вигляду та контролеру (рис.2.7).

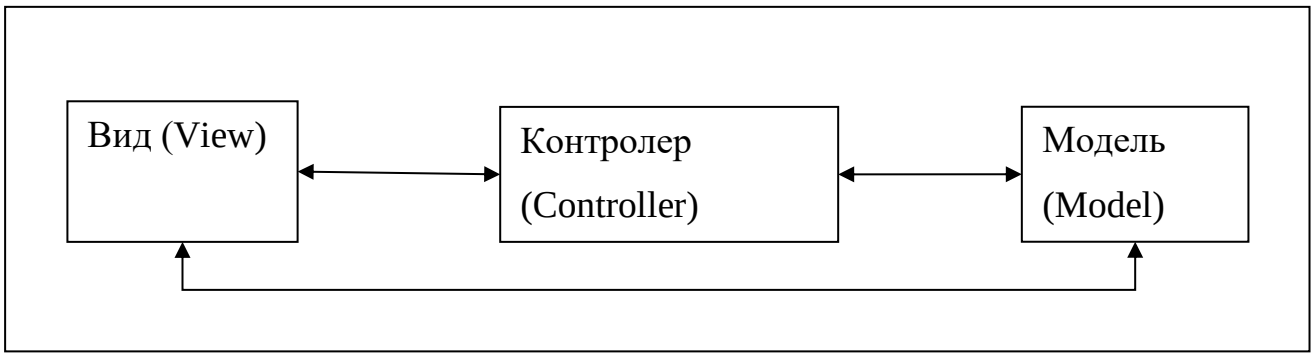


Рисунок 2.7 – Структура шаблону проєктування MVC

Види є деякими шаблонами вебсторінки, що збирається сервером по запиту клієнта. В рамках розроблюваного методу спостереження за очима основні види є сторінками для проходження тестування та сторінками перегляду результатів тестування. При цьому для сторінки проходження тестування варто звернути увагу на наявність логіки запису відео зображення на стороні клієнта та подальше його відправлення на сервер.

Контролери містять основну логіку обробки запитів, що поступають на сервер. Серед контролерів нам необхідно обробити запит на збереження

результатів тестування та аналіз зображення для спостереження за очима. Для обробки запиту на збереження результатів тестування ми створимо *ResponseController*, де міститиметься алгоритм оцінювання результатів тестування та обробка методу спостереження за очима. Оскільки завантаження відео файлів або встановлення прямого відео потоку потребує окремої обробки за рахунок необхідності завантаження по частинам, логічним є винесення логіки завантаження файлів по частинах у окремий *ChunkUploadController* (рис. 2.8). Після отримання відео зображення у повному об’ємі ми можемо приступити до перевірки результатів тестування. Для перевірки результатів тестування необхідно перевірити кожне завдання у тестуванні, в зв’язку з чим нам необхідно виділити логіку перевірки одного завдання у тестуванні в *ResponseAnswerController*. Після чого можна запустити алгоритм спостереження за поглядом.

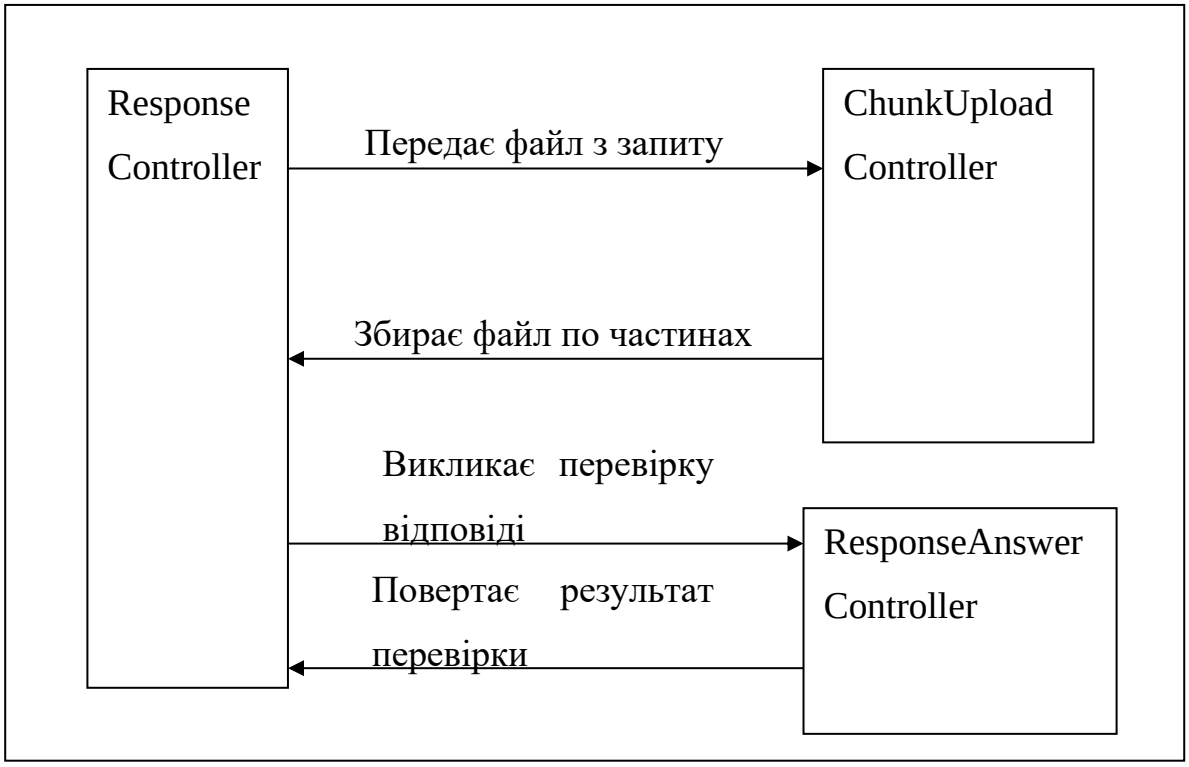


Рисунок 2.8 – Схема взаємодії контролерів при запиті збереження результатів тестування

Отже, у даному шаблоні моделі описують об’єкти та їх відношення у базі даних, контролери вміщують в собі основну логіку обробки запитів до серверу та взаємодію з БД, а вигляди є шаблонами сторінок, що збираються на стороні серверу та відображаються на стороні клієнту.

2.6 Модель бази даних для системи тестування знань

Моделі, з розглянутого у попередньому підрозділі шаблону, є описом сутностей та їх зв’язків у БД для формування запитів у вибраній СКБД. У нашому випадку, окрім сутностей, що описують структуру завдань на тестуванні, основними сутностями є користувач, одна спроба користувача, одно завдання у спробі користувача та відповідь на питання у спробі кожна з яких відповідає таблиці у базі даних (рис. 2.9). Результати обробки відео зображення на предмет погляду поза екраном записуються у вигляді загального часу поза екраном в таблиці спроби користувача, оскільки кожна окрема спроба повинна бути проаналізована методом спостереження за очима.

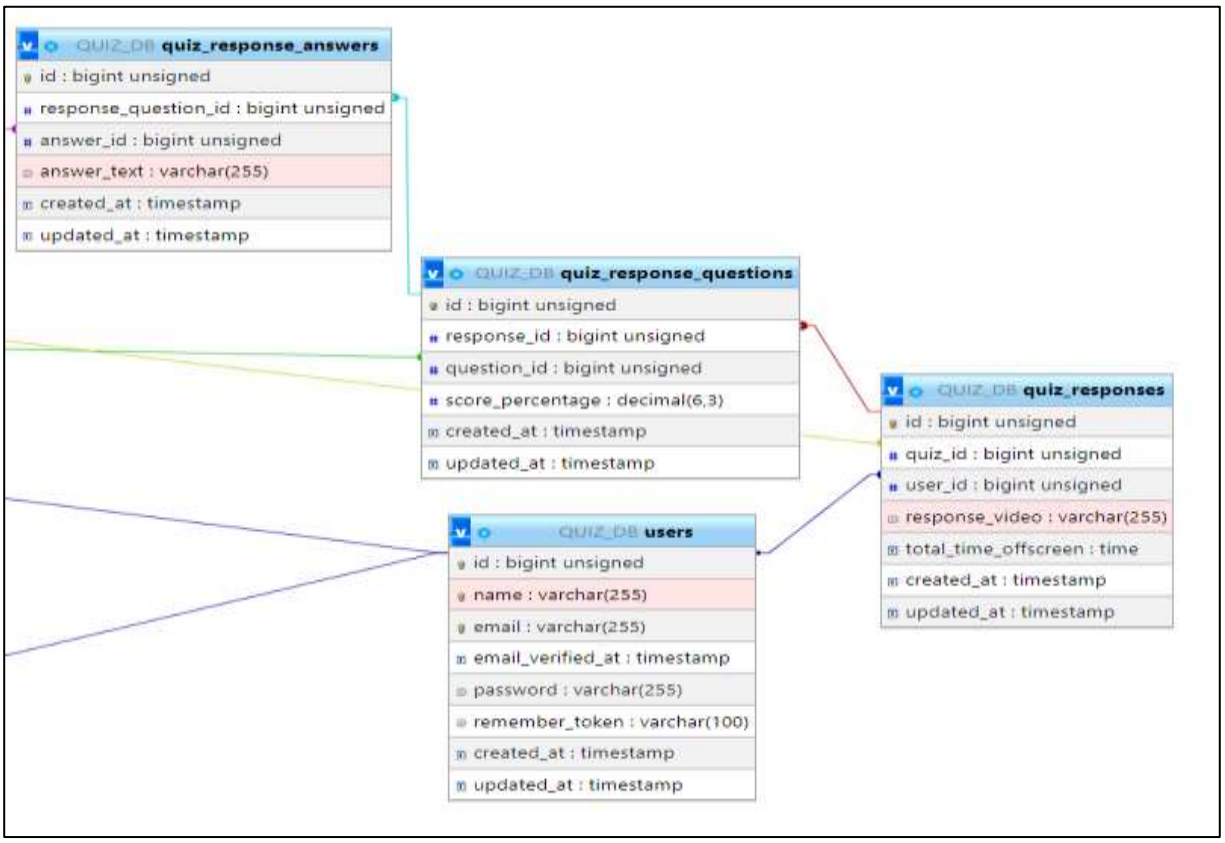


Рисунок 2.9 –Основні таблиці у структурі БД

Для функціонування системи тестування знань нам необхідно мати дані про існуючі тестування, предмети, спроби проходження тестувань, тощо. Для збереження та опрацювання необхідних даних було прийнято рішення використання СКБД MySQL [27] у системі тестування знань (рис. 2.10).

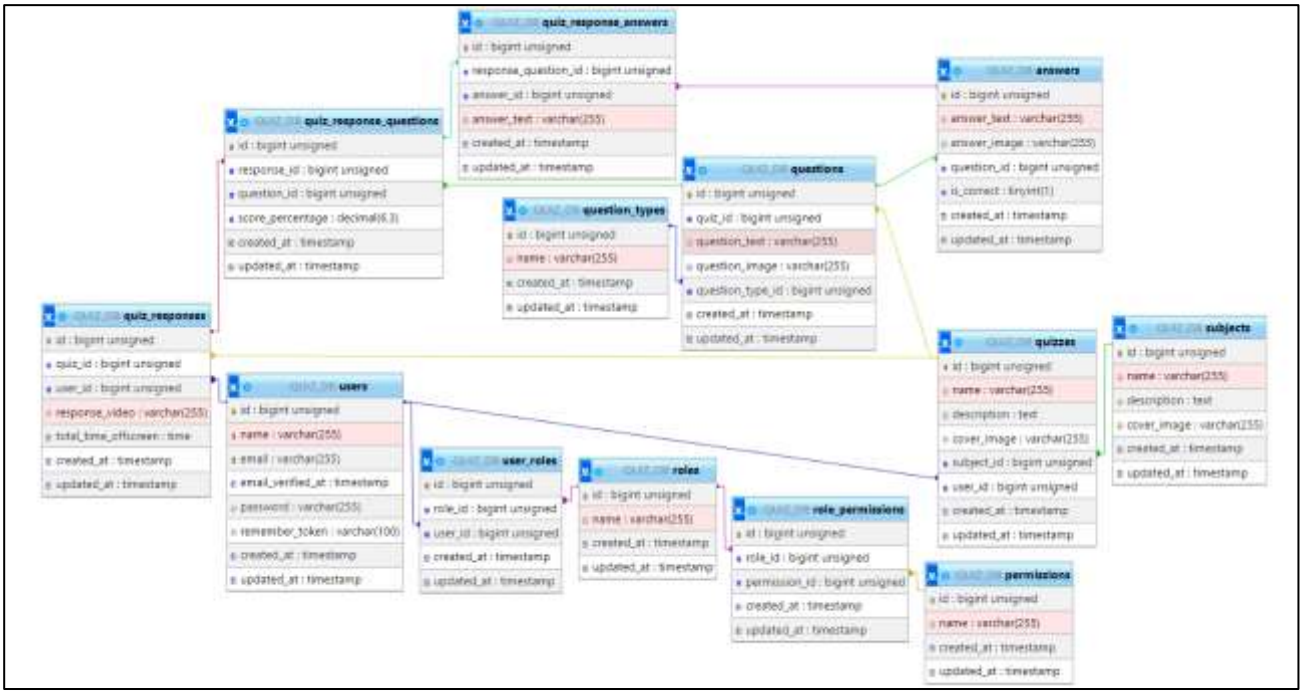


Рисунок 2.10 – Структура БД системи тестування знань

Для збереження даних про окремі тестування використовується таблиця *quizzes*, де основними параметрами є назва тестування *name*, опис тестування *description*, обкладинка тестування *cover_image*, посилання на предмет до якого належить тестування *subject_id* та посилання на користувача що створив тестування *user_id*. Також тут присутні параметри, що описують час створення та час внесення змін в запис у таблиці *created_at* і *updated_at* відповідно. Тестування можуть бути створені, редаговані та видалені в процесі використання системи. Доступ до функціоналу редагування та створення тестувань мають лише користувачі, що авторизовані як викладач або адміністратор. Запит на взаємодію з таблицею автоматично формується сервером, додатково зберігаючи дані про дату створення запису та дату останнього його редагування. При

редагуванні існуючого тестування сформований запит заміщує лише ті поля, що було змінено.

Маючи запис про тестування, нам необхідно категоризувати його, наприклад за предметом *subjects*. Таблиця предметів має такі параметри, як назву предмету *name*, обкладинка предмету *cover_image*, опис предмету *description*. Записи у даній таблиці можуть бути створені, редаговані та видалені лише користувачами, що авторизовані як адміністратор системи. При створенні предмету також записується його час створення та час останніх змін. При редагуванні запит відбувається лише до полів, що були змінені, та включає в себе час оновлення запису.

Кожне тестування створено деяким користувачем, отже нам необхідна таблиця *users*. Параметрами цієї таблиці є ім'я користувача *name*, адреса електронної пошти *email*, час підтвердження адреси електронної пошти *email_verified_at*, пароль password та код доступу *remember_token* для швидкого входу без введення паролю. Дана таблиця використовується сервером для авторизації користувачів. В рамках роботи вебсистеми доступ до цієї таблиці здійснюється лише для авторизації користувача та створення асоціаційних зв’язків між існуючими користувачами, тестуваннями що були ними створені та спроб проходження тестувань. Будь-які користувачі не мають прямого доступу до записів у цій таблиці.

Не кожен користувач повинен мати доступ до всього функціоналу створення, редагування та видалення об'єктів тестування, тому для розподілу доступів між користувачами необхідна таблиця *user_roles*. Ця таблиця розподіляє існуючі ролі між користувачами. Її основними параметрами є посилання на користувача *user_id* та посилання на існуючу роль *role_id*. При реєстрації нового користувача йому автоматично надається роль учня. Нові ролі можуть бути надані адміністратором вебсистеми.

Таблиця *roles* представляє список наявних ролей, що можуть бути надані користувачу. Її головним параметром є назва ролі *name*. Доступ до запитів до цієї таблиці здійснюється лише за умови наявності доступу адміністратора.

Для забезпечення можливості надання різних рівнів доступів для різних ролей без необхідності переписувати код застосунку ми зберігаємо список дозволених операцій для кожної ролі *role_permissions*. Призначенням цієї таблиці є об'єднання існуючих ролей з існуючими дозволами за параметрами *role_id* та *permission_id* відповідно. Доступ до цієї таблиці є частиною роботи з існуючими ролями, доступ до неї здійснюється лише за наявності прав адміністратора.

Таблиця *permissions* зберігає записи про можливі дозволи у системі. Кожен окремий дозвіл описується лише ім'ям *name*. В рамках роботи системи тестування знань ця таблиця не отримує будь-яких запитів після початкового інстанціювання дозволів та слугує для перевірки наявних у користувача доступів у системі.

Кожне тестування передбачає деякий набір завдань, для чого нам необхідно зберігати дані про кожне завдання у таблиці questions. Параметри цієї таблиці містять посилання на тестування до якого належить завдання *quiz_id*, текст завдання *question_text*, зображення завдання *question_image* та тип завдання *question_type_id*. Редагування завдань потребує ролі вчителя або адміністратора, аналогічно до редагування тестувань.

Для забезпечення можливості існування завдань різних типів маємо таблицю *question_types*, де зберігаємо назву типу завдання *name*. Подібно до таблиці доступів, усі записи у даній таблиці створюються лише один раз при налаштуванні системи та є статичними, тобто не повинні змінюватись у процесі її роботи.

При перевірці завдань нам необхідно мати список правильних та неправильних відповідей, для чого було створено таблицю *answers*, де містяться дані про конкретні відповіді на завдання. Її основними атрибутами є текст відповіді *answer_text*, зображення відповіді *answer_image*, посилання на завдання до якого відноситься *question_id* та помітка про правильність відповіді. Визначення відповідей на завдання є процесом створення тестування знань, отже

доступ до цього функціоналу повинен бути наявний для тих самих ролей, у нашому випадку це буде роль вчителя та адміністратора.

Для збереження результатів спроби тестування використовується таблиця *quiz_responses*. Її основними параметрами є посилання на тестування, що було пройдено *quiz_id,* посилання на користувача, що пройшов тест *user_id,* відео зображення спроби тестування *response_video* та загальний час поза екраном, отриманий за допомогою обробки зображення методом спостереження за очима *total_time_offscreen*. Запити до цієї таблиці не є явними для користувача, що проходить тестування та формуються автономно в залежності від відповідей, що були обрані користувачем під час тестування. Параметр *total_time_offscreen* може бути порожнім, оскільки нам необхідно створити запис перед використанням алгоритму спостереження за очима. При збереженні спроби проходження тестування спочатку створюється початковий запис, що містить лише посилання на тестування, користувача що здійснив спробу та шлях до відео файлу. Після аналізу погляду окремим запитом додається загальний час поза екраном.

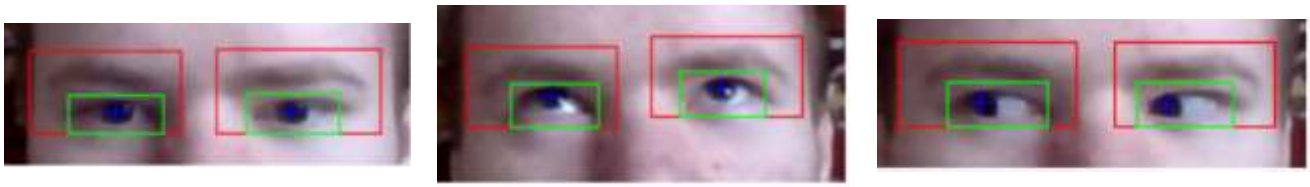
При обробці даних спроби нам необхідно дати оцінку правильності відповідей. У таблиці *quiz_responses_questions* ми зберігаємо дані про завдання, що були наявні у спробі та оцінка виконання для кожного завдання. Основними параметрами цієї таблиці є посилання на спробу тестування response_id, посилання на існуюче завдання у тестуванні question_id та оцінку правильності завдання score_percentage. Таким чином, для кожної спроби тестування ми можемо обрахувати загальну оцінку, комбінуючи оцінку кожного завдання у спробі. Аналогічно до спроби тестування спочатку створюється запис з усіма даними окрім оцінки завдання, що додається окремим запитом після перевірки.

Таблиця quiz_response_answers зберігає дані про конкретно вибрані відповіді у тестуванні та може бути використана для обрахунку оцінки і подальшого відображення обраних відповідей після обробки результатів спроби. Основними параметрами таблиці є посилання на завдання у спробі *quiz_response_question*, посилання на обрану відповідь у тестуванні *answer_id* та

текстова відповідь на завдання *answer_text*. При початковому збереженні результатів тестування ми автоматично записуємо кожну відповідь дану користувачем у спробі. Після збереження ми можемо здійснити запит на вибірку всіх відповідей для конкретної спроби та обрахувати оцінку.

2.7. Результати виявлення погляду

На рис. 2.11 зображені випадки визначення напрямків погляду за якими приймається рішення про академічну доброчесність під час тестування знань.



а) погляд прямо б) погляд вгору в) погляд вбік

Рисунок 2.11 – Результати виявлення напрямку погляду

2.8 Висновки до розділу 2

У розділі було запропоновано метод спостереження за очима у відео потоці, який вбудовано у вебсистему тестування знань для контролю академічної доброчесності під час процесу проходження тестувань.

Було наведено методом спостереження за очима у відео потоці, архітектуру вебсистеми та структуру бази даних.

Запропонований метод дозволятиме реалізувати академічну доброчесність під час проходження тестувань за аналізом погляду шляхом обрахунку кількості часу, коли погляд знаходиться поза екраном.

Розділ 3 Програмна реалізація вебсистеми тестування знань з вбудованим методом спостереження за очима

У даному розділі висвітлено загальну архітектуру, особливості реалізації вебсистеми тестування знань, інтеграції методу спостереження за очима у вебсистему та опис розробленої вебсистеми.

3.1 Загальна архітектура вебсистеми тестування знань

Вебсистема тестування знань побудована з використанням шаблону MVC. Даний шаблон поділяє систему на три види основних компонентів: model (моделі), view (види) та controller (контролери). Моделі є шаблонами сутностей, що використовуються при взаємодії з базою даних. Види вміщують в собі деякі елементи вебсторінки та шаблони їх збірки, що збираються на стороні серверу перед відправкою до клієнта. Контролери призначені для обробки запитів клієнта та вміщують в собі основну бізнес-логіку.

Для забезпечення постійного стилю вебсторінок без необхідності повторного визначення параметрів та елементів кожної сторінки було використано загальний шаблон вебсторінки (рис. 3.1).

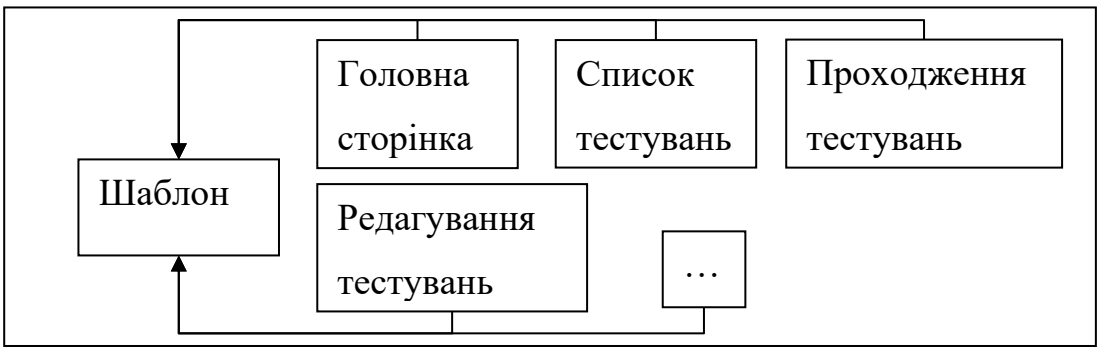


Рисунок 3.1 – Використання шаблону для побудови сторінок

Основний шаблон (рис. 3.2) включає в себе такі елементи, як назву сторінки, логотип сторінки, стилі елементів вебсторінки, верхня панель керування (header), нижня інформаційна панель (footer), та основний вміст

сторінки (body). Стилі елементів, нижня інформаційна панель та логотип є статичними елементами, що залишаються однаковими незалежно від обраної сторінки. Верхня панель керування містить посилання на основні сторінки вебресурсу, назву сторінки та елементи взаємодії з обліковим записом користувача (кнопки реєстрації та входу для неавторизованого користувача та випливаючи меню взаємодії для авторизованого користувача). Основний вміст сторінки у шаблоні включає в себе лише посилання, для заповнення сторінки елементами при її збірці.

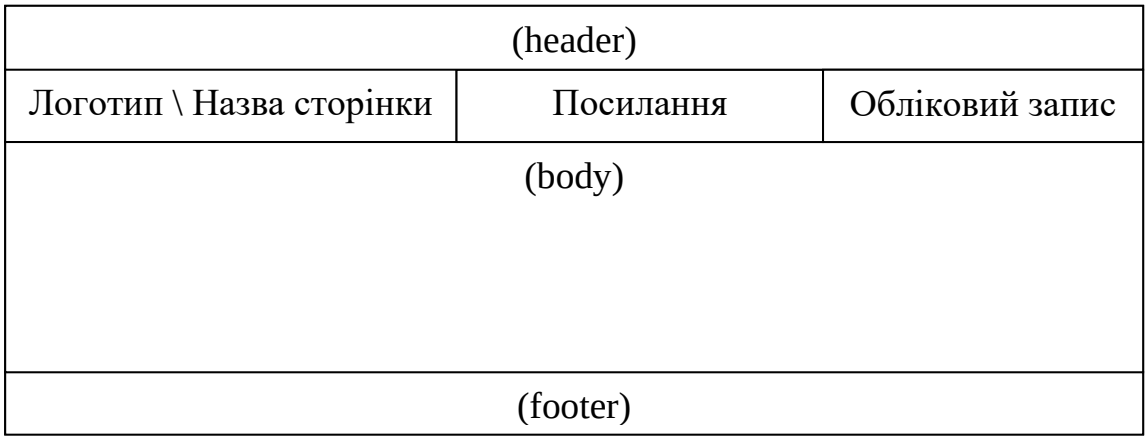


Рисунок 3.2 – Схематична структура шаблону вебсторінки

При обробці запиту клієнта відбувається збірка сторінки за деяким файлом виду. Такий файл вміщує загальну структуру сторінки та\або посилання на використовуваний шаблон та його частини, що заповнюються вмістом у межах даного виду.

Кожен окремий контролер відповідає за обробку операцій пов’язаних з деякою сутністю (рис. 3.3) або вміщує унікальний, незалежний від сутностей функціонал. Побудова сторінок відбувається за допомогою контролерів та моделей. При обробці запиту контролер повертає деякий набір даних, що може використовуватися для заповнення шаблону вебсторінки.

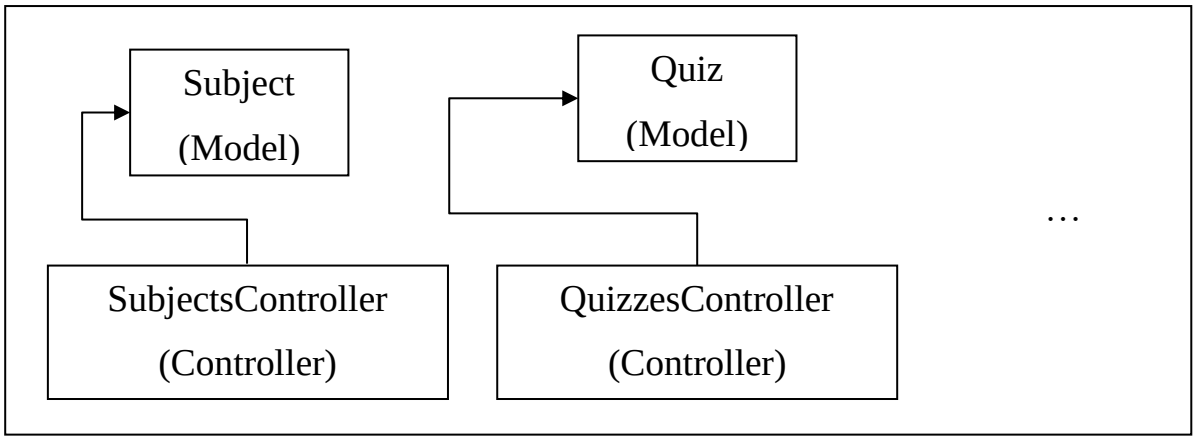


Рисунок 3.3 – Більшість контролерів призначені для операцій над однією сутністю

При цьому варто зазначити, що під час обробки запиту деякі контролери можуть взаємодіяти між собою. В рамках розроблюваної системи ми використовуємо окремий контролер для обробки завантаження файлу по частинах ChunkUploadController (рис. 3.4), функціонал якого може використовуватися при обробці запитів в інших контролерах, наприклад ResponseController.

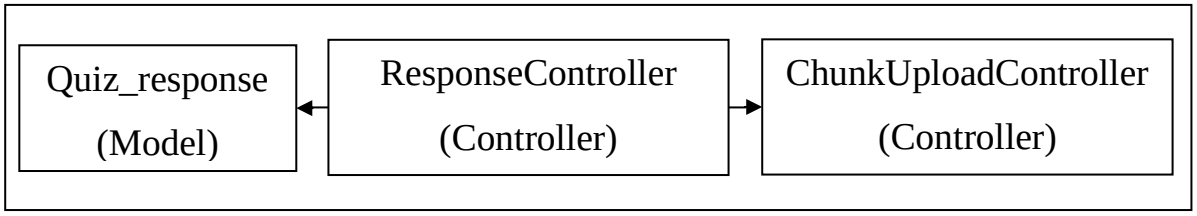


Рисунок 3.4 – Деякий функціонал винесено в окремий контролер

Маючи контролери, моделі та види можемо скласти приблизну схему взаємодії компонентів (рис. 3.5).

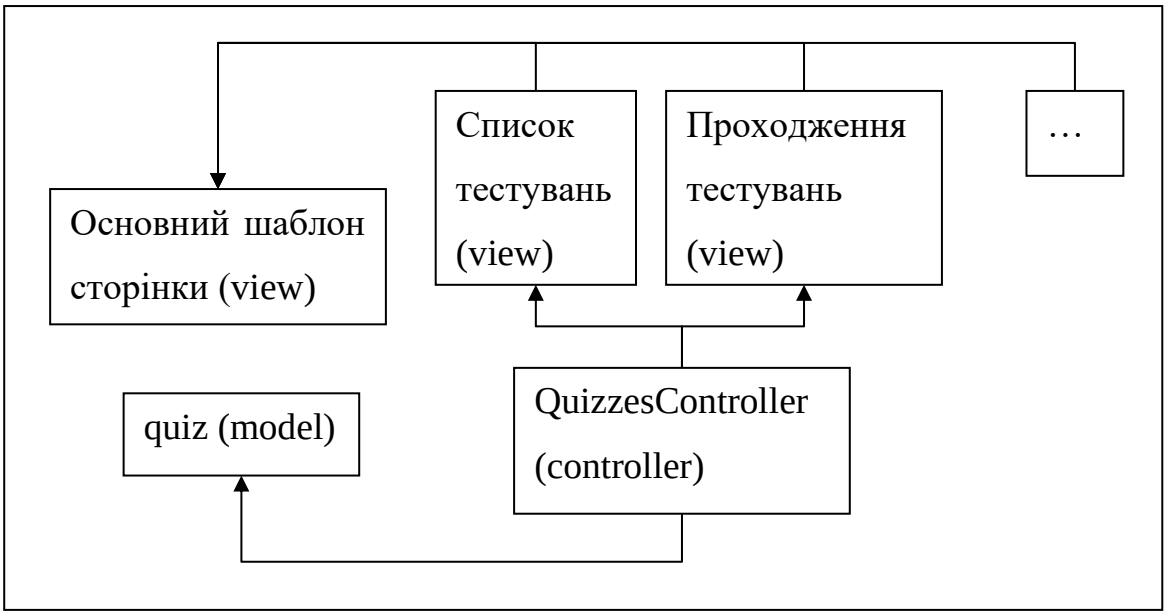


Рисунок 3.5 – Загальна схема взаємодії компонентів системи

Отже, під час обробки запиту контролер використовує модель для обробки даних, після чого формуються дані, необхідні для побудови конкретного виду. Кожен конкретний вид може використовувати інший вид у якості шаблону, що в нашому випадку реалізовано у вигляді основного шаблону сторінки, що забезпечує стандартизований вигляд сторінки без необхідності перевизначати основні елементи сторінки. При побудові сторінки необхідні елементи підставляються у шаблон, формуючи кінцеву HTML сторінку, що відправляється клієнту.

3.2 Структура та особливості реалізації методу спостереження за очима

Розроблюваний метод спостереження за очима був реалізований в середовищі Python з використанням бібліотек openCV, dlib та os. Основний функціонал методу був розроблений у формі бібліотеки з використанням деяких архітектурних принципів ООП.

Структура модуля спостереження за очима складається з основного скрипту, що викликає функції спостереження за очима *analyze_vision*, бібліотека

що здійснює аналіз погляду базуючись на результатах алгоритмів розпізнавання зображень *GazeTracking*, об’єкт що репрезентує повне зображення ока разом із бровою *Eye_full*, об’єкт що репрезентує видиму частину відкритого ока *Eye_visible*, об’єкт що представляє собою зіницю ока *Pupil* та об’єкт що відповідає за калібрацію межі відсікання кольорів для алгоритму пошуку зіниці ока *Calibration* (рис.3.6).

Рисунок 3.6 – Розгорнута діаграма класів розробленої Python бібліотеки

При роботі зі спостереженням за очима першим кроком є отримання зображення, що буде аналізуватися. Скрипт *analyze_vision* вміщує в собі функціонал завантаження зображення для аналізу та запуск аналізу зображення методом спостереження за очима у бібліотеці *GazeTracking*. Перед початком роботи скрипт перевіряє наявність зображення та завершує виконання програми у випадку помилки. Якщо зображення знайдено скрипт отримує основні параметри відео зображення, серед них розмір зображення у пікселях та кількість кадрів в секунду (FPS). Пересвідчившись в наявності зображення викликається клас *GazeTracking*, якому передається конкретне зображення або один кадр у відео. Отримавши результати аналізу зображення даний скрипт обраховує вихідні параметри та повертає їх у вигляді текстових рядків у консолі.

Клас *GazeTracking* вміщує у собі основний функціонал аналізу напрямку погляду та розпізнавання обличчя. При створенні екземпляру даного класу викликається функція *__init__()*, що створює змінні для зображення що обробляється *frame*, об’єкту калібрації *calibration*, моделі розпізнавання облич *_face_detector*, точкової моделі обличчя *_predictor* та об’єкти що відповідають лівому та правому оку, що розпізнається *eye_left* і *eye_right*. На цьому етапі також відбувається створення екземпляру класу калібрації *calibration* та завантаження моделей розпізнавання обличчя і точкової моделі *_face_detector* і *_predictor* відповідно. Після завершення даної функції екземпляр класу *GazeTracking* готовий для аналізу зображення. Для запуску аналізу зображення нам необхідно викликати функцію *refresh()*, що приймає зображення для аналізу як вхідний параметр та оновлює атрибут *frame* для подальшого аналізу даного зображення, після чого відбувається виклик функції *_analyze*. Функція *_analyze* виконує пошук обличчя на зображенні використовуючи попередньо визначену модель розпізнавання облич *_face_detector*, якому передається зображення для аналізу, що на виході повертає масив облич, що були знайдені на зображенні. Маючи обличчя на зображенні ми можемо використати точкову модель розпізнавання для визначення позиції очей на зображенні *_predictor*. Після розпізнавання зображення з використанням точкової моделі ми можемо перейти до аналізу конкретних частин обличчя, що у нашому випадку являються очима. Для аналізу очей на обличчі ми створюємо два екземпляра об'єктів *Eye_full*, що проводять розпізнавання зображень для кожного окремого ока. Після аналізу очей ми можемо окремо викликати функції, що повертають дані про результати аналізу, серед них: *pupil_left_coords()* та *pupil_right_coords()*, що повертають координати зіниці для лівого та правого ока відповідно; *pupil_left_located()*, *pupil_right_located()* та *both_pupils_located()*, що повертають булеву величину, яка описує чи було успішно знайдено зіницю лівого ока, зіницю правого ока та обидві зіниці для обидвох очей відповідно; *pupil_in_frame_horizontal_ratio()*, *pupil_in_frame_vertical_ratio()*, що обраховують середні горизонтальні та вертикальні координати зіниць обидвох очей на повному зображенні;

pupil_in_eye_horizontal_ratio(), pupil_in_eye_vertical_ratio(), що обраховують відносні середні горизонтальні та вертикальні координати для зіниці у кожному оці відносно кожного окремого ока. Маючи відносні координати зіниць ми можемо зробити висновок про положення очей на зображенні та напрямок погляду використовуючи методи *is_located_left(), is_located_right() is_located_top(), is_located_bottom(), is_located_center(),* що повертають булеву величину в залежності від положення зіниць на повному зображенні, а також методи *is_looking_left(), is_looking_right() is_looking_up(), is_looking_bottom(), is_looking_center()*, що повертають булеву величину в залежності від відносного положення зіниць на окремих зображеннях очей, що може бути використано для визначення напрямку погляду. Також для випадку закритих очей передбачено методи *is_blinking_right(), is_blinking_left()* та *is_blinking_both(),* що повертають булеву величину, яка репрезентує чи є закритим праве око, ліве око, або обидва ока відповідно. Для наглядної демонстрації роботи методу спостереження за очима передбачено функцію *annotated_frame(),* що повертає проаналізоване зображення з поміченими частинами зображення, що відповідають повному оку, видимій частині відкритого ока та позицією зіниці на зображенні. Функція *annotated_frame()* не є частиною системи спостереження за очима та в більшості використовується для тестування алгоритмів розпізнавання положення очей та напрямку погляду.

Клас *Eye_full* визначає повне зображення, що містить конкретне око. Для створення екземпляру класу *Eye_full* ми використовуємо такі атрибути, як зображення обличчя що аналізується *original_frame,* список точок що були розпізнані точковою моделлю обличчя *landmarks*, змінну що відповідає за сторону ока на обличчі (праве чи ліве око) *side* та посилання на загальний об’єкт калібрації *calibration*. При створенні екземпляру викликається конструктор *__init__(),* що створює змінні для ширини *width* та довжини *height* зображення, обрізаного зображення що міститиме лише око та брову над ним *frame,* координат першої та другої точок для обрізання зображення використовуючи побудову прямокутника за двома точками *origin* та *endpoint* відповідно, список

точок орієнтирів необхідних для виділення конкретного ока після обробки зображення точковою моделлю розпізнавання облич *landmark_points*, координати центру обрізаного зображення *center* та посилання на об’єкт *Eye_visible* що відповідає видимій частині відкритого ока *visible*. Після створення усіх змінних конструктор викликає функцію аналізу зображення _analyze, що на вході приймає початкове зображення обличчя *original_frame*, список розпізнаних точок *landmarks*, сторону ока *side* та посилання на об’єкт калібрації *calibration*. Ця функція вибирає набір точок, що відповідають вибраному оку *points* та викликає функцію обрахунку відношення сторін прямокутника що містить наше око *_eyebrow_ratio()*, що приймає на вхід такі параметри як список точок, розпізнаних точковою моделлю *landmarks* та список точок, що відповідають конкретному оку *points* та функцію виділення зображення конкретного ока *_isolate()*, що приймає на вхід початкове зображення original_frame, список усіх точок розпізнаних точковою моделлю *landmarks* та набір точок конкретного ока points. Після того як функція *_isolate* заповнює усі порожні змінні обрізаного зображення, його розміру та координат зображення, створюється екземпляр класу Eye_visible, що відповідає видимій частині відкритого ока. Функція *_eyebrow_ratio()* обраховує граничні координати прямокутника, що містить око, після чого відбувається обрахунок відношення ширини та висоти утвореного прямокутника шляхом операції ділення між відповідними величинами. Функція _isolate() обраховує розміри та позицію прямокутника, що вміщує в собі усі елементи ока та брови, шляхом пошуку мінімальних та максимальних координат у списку необхідних точок.

Клас Eye_visible призначений для визначення видимої частини ока та пошуку видимої зіниці. При створенні екземпляру класу, аналогічно до Eye_full, нам необхідно передати такі вхідні параметри, як *original_frame, landmarks, side, calibration,* що відповідають початковому зображенню обличчя для аналізу, списку точок розпізнаних точковою моделлю, стороні ока та посилання на екземпляр класу калібрації пошуку. Схожим чином маємо такі внутрішні атрибути, як *width, height, frame, origin, endpoint, center, landmark_points,* що

відповідають ширині та висоті обрізаного зображення, зображенню отриманому в результаті обрізання, початковим та кінцевим точкам для формування прямокутника, координатам центру зображення та набору точок, що використовуються для виділення зображення. Додатковими атрибутами є *blinking* та *pupil*, де *blinking* позначає чи є око закритим, а *pupil* є центральною точкою знайденої зіниці ока. Після визначення основних атрибутів викликається функція *_analyze()*, що, аналогічно до однойменної функції класу *Eye_full*, на вході приймає параметри *original_frame, landmarks, side ma calibration*. Першим кроком даної функції є вибір набору точок в залежності від сторони, на якій знаходиться наше око, на відміну від *Eye_full* тут вибираються лише точки, що описують контур відкритого ока *points*. Маючи вибрані точки контуру відкритого ока викликається функція _blinking_ratio(), що на вході отримує список розпізнаних точок *landmarks* та список точок, що описують контур ока *points*. Ця функція виділяє окремо граничні точки *left, right, top, bottom* ока та обраховує ширину *eye_width* і висоту *eye_height* видимої частини відкритого ока незалежно від положення голови, після чого обраховується відношення ширини до висоти шляхом ділення цих значень. Якщо око закрите, його видима ширина буде значно більшою ніж висота, збільшуючи значення відношення, навпроти коли око відкрите, відношення між шириною і висотою видимої частини ока прямує до нижчих значень, таким чином результат даної функції може бути використаний для визначення чи є око закритим на зображенні та ступеня закритості ока. Після цього викликається функція *_isolate()* на вході якої подаються такі параметри, як *original_frame, landmarks* та *points*. Ця функція вибирає координати усіх розпізнаних точок ока та обраховує мінімальні та максимальні координати, згідно яких будується прямокутник, що вміщує усі розпізнані точки відкритого ока. Використовуючи мінімальні та максимальні координати точок виділяється прямокутна частина зображення у вигляді частини матриці пікселів *frame*. Після цього записуються координати початкової та кінцевої точок побудови прямокутника *origin* та *endpoint*, обраховуються координати центру зображення *center* та розміри зображення *width, height*. Після

отримання зображення, що містить лише видиму відкриту частину ока відбувається перевірка калібрації *calibration.is_complete*(), що повертає булеву величину в залежності від того, чи було завершено калібрацію. Якщо калібрацію не було завершено, викликається функція калібрації *calibration_evaluate(),* що на вході, в якості параметрів, отримує обрізане зображення, що містить лише відкриту частину ока *frame* та сторону ока *side*. Цей метод покращує якість калібрації шляхом пошуку найкращого значення для подальшого бінарного відсікання зображення. По завершенню калібрації ми отримуємо значення межі відсікання *threshold* методом *calibration.threshold(),* що приймає сторону ока *side* як параметр. Останньою операцією функції *_analyze()* є створення екземпляру класу, що оброблятиме пошук зіниці у оці Pupil, де вхідними параметрами конструктору є обрізане зображення, що містить лише видиму частину відкритого ока *frame* та межу бінарного відсікання кольорів threshold.

Клас Calibration призначений для пошуку ідеального значення межі відсікання кольорів у процесі пошуку зіниці у оці. При створенні екземпляру даного класу проголошуються атрибути *nb_frames, thresholds_left* та *thresholds_right*, що відповідають кількості кадрів, межам відсікання для лівого ока та межам відсікання для правого ока. Функція *is_complete()* повертає булеву величину в залежності від кількості обрахованих значень межі відсікання. Якщо кількість значень у масивах *thresholds_right* та *thresholds_left* є рівним або вищим ніж значення атрибуту *nb_frames* метод повертає значення *True,* в іншому випадку повертається значення *False*. Функція *threshold()* на вході приймає сторону ока і повертає середнє значення усіх елементів у масиві *thresholds_right* або *thresholds_left* відповідно. Функція *evaluate()* відповідає за заповнення масивів межі відсікання, її вхідними параметрами є зображення ока, де необхідно знайти зіницю *eye_frame* та сторона, до якої дане зображення належить *side*. Ця функція викликає алгоритм пошуку найкращої межі для зображення *find_best_threshold()* та додає результат до масиву відповідного ока. Функція *find_best_threshold()* призначена для обрахунку найкращої межі відсікання для даного зображення, її єдиним вхідним параметром є зображення,

на якому необхідно знайти зіницю *eye_frame()*. Дана функція викликає статичний алгоритм виділення зіниці *image_processing()*, що на вході в якості параметрів приймає зображення що обробляється *eye_frame* та деяке обране значення межі відсікання *threshold* у класі *Pupil*, після чого виконує оцінку розміру зіниці в порівнянні із зображенням викликаючи функцію калібрації *iris_size()*, та порівнює отриманий результат з деяким значенням *average_iris_size*, що є константою. При цьому відбувається цикл вибору найкращої межі відсікання, де межа поступово змінюється, заповнюючи масив спроб, після чого вибирається та спроба, що є найближчою до значення *average_iris_size* як результат роботи методу, що повертається. Функція *iris_size()* приймає на вході зображення, що було оброблено методом бінарного відсікання кольорів та рахує кількість не порожніх пікселів, що залишилися у відсотковому відношенні.

Клас Pupil призначений для пошуку зіниці у оці та визначення координат зіниці. Основними атрибутами цього класу є *iris_frame()*, що представляє зображення зіниці після проведення операції відсікання кольорів, межу відсікання кольорів *threshold*, та координати зіниці ока *x* та *y*. Після створення цих основних змінних конструктор класу викликає функцію *detect_iris()*, що відповідає за визначення координат зіниці ока на зображенні. Першою операцією у *detect_iris()* є проведення відсікання кольорів для виділення контуру зіниці за допомогою статичної функції *image_processing()*, що в якості вхідних параметрів отримує зображення для аналізу *eye_frame* та значення межі відсікання *threshold*. Результат відсікання записується в атрибуті *iris_frame*. Маючи зображення після відсікання кольорів застосовується функція *findcountours()* бібліотеки openCV для пошуку контурів отриманої фігури зіниці, що на вході отримує оброблене зображення *iris_frame* після відсікання та параметри пошуку контурів. Маючи контур зіниці ока ми можемо знайти його центр, застосувавши функцію *moments()* бібліотеки openCV, що отримує контури зображення на вході та обраховує середньозважене значення інтенсивності кольорів пікселів зображення, після чого ми можемо обрахувати координати

зваженого центру контуру *x* та *y*. Визначені координати зіниці на зображенні можуть бути використані для визначення напрямку погляду.

Отже, було розглянуто основну структуру класів та їх функціонал для розроблюваного методу спостереження за очима

3.3 Структура та особливості реалізації вебсистеми тестування знань

Реалізована вебсистема тестування знань з використанням фреймворку Laravel, побудована з дотриманням шаблону MVC, що означає, що кожен запит користувача обробляється використовуючи контролер для опрацювання запиту, модель для створення стандартизованих сутностей та вид для побудови вебсторінки, що буде відправлена користувачу.

Структура контролерів вебсистеми має наступний вигляд (рис. 3.7):

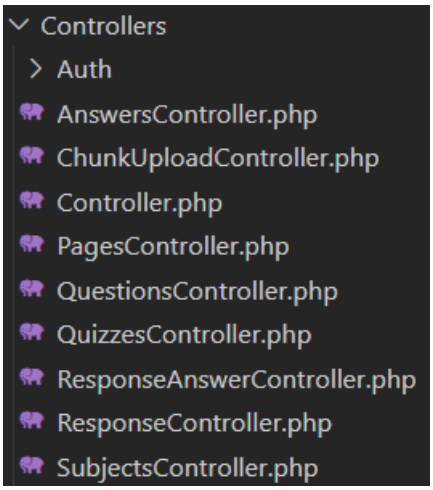


Рисунок 3.7 – Структура контролерів розроблюваної системи тестування знань

За функціоналом контролери можна поділити на сутнісні що відповідають за операції щодо записів у БД деякої сутності та функціональні що вміщують деякий функціонал обробки даних, але напряму не взаємодіють з сутностями бази даних.

Контролер Controller не містить в собі будь-якого функціоналу та слугує деяким шаблоном для побудови інших контролерів та надає доступ до загальних бібліотек, що використовуються в усіх інших контролерах що його наслідують.

Для реалізації функціоналу взаємодії з тестуваннями і предметами було створено контролери *SubjectsController*, *QuizzesController*, *QuestionsController* та *AnswersController*. Доступ до функцій даних контролерів здійснюється лише за умови наявності відповідних прав доступу у користувача.

Контролер *SubjectsController* відповідає за обробку запитів, що стосуються сутності предметів *subject*. Його основними функціями є *index(), getList(), create(), edit(), store(), update(), destroy()* та *show()*. Функція *index()* виконує пошук списку предметів за іменем та повертає список знайдених предметів. Функція *getList()* повертає список усіх предметів. Функція *create()* повертає вид для створення нового предмету. Функція *edit()* повертає вид для редагування існуючого предмету. Функція *store()* зберігає новий предмет у БД. Функція *update()* оновлює існуючий запис про предмет у БД. Функція *destroy()* видаляє предмет з БД. Функція *show()* повертає список усіх наявних тестувань у предметі.

Контролер *QuizzesController* відповідає за обробку запитів, що стосуються сутності тестувань *quiz*. Його основними функціями *index(), create(), edit(), store(), update(), destroy(), move()* та *show()*. Функція *index()* виконує пошук тестувань у предметі за назвою повертає вид списку знайдених тестувань. Функція *create()* повертає вид з формою для створення нового тестування. *edit()* повертає вид з формою для редагування існуючого тестування. Функція *store()* зберігає нове тестування у БД та повертає вид повного списку тестувань. Функція *update()* змінює існуючий запис про тестування у БД та повертає вид повного списку тестувань. Функція *destroy()* видаляє запис про тестування у БД та повертає вид списку тестувань. Функція *move()* змінює предмет до якого відноситься тестування у БД та повертає вид списку тестувань. Функція *show()* повертає вид завдань та відповідей у тестуванні, що призначений для редагування завдань та доступних варіантів відповідей.

Контролер *QuestionsController* відповідає за обробку запитів, що стосуються сутності завдань у тестуванні *question*. Його основними функціями є *index(), create(), edit(), store(), update()* та *destroy()*. Функція *index()* виконує

пошук усіх завдань у БД назва яких має збіг з деякою текстовою величиною та повертає вид списку завдань у тестуванні з елементами редагування. Функція *create()* повертає вид з формою для створення нового завдання у конкретному тестуванні. Функція *edit()* повертає вид з формою для редагування існуючого завдання у конкретному тестуванні. Функція *store()* зберігає нове завдання у тестуванні та повертає вид повного списку завдань у даному тестуванні. Функція *update()* змінює існуючий запис про завдання у тестуванні та повертає вид повного списку завдань у даному тестуванні. Функція *destroy()* видаляє запис про завдання у БД та повертає вид списку тестувань.

Контролер *AnswersController* відповідає за обробку запитів, що стосуються сутності відповідей на завдання у тестуванні. Його основними функціями є *create(), edit(), store(), update()* та *destroy()*. Функція *create()* повертає вид з формою створення нової відповіді для конкретного завдання. Функція *edit()* повертає вид з формою редагуванні існуючої відповіді на конкретне завдання. Функція *store()* зберігає нову відповідь для конкретного завдання. Функція *update()* оновлює існуючий запис про відповідь на завдання. Функція *destroy()* видаляє відповідь на конкретне завдання з БД.

Для обробки та збереження результатів тестування та використання методу спостереження за очима було створено контролер *ResponseController*. Інтегруючи метод спостереження за очима в вебсистему побудовану з використанням шаблону MVC ми можемо скласти наступну схему (рис. 3.8):

Контролер *ResponseController* призначений для збереження та оцінювання конкретної спроби проходження тестування. Його основними функціями є *index(), show(), store(), submit(), check_answers(), check_video()*. Функція *index()* виконує пошук конкретних результатів тестування за тестуванням та іменем користувача. Функція *show()* повертає вид для відображення оцінених результатів спроби проходження тестування. Функція *store()* зберігає дані про спробу проходження тестування. Функція *submit()* обробляє запит на завершення тестування та повертає вид з результатами оцінювання даної спроби.

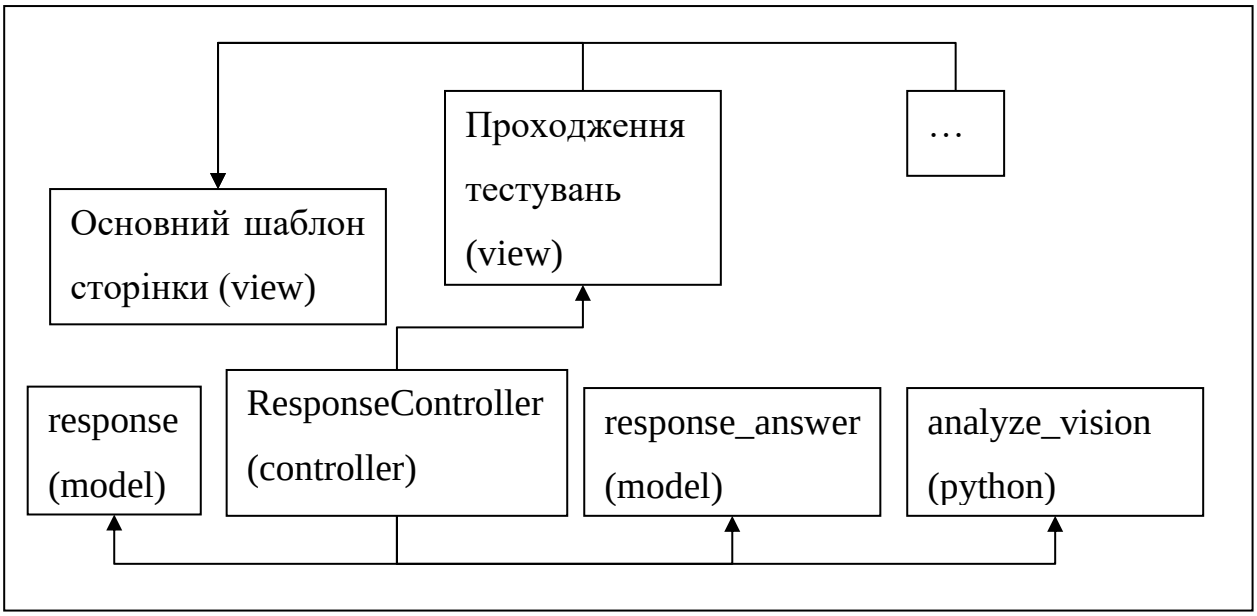


Рисунок 3.8 – Загальна схема взаємодії компонентів при використанні модуля аналізу погляду

При обробці завершення тестування відбувається перевірка відео наявності файлу, при наявності відео файлу обробка його завантаження виконується за допомогою функціоналу класу призначеного для обробки завантаження файлів по частинам *ChunkUploadController*, після чого виконується перевірка готовності завантаженого файлу. Після того як файл було повністю завантажено відбувається збереження обраних відповідей у тестуванні використовуючи функцію *store()*. По завершенню збереження викликається функція *check_answers()*, що що перевіряє кожну відповідь у тестуванні та зберігає обраховану оцінку. Останньою операцією обробки завершення тестування є виклик функції check_video() для застосування методу спостереження за очима для даної спроби. Функція *check_answers()* на початку виконує пошук тестування, що перевіряється, після чого вибираються завдання та варіанти відповідей на них. Маючи списки усіх завдань, можливих варіантів відповідей та список відповідей вибраних користувачем функція ітеративну перевіряє кожну відповідь у тестуванні і обраховує оцінку кожного завдання. Після оцінки кожного завдання обраховується кінцева оцінка у відсотковому вигляді. Функція *check_video()* відповідає за обробку відео зображення методом спостереження за очима, що оцінює довжину погляду поза екраном. При обробці

відео функція першим кроком виконує пошук сутності конкретної спроби тестування, що оцінюється та вибирає шлях до відео файлу для подальшого аналізу. Маючи шлях до файлу, що аналізується відбувається формування команди для виклику Python програми з шляхом до файлу в якості параметру командного рядка. Сама програма викликається з використанням консольної команди, де після завершення її роботи вихідні дані експортуються у вигляді текстових рядків. Отримані дані записуються в сутність спроби тестування та зберігаються в БД.

Контролери *Auth* призначені для забезпечення функціоналу реєстрації, авторизації, скидання паролю, контролю доступів та є автоматично створеними при підключенні Laravel бібліотеки *Auth*.

Отже, було описано загальні особливості реалізації вебсистеми тестування знань. При інтеграції зовнішнього модуля, що реалізує метод спостереження за очима зв'язок з вебсистемою відбувається в рамках контролеру *ResponseController*, де відбувається виклик його виконання за допомогою консольної команди.

3.4 Вимоги до публікації вебсистеми

Оскільки розроблюваний програмний продукт представляє собою вебзастосунок, доцільним є розділення вимог на клієнтські та серверні вимоги.

Для використання вебсистеми для тестування знань з функціоналом спостереження за очима зі сторони клієнта необхідно забезпечити:

– наявність підключеної камери з роздільною здатністю 640x480 пікселів або більше;

– наявність актуальної версії будь-якого браузера;

– об'єм ОЗП 2Гб або більше;

– об'єм вільного місця на диску 1 Гб або більше.

Для розгортання та функціонування серверної частини необхідно забезпечити:

– об'єм ОЗП 8 Гб або більше;
– об'єм вільного місця на диску 100 Гб або більше;
– операційна система Windows 11;
– інсталяція Python 3.9.16 або новіша версія;
– СКБД MySQL або інша СКБД що підтримується Laravel;
– MS Visual Studio Code або інший редактор коду з підтримкою Laravel.

3.5 Опис ІС тестування знань з вбудованим контролем за поглядом

3.5.1 Інструкція для розгортання серверної частини

Для початку роботи з серверною частиною необхідно виконати встановлення та налаштування вебзастосунку.

Для початкового встановлення застосунку необхідно виконати наведені далі кроки.

1. Завантажити архів та розпакувати у деякій папці на диску (шлях до папки повинен складатися лише з латинських символів).
2. Запустити СКБД MySQL або його аналог.
3. Перейти до папки застосунку та відкрити файл *.env* за допомогою VS Code або іншого редактору коду (рис. 3.9).

Рисунок 3.9 – Файл налаштувань *.env* у розробленій вебсистемі

4. Замінити параметри для доступу до СКБД та шлях до виконуваного файлу Python, що використовуватиметься для роботи модуля спостереження за очима (рис. 3.10).

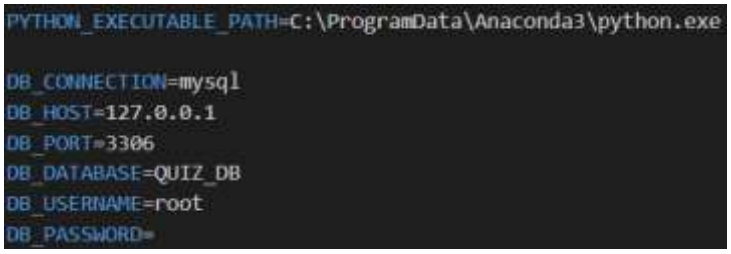


Рисунок 3.10 – Параметри налаштування СКБД та виконуваного файлу Python

5. У цьому ж файлі за необхідності можна підключити існуючий поштовий сервіс при умові існування доступного облікового запису (рис. 3.11).

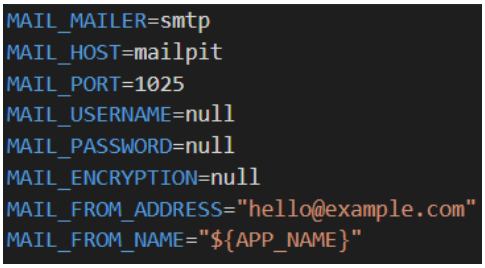


Рисунок 3.11 – Параметри налаштування поштового сервісу для авторизації

6. Після налаштування основних параметрів необхідно створити базу даних та її таблиці. Для цього необхідно перейти у консоль на виконати команду *php artisan migrate*, що створить таблиці за розробленим шаблоном та запропонує створити базу даних у з'єднаній СКБД, якщо її не було створено заздалегідь (рис. 3.12).

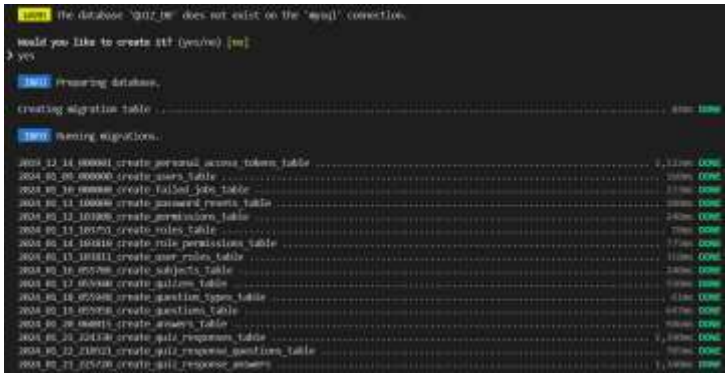


Рисунок 3.12 – Результат проведення міграції розробленого вебзастосунку

7. Створена БД не може функціонувати без початкового набору даних, що містить записи про типи тестувань та доступи користувачів. Для створення мінімальних необхідних записів для роботи вебсистеми необхідно використати консольну команду *php artisan db:seed*. По завершенню створення початкових записів вебсистема є готовою до використання (Рис. 3.13).

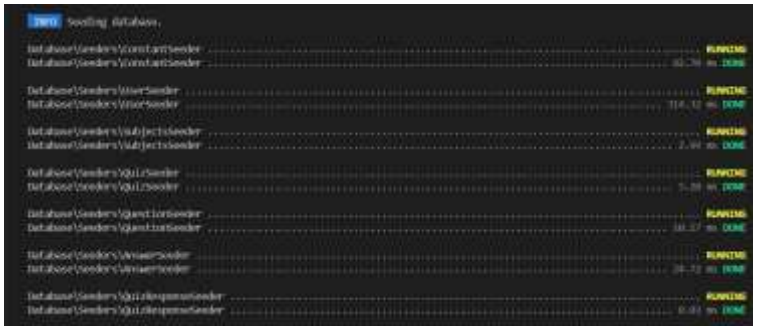


Рисунок 3.13 – Результат заповнення бази даних початковими даними

8. Для активації вебсистеми необхідно використати команду *php artisan serve*. При успішному виконанні команди результат має повернути URL адресу для доступу до серверу (рис. 3.14).

Рисунок 3.14 – Результат запуску серверної частини у консолі

9. Для початку роботи з розробленою вебсистемою можна перейти за посиланням, що були виведено у консоль при запуску серверу (рис. 3.15)

Рисунок 3.15 – Відкриття головної сторінки вебсистеми за посиланням

10. Для завершення роботи серверу необхідно перейти до консолі, що була використана для його запуску та натиснути комбінацію клавіш *Ctrl+C*.

11. При внесенні змін в основні елементи шаблону MVC або алгоритм спостереження за очима достатньо лише зберегти файл, без необхідності зупиняти та перезапускати сервер.

Отже, було наведено послідовність дій для встановлення, налаштування та розгортання вебсистеми для тестування знань.

3.5.2 Інструкція для проходження тестування

Для проходження тестування користувачем виконуються наведені далі кроки.

1. Для доступу до проходження тестувань необхідно здійснити вхід в існуючий обліковий запис користувача або створити новий. Для цього у правому верхньому куті передбачено кнопки «Увійти» та «Зареєструватися» (рис. 3.16)

Рисунок 3.16 – Елементи для авторизації користувача

2. Для створення облікового запису необхідно натиснути кнопку «Зареєструватися» після чого необхідно заповнити форму та натиснути кнопку «Register» (рис. 3.17). Для реєстрації необхідно лише вигадати ім'я користувача та пароль і вказати адресу електронної пошти. В залежності від налаштування системи авторизації може знадобитися підтвердження адреси електронної пошти.

Рисунок 3.17 – Форма для реєстрації користувача

3. Для входу в створений обліковий запис необхідно натиснути кнопку «Увійти» та ввести свою адресу електронної пошти та пароль (рис. 3.18). За бажання можна обрати опцію «Запам’ятати мене» для наступного входу з даного браузера без необхідності введення даних вручну.

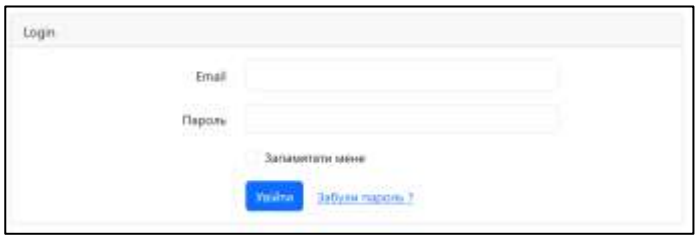


Рисунок 3.18 – Форма для входу користувача

4. При успішному вході у правому верхньому куті елементи авторизації будуть заміщені випадаючим меню користувача. (рис. 3.19).

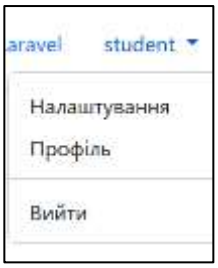


Рисунок 3.19 – Форма для входу користувача

5. Для пошуку тестувань спочатку необхідно Для пошуку необхідного предмету можна застосувати функцію пошуку (рис. 3.20).

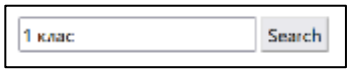


Рисунок 3.20 – Функція пошуку у верхній панелі керування сторінки

6. Обрати предмет до якого відноситься необхідне тестування та натиснути кнопку «Переглянути» (рис. 3.21).

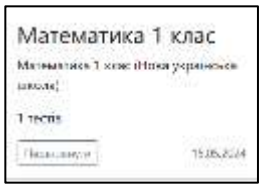


Рисунок 3.21 – Перехід до тестувань у предметі

7. За необхідності використати функцію пошуку для вибору необхідного тестування.

8. Обрати тестування та натиснути на кнопку «Розпочати тест» (рис. 3.22).

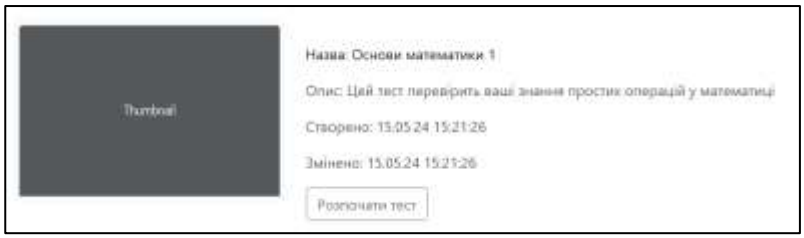


Рисунок 3.22 – Одне тестування у списку тестувань

9. Підтвердити проходження тестування у спливаючому вікні (рис. 3.23)

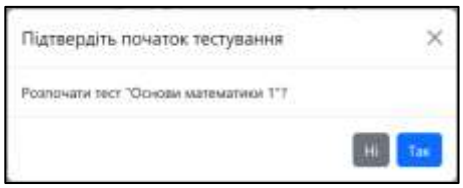


Рисунок 3.23 – Перехід до початку тестування у предметі

10. Для забезпечення роботи алгоритму спостереження за очима необхідно надати доступ до камери (рис. 3.24). В іншому випадку елементи керування у тестуванні автоматично блокуються.

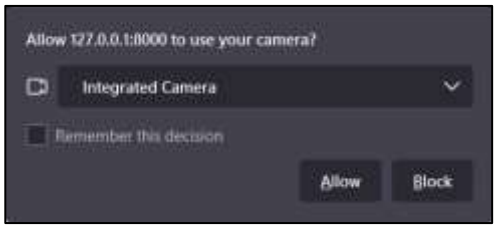


Рисунок 3.24 – Перехід до початку тестування у предметі

11. Пройти тестування знань обираючи відповіді для кожного завдання. По завершенню тестування натиснути на кнопку «Завершити тест» (рис. 3.25).

Рисунок 3.25 – Проходження тестування

12. По завершенню тестування та аналізу зображення на сторінці буде відображено результат тестування (рис. 3.26).

Рисунок 3.26 – Проходження тестування

Отже, було розглянуто алгоритм дій для проходження тестування з використанням методу спостереження за очима.

3.5.3 Інструкція для редагування сутностей

Далі наведені основні кроки для редагування сутностей (предмети та тестування).

1. Для доступу до редагування тестувань та предметів необхідно увійти в обліковий запис з правами адміністратора. Логін та пароль для входу створюється при заповненні БД початковими даними та може бути знайдений у файлі UserSeeder.php (рис. 3.27).

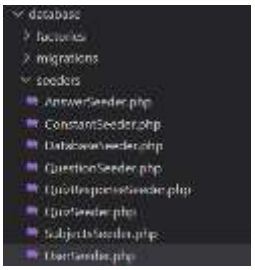


Рисунок 3.27 – Положення файлу UserSeeder.php в основній папці застосунку

За необхідності в даний файл можна внести зміни, але для їх проведення необхідно видалити БД та повторити процес її створення (рис. 3.28).

```php
//Профіль адміну
User::create([
    'name' => 'admin',
    'email' => 'admin@example.com',
    'password' => Hash::make('admin'),
]);
user_role::create([
    'user_id' => 1,
    'role_id' => 1
]);
```

Рисунок 3.28 – Запис про профіль адміну у файлі UserSeeder.php

2. Після авторизації як адміністратор на сторінці відображаються додаткові елементи керування (рис. 3.29).

а) б) в)

Рисунок 3.29 – а) Кнопка «редагувати» у списку предметів, б) Функціонал додавання нових сутностей у випадаючому меню, в) Додаткові кнопки взаємодії з сутністю тестування

3. При редагуванні тестувань необхідно внести зміни в текстові поля та за бажанням додати нову обкладинку тестування, після чого для збереження необхідно натиснути на кнопку «Підтвердити зміни» (рис. 3.30).

Рисунок 3.30 – Форма редагування тестування

4. Для редагування завдань у питанні у формі редагування необхідно натиснути кнопку «Редагувати питання», після чого на екрані відобразиться інтерфейс редагування завдань у конкретному тестуванні (рис. 3.31).

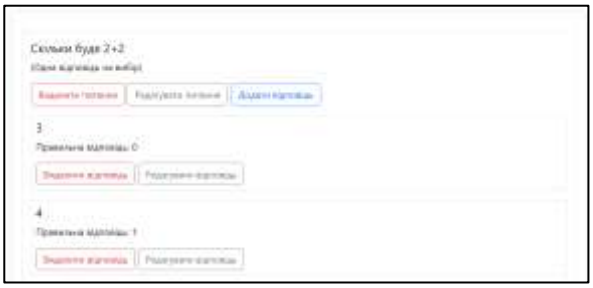


Рисунок 3.31 – Загальний інтерфейс редагування завдань у тестуванні

Отже, було розглянуто інструкцію користування для функціоналу редагування та створення нових предметів та тестувань.

3.6 Висновки до розділу 3

Отже, в даному розділі було висвітлено особливості програмної реалізації розроблюваної вебсистеми для тестування знань з вбудованим методом спостереження за очима. Було розглянуто основну архітектуру вебсистеми тестування знань, структуру та особливості реалізації вебсистеми та її інтеграції з методом спостереження за очима, вимоги до розгортання вебсистеми, інструкцію користувача для розгортання, налаштування та використання вебсистеми для тестування знань.

Загальні висновки

Під час виконання КРБ було розглянуто існуючі рішення для спостереження за очима, розроблено метод спостереження за очима для систем тестування знань та проведено інтеграцію модуля спостереження за очима у розроблену вебсистему тестування знань. Розроблений метод може бути застосований для підвищення академічної доброчесності при проходженні онлайн тестувань.

Розроблений програмний продукт має наступний функціонал

– реєстрація та вхід користувачів:
– взаємодія з сутностями БД для авторизованих користувачів;
– відображення списків предметів та тестувань
– можливість проходження тестувань для авторизованих користувачів.
– аналіз спроби проходження тестування методом спостереження за очима;
– збереження та відображення даних про результати проходження тестування.

Отже, було досягнуто усіх завдань поставленої мети. Розроблена вебсистема та метод спостереження за очима можуть бути використані для проведення тестування знань в онлайн форматі та забезпечення підвищеного рівня академічної доброчесності під час проходження тестування знань.

Розроблена система може бути покращена шляхом оптимізації алгоритму спостереження за очима для зменшення кількості необхідних обчислювальних ресурсів, створенням та налаштуванням поштового сервісу для інтеграції з системою авторизації та змінами у дизайн інтерфейсу сторінок.

Перелік посилань

1. Eye Tracking | Usability.gov URL: https://www.usability.gov/how-to-and-tools/methods/eye-tracking.html

2. Top 7 3D Eye Tracking Use Cases & Applications URL: https://eyeware.tech/blog/top-7-use-cases-for-3d-eye-tracking/

3. Practical Applications of Eye Tracking Technology | by Ankit Narayan Singh | Medium URL: https://ankitnsingh.medium.com/practical-applications-of-eye-tracking-technology-d30bfe0c131e

4. Eye Gaze Tracking: Applications, Techniques, and Key Metrics - Datagen URL: https://datagen.tech

5. The Different Kinds of Eye Tracking Devices | Bitbrain URL: https://www.bitbrain.com/blog/eye-tracking-devices

6. Learn here what are the different eye-tracking techniques and methods to record eye movements accurately, how to calibrate, and what are the limitations. | Bitbrain URL: https://www.bitbrain.com/blog/eye-tracking-technology

7. Eye Gaze Detection Based on Computational Visual Perception and Facial Landmarks URL: https://www.techscience.com/cmc/v68n2/42153/html

8. Face Detection with Python using OpenCV Tutorial | DataCamp URL: https://www.datacamp.com/tutorial/face-detection-python-opencv

9. Eye motion tracking - Opencv with Python - Pysource URL: https://pysource.com/2019/01/04/eye-motion-tracking-opencv-with-python/

10. The 68 landmarks detected by dlib library. This image was created by... | Download Scientific Diagram URL: https://www.researchgate.net/figure/The-68-landmarks-detected-by-dlib-library-This-image-was-created-by-Brandon-Amos-of-CMU_fig2_329392737

11. Infrared Eye Tracking – EYEVIDO URL: https://eyevido.de/en/infrared-eye-tracking/

12. A free geometry model-independent neural eye-gaze tracking system - PMC URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3543256/

13. The Top 15 Eye Tracking Platforms for Market & User Research | Insight Platforms URL: https://www.insightplatforms.com/the-top-15-eye-tracking-platforms-for-market-user-research/

14. Eye-tracking – Lumen Research URL: https://lumen-research.com/eye-tracking/

15. The global leader in eye tracking since 2001 - Tobii URL: https://www.tobii.com/company/this-is-tobii

16. Webcam Eye Tracking as an effective solution | eyezag URL: https://eyezag.com/eye-tracking/

17. Element Human - The All-In-One Human Experience Platform URL: https://www.elementhuman.com

18. UX research, packaging and planogram testing, ads effectiveness URL: https://www.realeye.io/use-cases

19. Презентували технологію EyePass, яка верифікує дистанційне навчання | UACRISIS.ORG URL: https://uacrisis.org/uk/75327-eye-pass-presentation

20. EyePass. Глибока аналітика онлайн-навчання починається тут! URL: https://eyepass.tech/ukr.html#rec165087895

21. Welcome to the Moodle Developer Resource site | Moodle Developer Resources URL: https://moodledev.io/

22. Тести - Онлайн-школа «На Урок» URL: https://naurok.ua/student/tests

23. Тести ЗНО/НМТ онлайн - тренувальні тести – сайт ЗНО.Освіта.UA URL: https://zno.osvita.ua/

24. Бібліотека тестів URL: https://vseosvita.ua/test

25. About Python™ | Python.org URL: https://www.python.org/about/

26. Installation - Laravel 11.x - The PHP Framework For Web Artisans URL: https://laravel.com/docs/11.x

27. MySQL URL: https://www.mysql.com

ДОДАТКИ

Додаток А

Програмні коди

Лістинг SubjectsController.php:

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Str;

use App\Models\subject;

class SubjectsController extends Controller
{
    //Знайти предмет за назвою
    public function index(Request $request)
    {
        $search = $request->input('search');

        $subjects = Subject::when($search, function ($query) use ($search) {
            $query->where('name', 'like', '%' . $search . '%');
        })->paginate(10);

        return view('home', compact('subjects'))->with('pageTitle', 'Навчальні
предмети');
    }
    //Список предметів
    public function getList()
    {
        $subjects = Subject::all();

        return view('home')->with('subjects', $subjects)->with('pageTitle', 'Навчальні
предмети');
    }
    //Додати предмет
    public function create()
    {
        return view('subjects/create'); // форма створення предмету
    }
    //Редагувати предмет
    public function edit($id)
    {
        $subject = Subject::find($id);
        return view('subjects/edit')->with('subject', $subject);
    }
    //Зберегти зміни в БД
    public function store(Request $request)
    {
        $subject = new Subject();

        $file = $request->file('cover_image');
        $random_name = Str::random(8);
        $destinationPath = 'assets_subjects/';
        $extension = $file->getClientOriginalExtension();
        $filename = $random_name . '_cover.' . $extension;
        $uploadSuccess = $request->file('cover_image')->move($destinationPath,
$filename);
```

```
        $subject->name = $request->get('name');
        $subject->description = $request->get('description');
        $subject->cover_image = $filename;
        $subject->save();
        return redirect('/');
    }
    //Оновити предмет
    public function update(Request $request, $id)
    {
        $subject = Subject::find($id);
        if (isset($request->file)) {
            $file = $request->file('cover_image');
            $random_name = Str::random(8);
            $destinationPath = 'subjects/';
            $extension = $file->getClientOriginalExtension();
            $filename = $random_name . '_cover.' . $extension;
            $uploadSuccess = $request->file('cover_image')->move($destinationPath,
$filename);

            $subject->cover_image = $filename;
        }

        $subject->name =  $request->name;
        $subject->description = $request->description;
        $subject->save();
        return redirect('/');
    }
    //Видалити предмет
    public function destroy($id)
    {
        $subject = Subject::find($id);
        $subject->delete();
        return redirect('/');
    }
    //Показати вміст предмету

    public function show($id)
    {
        $subject = Subject::find($id);
        $subjects = Subject::all();

        $quizzes = $subject->quizzes;

        return view('subjects/show')->with('subject', $subject)-
>with('quizzes',$quizzes) //поточний предмет
            ->with('subjects', $subjects); //всі предмети
    }
}
```

Лістинг QuizzesController.php:

```
<?php

namespace App\Http\Controllers;

use Illuminate\Support\Facades\Auth;
use Illuminate\Http\Request;
use Illuminate\Support\Str;

use App\Models\quiz;
use App\Models\subject;

class QuizzesController extends Controller
```

```
{
    //Пошук тесту
    public function index(Request $request, $subjectId)
    {
        $subject = Subject::find($subjectId);

        $search = $request->input('search');

        $quizzes = $subject->quizzes()->when($search, function ($query) use ($search) {
            $query->where('name', 'like', '%' . $search . '%');
        })->paginate(10);

        $subjects = Subject::all();

        return view('subjects/show')->with('subject', $subject)->with('quizzes',
$quizzes) //поточний предмет
            ->with('subjects', $subjects); //всі предмети
    }
    //Додати тест
    public function create($subject_id)
    {
        return view('quizzes/create')->with('subject_id', $subject_id); // форма
створення тесту
    }
    //Редагувати тест
    public function edit($id)
    {
        $quiz = quiz::find($id);
        return view('quizzes/edit')->with('quiz', $quiz);
    }
    //Зберегти зміни в БД
    public function store(Request $request, $subject_id)
    {
        if (Auth::check()) {
            if (isset($request->file)) {
                $file = $request->file('quiz');
                $random_name = Str::random(8);
                $destinationPath = 'assets_quizzes/';
                $extension = $file->getClientOriginalExtension();
                $filename = $random_name . '.' . $extension;
                $uploadSuccess = $request->file('quiz')->move($destinationPath,
$filename);
            } else {
                $filename = 'default.png';
            }

            $quiz = new quiz();
            $quiz->user_id = Auth::id();
            $quiz->subject_id = $subject_id;
            $quiz->name = $request->get('name');
            $quiz->description = $request->get('description');
            $quiz->cover_image = $filename;
            $quiz->save();
        }
        return redirect()->route('subjects.show', $quiz->subject->id);
    }
    //Оновити тест
    public function update(Request $request, $id)
    {
        $quiz = quiz::find($id);
        if (isset($request->file)) {
            $file = $request->file('cover_image');
            $random_name = Str::random(8);
```

```
            $destinationPath = 'assets_quizzes/';
            $extension = $file->getClientOriginalExtension();
            $filename = $random_name . '_cover.' . $extension;
            $uploadSuccess = $request->file('cover_image')->move($destinationPath,
$filename);

            $quiz->cover_image = $filename;
        }

        $quiz->name =  $request->name;
        $quiz->description = $request->description;
        $quiz->save();
        return redirect()->route('subjects.show', $quiz->subject->id);
    }
    //Видалити тест
    public function destroy($id)
    {
        $quiz = quiz::find($id);
        $quiz->delete();
        return redirect()->route('subjects.show', $quiz->subject->id);
    }
    //Показати вміст тесту (редактор питань)
    public function show($id)
    {
        $quiz = quiz::find($id);

        $questions = $quiz->questions;

        return view('quizzes/show')->with('quiz', $quiz)->with('questions', $questions);
    }
    //Перемістити тест
    public function move(Request $request, $id)
    {
        $quiz = quiz::find($id);
        $quiz->subject_id = $request->get('new_subject');
        $quiz->save();
        return redirect()->route('subjects.show', $quiz->subject->id);
    }
}
```

Лістинг QuestionsController.php:

```
<?php

namespace App\Http\Controllers;
use Illuminate\Http\Request;
use Illuminate\Support\Str;

use App\Models\quiz;
use App\Models\subject;
use App\Models\question;
use App\Models\question_type;

class QuestionsController extends Controller
{
    //Пошук питання
    public function index(Request $request, $quiz_id)
    {
        $quiz = quiz::find($quiz_id);

        $search = $request->input('search');

        $questions = $quiz->questions()->when($search, function ($query) use ($search) {
```

```
            $query->where('question_text', 'like', '%' . $search . '%');
        })->paginate(10);

        return view('quizzes/show')->with('quiz', $quiz)->with('questions', $questions);
    }

    //Форма створення
    public function create($quiz_id)
    {
        $quiz = quiz::find($quiz_id);
        $question_types = question_type::all();

        $subject = $quiz->subject;
        return view('questions/create')->with('quiz', $quiz)->with('subject', $subject)-
>with('question_types', $question_types); // форма створення питання
    }

    //Форма редагування
    public function edit($id)
    {
        $question = question::find($id);

        $question_types = question_type::all();

        return view('questions/edit')->with('question', $question)-
>with('question_types', $question_types);
    }

    //Зберегти нове питання
    public function store(Request $request, $quiz_id)
    {
        $question = new question();

        //Зберегти зображення для питання
        if (isset($request->question_image)) {
            $file = $request->file('question_image');
            $random_name = Str::random(8);
            $destinationPath = 'assets_questions/';
            $extension = $file->getClientOriginalExtension();
            $filename = $random_name . '.' . $extension;
            $uploadSuccess = $request->file('question_image')->move($destinationPath,
$filename);
            $question->question_image = $filename;
        }

        //Зберегти питання в базу даних
        $question->quiz_id = $quiz_id;
        $question->question_text = $request->get('question_text');
        $question->question_type_id = $request->get('question_type_id');
        $question->save();

        return redirect()->route('quizzes.show', $quiz_id);
    }

    //Редагувати існуюче питання
    public function update(Request $request, $question_id)
    {
        $question = question::find($question_id);

        if (!isset($request->question_image)) {
            $file = $request->file('question_image');
            $random_name = Str::random(8);
            $destinationPath = 'assets_questions/';
```

```
            if (is_null($file)) {
                $question->question_image = null;
            } else {
                $extension = $file->getClientOriginalExtension();
                $filename = $random_name . '_cover.' . $extension;
                $uploadSuccess = $request->file('question_image')-
>move($destinationPath, $filename);

                $question->question_image = $filename;
            }
        }
        $question->question_text = $request->get('question_text');
        $question->question_type_id = $request->get('question_type_id');
        $question->save();

        $quiz = $question->quiz;
        return redirect()->route('quizzes.show', $quiz->id);
    }

    //Видалити питання
    public function destroy($id)
    {
        $question = Question::find($id);
        $quiz = $question->quiz;

        $question->delete();
        return redirect()->route('quizzes.show', $quiz->id);
    }
}
```

Лістинг AnswersController.php:

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Str;

use App\Models\question;
use App\Models\answer;

use Illuminate\Support\Facades\Log;

class AnswersController extends Controller
{
    //Додати відповідь
    public function create($question_id)
    {
        $question = question::find($question_id);
        $quiz = $question->quiz;
        $subject = $quiz->subject;

        return view('answers/create')->with('subject', $subject)->with('quiz', $quiz)-
>with('question', $question); // форма створення питання
    }
    //Редагувати відповідь
    public function edit($id)
    {
        $answer = answer::find($id);

        return view('answers/edit')->with('answer', $answer);
```

```
    }
    //Зберегти відповідь в БД
    public function store(Request $request, $question_id)
    {
        $answer = new answer();
        $question = question::find($question_id);

        if (isset($request->answer_image)) {
            $file = $request->file('answer_image');
            $random_name = Str::random(8);
            $destinationPath = 'assets_answers/';
            $extension = $file->getClientOriginalExtension();
            $filename = $random_name . '.' . $extension;
            $uploadSuccess = $request->file('question_image')->move($destinationPath,
$filename);
            $answer->answer_image = $filename;
        }

        $answer->question_id = $question_id;
        $answer->answer_text = $request->get('answer_text');

        if($request->get('is_correct'))
        {
            $answer->is_correct = $request->get('is_correct');
        }
        else
        {
            $answer->is_correct = 0;
        }

        $answer->save();

        return redirect()->route('quizzes.show', $question->quiz);
    }
    //Оновити відповідь
    public function update(Request $request, $answer_id)
    {
        $answer = Answer::find($answer_id);

        if (isset($request->answer_image)) {
            $file = $request->file('answer_image');
            $random_name = Str::random(8);
            $destinationPath = 'assets_answers/';

            if(is_null($file))
            {
                $answer->answer_image = null;
            }
            else
            {
                $extension = $file->getClientOriginalExtension();
                $filename = $random_name . '_cover.' . $extension;
                $uploadSuccess = $request->file('answer_image')->move($destinationPath,
$filename);

                $answer->answer_image = $filename;
            }
        }
        $answer->answer_text = $request->get('answer_text');

        if($request->get('is_correct'))
        {
            $answer->is_correct = $request->get('is_correct');
```

```
        }
        else
        {
            $answer->is_correct = 0;
        }

        $answer->save();
        $quiz = $answer->question->quiz;
        return redirect()->route('quizzes.show', $quiz->id);
    }
    //Видалити відповідь
    public function destroy($id)
    {
        $answer = answer::find($id);
        $quiz = $answer->question->quiz;

        $answer->delete();
        return redirect()->route('quizzes.show', $quiz->id);
    }
}
```

Лістинг ChunkUploadController.php:

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Http\UploadedFile;
use Illuminate\Support\Facades\Log;

use Pion\Laravel\ChunkUpload\Handler\HandlerFactory;
use Pion\Laravel\ChunkUpload\Receiver\FileReceiver;
use Pion\Laravel\ChunkUpload\Exceptions\UploadMissingFileException;
use Pion\Laravel\ChunkUpload\Exceptions\UploadFailedException;

class ChunkUploadController extends Controller
{
    /*public function index()
    {
        return view('upload');
    }*/

    /*
     * Handles the file upload
     *
     * @param Request $request
     *
     * @return JsonResponse
     *
     * @throws UploadMissingFileException
     * @throws UploadFailedException
     */
    public function store(Request $request)
    {
        //Почати прийняття файлу
        $receiver = new FileReceiver("file", $request,
HandlerFactory::classFromRequest($request));
        //Перевірити наявність файлу
        if ($receiver->isUploaded() === false) {
            throw new UploadMissingFileException();
        }
        //Отримати частину файлу
```

```
        $save = $receiver->receive();
        Log::debug('Chunk received.');

        //Зберегти файл якщо усі частини було завантажено
        if ($save->isFinished()) {
            Log::debug('Download complete.');
            return $this->saveFile($save->getFile());
        }
            // save the file and return any response you need, current example uses
`move` function. If you are
            // not using move, you need to manually delete the file by unlink($save-
>getFile()->getPathname())

        //Повернути прогрес завантаження, якщо файл завантажується по частинах
        $handler = $save->handler();
        Log::debug("Download progress: " . $handler->getPercentageDone() . "%");
        return response()->json([
            "done" => $handler->getPercentageDone(),
            'status' => true
        ]);
    }
    /*
     * Saves the file
     *
     * @param UploadedFile $file
     *
     * @return JsonResponse
     */
    protected function saveFile(UploadedFile $file)
    {
        $fileName = $this->createFilename($file);

        //Згрупувати файли за типом файлу
        $mime = str_replace('/', '-', $file->getMimeType());

        //Згрупувати файли за датою
        $dateFolder = date("Y-m-W");

        //Шлях до папки де буде збережено файл
        $filePath = "upload/{$mime}/{$dateFolder}";
        $finalPath = storage_path("app/public/" . $filePath);

        //Перемістити файл
        $file->move($finalPath, $fileName);

        $path = asset('storage/' . $filePath. '/'.$fileName);

        $local_path = $finalPath.'/'.$fileName;

        Log::debug('File saved.');

        return response()->json([
            'machine_path' => $local_path,
            'network_path' => $path,
            'mime_type' => $mime
        ]);
    }

    /*
     * Create unique filename for uploaded file
     * @param UploadedFile $file
     * @return string
     */
```

```
    protected function createFilename(UploadedFile $file)
    {
        $extension = $file->guessExtension() ?: 'unknown';
        $filename = str_replace("." . $extension, "", $file->getClientOriginalName());

        //Додати унікальний хеш до назви файлу
        $filename .= "_" . md5(time()) . "." . $extension;

        return $filename;
    }
}
```

Лістинг PagesController.php:

```
<?php

namespace App\Http\Controllers;

class PagesController extends Controller
{
    public function home()
    {
        return view('home');
    }
    public function about()
    {
        return view('about');
    }
    public function laravel()
    {
        return view('laravel');
    }
}
```

Лістинг ResponseController.php:

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\App;
use Illuminate\Support\Facades\Log;
use Illuminate\Support\Facades\Auth;

use Symfony\Component\Process\Process;
use Symfony\Component\Process\Exception\ProcessFailedException;

use App\Models\User;
use App\Models\answer;
use App\Models\question;
use App\Models\quiz;
use App\Models\quiz_response;
use App\Models\response_question;
use App\Models\response_answer;

class ResponseController extends Controller
{
    //Знайти відповіді за тестом та користувачем
    public function index(Request $request)
    {
        $userName = $request->input('userName');
        $quizID = $request->input('quizID');
```

```
        //Знайти користувачів за іменем
        $userIDs = User::when($userName, function ($query) use ($userName) {
            $query->where('name', 'like', '%' . $userName . '%');
        })->pluck('id');
        //Вивести усі відповіді для обраних користувачів
        $responses = Quiz_Response::when($quizID, function ($query) use ($quizID) {
            $query->where('quiz_id', $quizID);
        })->when($userIDs->isNotEmpty(), function ($query) use ($userIDs) {
            $query->whereIn('user_id', $userIDs);
        })->paginate(10);
    }

    //Розпочати тест
    public function start($quiz_id)
    {
        //Якщо користувач не зареєстрований перенаправити на сторінку логіну
        if (!Auth::check()) {
            return redirect()->route('login');
        }

        $quiz = quiz::find($quiz_id);

        $questions = $quiz->questions;
        return view('quiz_responses/run')->with('quiz', $quiz)->with('questions',
$questions);
    }

    //Обробка запиту завершення тестування
    public function submit(Request $request)
    {
        //Якщо користувач не зареєстрований перенаправити на сторінку логіну
        if (!Auth::check()) {
            return redirect()->route('login');
        }

        //Обробка завантаження відео по частинах
        $file_status = App::call('App\Http\Controllers\ChunkUploadController@store',
[$request]);
        $file_status = json_decode($file_status->getContent(), true);

        //Повернути результати перевірки тестів
        if (isset($file_status['machine_path'])) {
            Log::debug('Video uploaded successfully. Path: ' .
($file_status['machine_path']) . "\n");
        } else {
            //Чекати поки файл не завантажиться
            return null;
        }
        $video = $file_status['machine_path'];

        //Збереження даних з форми тесту
        $quiz_response = $this->store($request);
        //Виставлення оцінки
        $quiz_response = $this->check_answers($quiz_response->id);
        //Збереження відео для аналізу
        $quiz_response->response_video = $video;
        $quiz_response->save();
        //Аналіз відео
        $quiz_response = $this->check_video($quiz_response->id);

        //Повернути результат користувачу
        $redirectUrl = route('responses.show', ['response' => $quiz_response->id]);
```

```
        return response()->json($redirectUrl);
    }
    //Збереження відповідей на тестування
    public function store(Request $request)
    {
        //Якщо користувач не зареєстрований перенаправити на сторінку логіну
        if (!Auth::check()) {
            return redirect()->route('login');
        }

        Log::debug('Storing quiz answers...');
        //Отримати користувача, що пройшов тест
        if (Auth::check()) {
            //Обєкт відповіді на тест
            $response = new quiz_response();

            $quiz = quiz::find($request->query('quiz_id'));
            $response->quiz_id = $quiz->id;

            $user = Auth::user();
            $response->user_id = $user->id;

            //Тимчасово зберегти незавершений обєкт
            $response->save();

            //Відповіді обрані користувачем
            $selectedAnswers = $request->query('selected_answers');

            //Перебирати лише правильно зформульовані питання (пропустит питання без
правильної відповіді)
            $validQuestions = $quiz->questions->filter(function ($question) {
                return $question->answers->where('is_correct', 1)->count() > 0;
            });

            foreach ($validQuestions as $question) {
                //Записати питання в БД
                $response_question = new response_question();
                $response_question->question_id = $question->id;
                $response_question->response_id = $response->id;
                $response_question->save();

                //Записати в БД відповідь на кожне окреме питання
                if (isset($selectedAnswers[$question->id])) {
                    //Вибрати відповідь на питання
                    $selectedOption = $selectedAnswers[$question->id];
                    //Тип завдання
                    $questionTypeId = $question->question_type->id;

                    //Зберегти вибрану відповідь
                    if ($questionTypeId == 1) {
                        //Обєкт відповіді
                        $response_answer = new response_answer();
                        $response_answer->response_question_id = $response_question->id;

                        $selected = answer::find($selectedOption);
                        $response_answer->answer_id = $selected->id;
                        $response_answer->answer_text = $selected->answer_text;

                        $response_answer->save();
                    }
                    //Зберегти декілька відповідей
                    if ($questionTypeId == 2) {
```

```
                        $selectedOptions = answer::find($selectedOption)->pluck('id')-
>toArray();

                        foreach ($selectedOptions as $selectedOption) {
                            //Обєкт відповіді
                            $response_answer = new response_answer();
                            $response_answer->response_question_id = $response_question-
>id;

                            $selected = answer::find($selectedOption);
                            $response_answer->answer_id = $selected->id;
                            $response_answer->answer_text = $selected->answer_text;

                            $response_answer->save();
                        }
                    }
                    //Зберегти текстову відповідь
                    if ($questionTypeId == 3) {
                        $selectedText = $selectedOption;

                        $response_answer = new response_answer();
                        $response_answer->response_question_id = $response_question->id;

                        $selectedAnswer = $question->answers->first(function ($answer)
use ($selectedText) {
                            return strtolower($answer->answer_text) ===
strtolower($selectedText);
                        });

                        if ($selectedAnswer) {
                            $response_answer->answer_id = $selectedAnswer->id;
                        }

                        $response_answer->answer_text = $selectedText;

                        $response_answer->save();
                    }
                }
            }
            Log::debug("Response answers recorded successfully!\n");
            return $response;
        } else {
            Log::debug('Unable to store data: User is not logged in !');
        }
    }

    //Перевірка тесту та виставлення оцінок
    public function check_answers($quiz_response_id)
    {
        $quiz_response = quiz_response::find($quiz_response_id);

        $response_questions = $quiz_response->response_questions;

        Log::debug('Calculating response questions score.');

        //Перебираємо кожне правильно сформульоване питання та перевіряємо відповіді
        foreach ($response_questions as $response_question) {

            $question = $response_question->question;

            //Вибрати відповіді на конкретне питання
            $response_question_answers = $response_question->response_answers;
            //Вибрати відповіді на конкретне питання
```

```
            if (!$response_question_answers->isEmpty()) { //Якщо відповідь не вибрано
пропустити питання
                //Тип завдання
                $questionTypeId = $question->question_type->id;

                if ($questionTypeId == 1) { //Обробка одної відповіді на вибір
                    Log::debug('Checking one option question...');

                    //Вибрана відповідь
                    $SelectedAnswer = $response_question_answers->first()->answer;

                    //Перевірити чи відповідь правильна
                    if ($SelectedAnswer->is_correct === 1) {
                        $response_question->score_percentage = 100;
                        $response_question->save();

                        Log::debug('Answer is correct !');
                    } else {
                        $response_question->score_percentage = 0;
                        $response_question->save();

                        Log::debug('Answer is wrong !');
                    }
                } elseif ($questionTypeId == 2) { //Обробка декількох відповідей на
вибір
                    Log::debug('Checking multiple options question...');

                    //Список вибраних відповідей
                    $selectedAnswers = [];

                    //Перебрати Collection в Array
                    foreach ($response_question_answers as $response_answer) {
                        $selectedAnswer = answer::find($response_answer->answer_id);

                        if ($selectedAnswer) {
                            $selectedAnswers[] = $selectedAnswer->id;
                        }
                    }

                    //Список правильних та неправильних відповідей
                    $correctAnswers = $question->answers()->where('is_correct', 1)-
>pluck('id')->toArray();
                    $correctAmount = sizeOf($correctAnswers);
                    $wrongAnswers = $question->answers()->where('is_correct', 0)-
>pluck('id')->toArray();
                    $wrongAmount = sizeOf($wrongAnswers);

                    //Кількість правильних та неправильних відповідей
                    $selectedCorrectAmount = sizeOf(array_intersect($selectedAnswers,
$correctAnswers));
                    $selectedWrongAmount = sizeOf(array_intersect($selectedAnswers,
$wrongAnswers));

                    //Загальна кількість відповідей
                    $totalAmount = $correctAmount + $wrongAmount;

                    //Перевірка збігу зі списками правильних та неправильних відповідей
                    $selectedCorrect = $selectedCorrectAmount + ($wrongAmount -
$selectedWrongAmount);

                    //Розрахована оцінка за кількістю вибраних правильних і неправильних
відповідей
```

```
                    $question_score = $selectedCorrect / $totalAmount;

                    //Зберегти оцінку
                    $response_question->score_percentage = $question_score * 100;
                    $response_question->save();

                    Log::debug('Answer is ' . ($question_score * 100) . '% correct');
                } elseif ($questionTypeId == 3) {   //Обробка текстової відповіді
                    Log::debug('Checking text answer question...');

                    //Список правильних відповідей
                    $correctAnswers = $question->answers()->where('is_correct', 1)-
>pluck('answer_text');

                    //Звірити вибрану відповідь зі списком правильних відповідей
                    foreach ($correctAnswers as $correctAnswer) {
                        //Перевірити чи відповідь правильна
                        if (mb_strtolower($response_question_answers->first()-
>answer_text) === mb_strtolower($correctAnswer)) {
                            Log::debug('Answer is correct !');

                            $response_question->score_percentage = 100;
                            $response_question->save();

                            break;
                        } else {
                            $response_question->score_percentage = 0;
                            $response_question->save();

                            Log::debug('Answer is wrong !');
                        }
                    }
                    Log::debug('Done checking text answer question...');
                }
            } else {
                Log::debug('No answer to question is given. Skipping question...');
            }
        }
        Log::debug("Results checking complete.\n");

        return $quiz_response;
    }

    public function check_video($quiz_response_id)
    {
        //Отримати відповідь для перевірки
        Log::debug('Checking video file...');
        $quiz_response = quiz_response::find($quiz_response_id);

        //Отримати відео для перевірки
        $video = $quiz_response->response_video;
        Log::debug('Grabbed video path: ' . $video);

        //Python та скрипт на виконання
        $pythonExecutable = env('PYTHON_EXECUTABLE_PATH', '');
        $pythonScript = public_path("python\GazeTracking\analyze_vision.py");

        $command = $pythonExecutable . ' ' . $pythonScript . ' ' . $video;
        Log::debug('Formed command: ' . $command);
        Log::debug('Executing...');

        $result = shell_exec($command);
```

```
        Log::debug('Executing result: ' . $result);

        $result = explode("\n", $result);

        $response_code = $result[0];
        $fps = $result[1];

        $center_seconds = $result[2];

        $offcenter_seconds = $result[3];
        $blink_seconds = $result[4];

        $quiz_response->total_time_offscreen = $offcenter_seconds + $blink_seconds;
        $quiz_response->save();

        Log::debug("Video file finished processing !\n");
        return $quiz_response;
    }
    //Вивести результати користувача
    public function show($quiz_response_id)
    {
        //Відповідь
        $quiz_response = quiz_response::find($quiz_response_id);

        //Отримати питання з відповіді
        $response_questions = $quiz_response->response_questions;

        //Оцінка тестування
        $totalScore = 0;

        foreach ($response_questions as $response_question) {
            $totalScore += $response_question->score_percentage;
        }

        //Вивести на сторінці
        return view('quiz_responses/show', compact('quiz_response', 'totalScore'));
    }
}
```

Лістинг user.php:

```
<?php

namespace App\Models;

// use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;
use Illuminate\Support\Facades\Log;

class User extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var array<int, string>
     */
    protected $fillable = [
        'name', 'email', 'password',
```

```
        'created_at', 'updated_at'
    ];

    /**
     * The attributes that should be hidden for serialization.
     *
     * @var array<int, string>
     */
    protected $hidden = [
        'password',
        'remember_token',
    ];

    /**
     * The attributes that should be cast.
     *
     * @var array<string, string>
     */
    protected $casts = [
        'email_verified_at' => 'datetime',
    ];
    //Ролі користувача
    public function roles()
    {
        return $this->hasMany(user_role::class, 'user_id');
    }
    //Тестування створені користувачем
    public function quizzes()
    {
        return $this->hasMany(quiz::class, 'user_id');
    }
    //Тестування пройдені користувачем
    public function quiz_responses()
    {
        return $this->hasMany(quiz_response::class, 'user_id');
    }

    //Перевірити права доступу користувача
    public function hasPermission($permissionName)
    {
        $userRoles = $this->roles;

        foreach ($userRoles as $userRole) {
            $rolePermissions = $userRole->role->permissions;
            foreach($rolePermissions as $rolePermission)
            {
                if ($rolePermission->permission->name == $permissionName)
                {
                    return True;
                }
            }
        }
        return false;
    }
}
```

Лістинг user_role.php:

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
```

```
use Illuminate\Database\Eloquent\Model;

class user_role extends Model
{
    use HasFactory;

    protected $table = 'user_roles';

    protected $fillable = array('role_id', 'user_id',
    'created_at', 'updated_at');
    //Роль
    public function role()
    {
        return $this->belongsTo(Role::class, 'role_id');
    }
    //Користувач, якому надається роль
    public function user()
    {
        return $this->belongsTo(User::class, 'user_id');
    }
}
```

Лістинг role.php:

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class role extends Model
{
    use HasFactory;

    protected $table = 'roles';
    protected $fillable = array('name', 'created_at', 'updated_at');

    //Дозволи конкретної ролі
    public function permissions(){
        return $this->hasmany(role_permission::class, 'role_id');
    }
}
```

Лістинг role_permission.php:

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class role_permission extends Model
{
    use HasFactory;

    protected $table = 'role_permissions';

    protected $fillable = array('role_id', 'permission_id', 'created_at', 'updated_at');
    //Роль
    public function role()
    {
```

```
        return $this->belongsTo(Role::class, 'role_id');
    }
    //Дозвіл
    public function permission()
    {
        return $this->belongsTo(Permission::class, 'permission_id');
    }
}
```

Лістинг permission.php:

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class permission extends Model
{
    use HasFactory;
    protected $table = 'permissions';

    protected $fillable = array('name', 'created_at', 'updated_at');
    //Ролі яким надається дозвіл
    public function roles()
    {
        return $this->hasMany(role_permission::class, 'permission_id');
    }
}
```

Лістинг subject.php:

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class subject extends Model
{
    use HasFactory;

    protected $table = 'subjects';

    protected $fillable = array('name', 'description',
    'cover_image', 'created_at', 'updated_at');
    //Тестування, що відносятся до предмету
    public function quizzes(){
        return $this->hasmany(quiz::class, 'subject_id');
    }
}
```

Лістинг quiz.php:

```
<?php
```

```php
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class quiz extends Model
{
    use HasFactory;

    protected $table = 'quizzes';

    protected $fillable = array('name', 'description', 'cover_image',
    'subject_id', 'user_id', 'created_at', 'updated_at');

    //Предмет, якому належить тестування
    public function subject()
    {
        return $this->belongsTo(subject::class, 'subject_id');
    }
    //Користувач, що створив тестування
    public function user()
    {
        return $this->belongsTo(user::class, 'user_id');
    }
    //Питання, що містяться в тестуванні
    public function questions()
    {
        return $this->hasMany(question::class, 'quiz_id');
    }
    //Результати проходження тестувань
    public function quiz_responses()
    {
        return $this->hasMany(quiz_response::class, 'quiz_id');
    }
}
```

Лістинг question.php:

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class question extends Model
{
    use HasFactory;

    protected $table = 'questions';

    protected $fillable = array('quiz_id','question_text', 'question_image',
'question_type_id', 'created_at', 'updated_at');
    //Тест якому належить питання
    public function quiz()
    {
        return $this->belongsTo(Quiz::class, 'quiz_id');
    }
    //Тип питання
    public function question_type()
    {
```

```
        return $this->belongsTo(question_type::class, 'question_type_id');
    }

    //Відповіді на питання
    public function answers()
    {
        return $this->hasMany(answer::class, 'question_id');
    }
    //Питання у пройдених тестуваннях
    public function response_questions()
    {
        return $this->hasMany(response_question::class, 'question_id');
    }
}
```

Лістинг question_type.php:

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class question_type extends Model
{
    use HasFactory;

    protected $table = 'question_types';

    protected $fillable = array('name', 'created_at', 'updated_at');
    //Питання, що належать цьому типу питань
    public function questions()
    {
        return $this->hasMany(Question::class, 'question_type_id');
    }
}
```

Лістинг answer.php:

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class answer extends Model
{
    use HasFactory;

    protected $table = 'answers';

    protected $fillable = array('answer_text', 'answer_image', 'question_id',
'is_correct', 'created_at', 'updated_at');

    //Питання в яких відповідь може бути обраною
    public function question()
    {
        return $this->belongsTo(question::class, 'question_id');
    }
```

```
    //Результати в яких відповідь була обраною
    public function response_answers()
    {
        return $this->hasMany(response_answer::class, 'answer_id');
    }
}
```

Лістинг quiz_response.php:

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class quiz_response extends Model
{
    use HasFactory;

    protected $table = 'quiz_responses';

    protected $fillable = array(
        'quiz_id', 'user_id',
        'response_video', 'total_time_offscreen',
        'created_at', 'updated_at');
    //Користувач, що пройшов тестування
    public function user()
    {
        return $this->belongsTo(user::class, 'user_id')->withDefault([
            'name' => 'Guest'   //Якщо користувач не увійшов в систему його результати
записуються як Гість.
        ]);
    }
    //Тестування, що проходиться
    public function quiz()
    {
        return $this->belongsTo(quiz::class, 'quiz_id');
    }
    //Відповіді обрані в тестуванні
    public function response_questions()
    {
        return $this->hasMany(response_question::class, 'response_id');
    }
}
```

Лістинг response_question.php:

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class response_question extends Model
{
    use HasFactory;

    protected $table = 'quiz_response_questions';
```

```
    protected $fillable = array('question_id', 'response_id', 'score_percentage',
'created_at', 'updated_at');

    //Питання на яке дається відповідь
    public function question()
    {
        return $this->belongsTo(question::class, 'question_id');
    }

    //Обєкт відповіді на тест
    public function response()
    {
        return $this->belongsTo(quiz_response::class, 'response_id');
    }

    //Відповіді на питання
    public function response_answers()
    {
        return $this->hasMany(response_answer::class, 'response_question_id');
    }
}
```

Лістинг response_answer.php:

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class response_answer extends Model
{
    use HasFactory;

    protected $table = 'quiz_response_answers';

    protected $fillable = array('response_question_id', 'answer_id', 'answer_text',
'created_at', 'updated_at');

    //Питання на яке дається відповідь
    public function response_question()
    {
        return $this->belongsTo(quiz_response::class, 'response_question_id');
    }
    //Обрана відповідь на питання
    public function answer()
    {
        return $this->belongsTo(answer::class, 'answer_id');
    }
}
```

Лістинг 2024_01_09_000000_create_users_table.php:

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
```

```
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();

            $table->string('name')->unique();
            $table->string('email')->unique();
            $table->timestamp('email_verified_at')->nullable();

            $table->string('password');
            $table->rememberToken();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('users');
    }
};
```

Лістинг 2024_01_12_103808_create_permissions_table.php:

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('permissions', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
```

```
     */
    public function down()
    {
        Schema::dropIfExists('permissions');
    }
};
```

Лістинг 2024_01_13_103751_create_roles_table.php:

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('roles', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('roles');
    }
};
```

Лістинг 2024_01_15_103811_create_user_roles_table.php:

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
```

```
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('user_roles', function (Blueprint $table) {
            $table->id();

            $table-> foreignId('role_id');
            $table->foreign('role_id')->references('id')->on('roles')-
>onDelete('CASCADE')->onUpdate('CASCADE');

            $table-> foreignId('user_id');
            $table->foreign('user_id')->references('id')->on('users')-
>onDelete('CASCADE')->onUpdate('CASCADE');

            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('user_roles');
    }
};
```

Лістинг 2024_01_16_055706_create_subjects_table.php:

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('subjects', function (Blueprint $table) {
            $table->id();
            $table->string('name');                    //Назва предмету
```

```
            $table->text('description')->nullable();    //Опис предмету
            $table->string('cover_image');              //зображення
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('subjects');
    }
};
```

Лістинг 2024_01_17_055940_create_quizzes_table.php:

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('quizzes', function (Blueprint $table) {
            $table->id();
            $table->string('name')->default('DefaultQuizName');//Назва тесту

            $table->text('description');    //опис тесту

            $table->string('cover_image');  //зображення

            $table->foreignId('subject_id');//Предмет
            $table->foreign('subject_id')->references('id')->on('subjects')-
>onDelete('CASCADE')->onUpdate('CASCADE');


            $table->foreignId('user_id');//Користувач, що створив тест
            $table->foreign('user_id')->references('id')->on('users')-
>onDelete('CASCADE')->onUpdate('CASCADE');

            $table->timestamps();
```

```
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('quizzes');
    }
};
```

Лістинг 2024_01_18_055948_create_question_types_table.php:

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('question_types', function (Blueprint $table) {
            $table->id();
            $table->string('name'); //Назва типу питання
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('question_types');
    }
};
```

Лістинг 2024_01_19_055958_create_questions_table.php:

```
<?php
```

```
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('questions', function (Blueprint $table) {
            $table->id();

            $table->foreignId('quiz_id');                    //ID Тесту
            $table->foreign('quiz_id')->references('id')->on('quizzes')-
>onDelete('CASCADE')->onUpdate('CASCADE');

            $table->string('question_text');                 //Текст завдання
            $table->string('question_image')->nullable();    //зображення

            $table->foreignId('question_type_id');           //Тип завдання
            $table->foreign('question_type_id')->references('id')->on('question_types')-
>onDelete('CASCADE')->onUpdate('CASCADE');

            $table->timestamps();                            //Час створення
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('questions');
    }
};
```

Лістинг 2024_01_20_060015_create_answers_table.php:

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
```

```
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('answers', function (Blueprint $table) {
            $table->id();

            $table->string('answer_text')->nullable();      //Текст відповіді
            $table->string('answer_image')->nullable();     //Зображення відповіді

            $table-> foreignId('question_id');              //ID Завдання
            $table->foreign('question_id')->references('id')->on('questions')-
>onDelete('CASCADE')->onUpdate('CASCADE');

            $table->boolean('is_correct');                   //Чи правильна відповідь

            $table->timestamps();                           //Час створення
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('answers');
    }
};
```

Лістинг 2024_01_21_224330_create_quiz_responses_table.php:

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('quiz_responses', function (Blueprint $table) {
```

```
            $table->id();//id
            $table->foreignId('quiz_id');//Тестування
            $table->foreign('quiz_id')->references('id')->on('quizzes')-
>onDelete('CASCADE')->onUpdate('CASCADE');

            $table->foreignId('user_id');//Користувач
            $table->foreign('user_id')->references('id')->on('users')-
>onDelete('CASCADE')->onUpdate('CASCADE');

            $table->string('response_video')->nullable();
            $table->time('total_time_offscreen')->nullable();//Загальний час поза
екраном

            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('quiz_responses');
    }
};
```

Лістинг 2024_01_22_210521_create_quiz_response_questions_table.php:

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('quiz_response_questions', function (Blueprint $table) {
            $table->id();

            $table->foreignId('response_id');//Питання
            $table->foreign('response_id')->references('id')->on('quiz_responses')-
>onDelete('CASCADE')->onUpdate('CASCADE');
```

```
            $table->foreignId('question_id');//Питання
            $table->foreign('question_id')->references('id')->on('questions')-
>onDelete('CASCADE')->onUpdate('CASCADE');

            $table->decimal('score_percentage', 6, 3)->nullable();//Скільки балів зі 100
було отримано

            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('quiz_response_results');
    }
};
```

Лістинг 2024_01_23_225720_create_quiz_response_answers_table.php:

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('quiz_response_answers', function (Blueprint $table) {
            $table->id();//id

            $table->foreignId('response_question_id');//Питання у пройденому тесті
            $table->foreign('response_question_id')->references('id')-
>on('quiz_response_questions')->onDelete('CASCADE')->onUpdate('CASCADE');

            $table->foreignId('answer_id')->nullable();//ID відповіді
            $table->foreign('answer_id')->references('id')->on('answers')-
>onDelete('CASCADE')->onUpdate('CASCADE');

            $table->string('answer_text')->nullable();//Текст відповіді
```

```
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('quiz_response_answers');
    }
};
```

Лістинг DatabaseSeeder.php:

```
<?php

namespace Database\Seeders;

use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
{
    /**
     * Seed the application's database.
     *
     * @return void
     */
    public function run()
    {
        $this->call(ConstantSeeder::class); //Константи|Налаштування

        $this->call(UserSeeder::class); //Створити користувачів

        $this->call(SubjectsSeeder::class); //Створити предмети

        $this->call(QuizSeeder::class); //Створити тестування

        $this->call(QuestionSeeder::class); //Створити питання для тестувань

        $this->call(AnswerSeeder::class); //Створити відповіді до питань тестів

        $this->call(QuizResponseSeeder::class); //Створити результати тестування
    }
}
```

Лістинг ConstantSeeder.php:

```
<?php

namespace Database\Seeders;

use App\Models\permission;
use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;
```

```php
use App\Models\question_type;
use App\Models\role;
use App\Models\role_permission;

class ConstantSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        //-----Статичні дані-----

        //Типи питань
        question_type::create([
            'name' => 'Одна відповідь на вибір',
        ]);

        question_type::create([
            'name' => 'Декілька відповідей на вибір',
        ]);

        question_type::create([
            'name' => 'Ввести відповідь з клавіатури',
        ]);

        //Типи доступу
        permission::create([
            'name' => 'Create'
        ]);
        permission::create([
            'name' => 'Edit'
        ]);
        permission::create([
            'name' => 'Delete'
        ]);
        permission::create([
            'name' => 'Take Quiz'
        ]);

        //Ролі
        role::create([
            'name' => 'Адміністратор'
        ]);
        role::create([
            'name' => 'Вчитель'
        ]);
        role::create([
            'name' => 'Учень'
```

```
        ]);


        //Ролі та доступи

        //Admin
        role_permission::create([
            'role_id' => 1,
            'permission_id' => 1
        ]);

        role_permission::create([
            'role_id' => 1,
            'permission_id' => 2
        ]);

        role_permission::create([
            'role_id' => 1,
            'permission_id' => 3
        ]);

        role_permission::create([
            'role_id' => 1,
            'permission_id' => 4
        ]);

        //Вчитель
        role_permission::create([
            'role_id' => 2,
            'permission_id' => 1
        ]);

        role_permission::create([
            'role_id' => 2,
            'permission_id' => 2
        ]);

        role_permission::create([
            'role_id' => 2,
            'permission_id' => 4
        ]);

        //Учень
        role_permission::create([
            'role_id' => 3,
            'permission_id' => 4
        ]);
    }
}
```

Лістинг UserSeeder.php:

```
<?php
```

```php
namespace Database\Seeders;

use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;

use Illuminate\Support\Facades\Hash;

use App\Models\User;
use App\Models\user_role;

class UserSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        //Профіль адміну
        User::create([
            'name' => 'admin',
            'email' => 'admin@example.com',
            'password' => Hash::make('admin'),
        ]);
        user_role::create([
            'user_id' => 1,
            'role_id' => 1
        ]);

        //Профіль вчителя
        User::create(
            [
                'name' => 'teacher',
                'email' => 'teacher@example.com',
                'password' => Hash::make('teacher'), // Encrypt the password using
bcrypt
            ]
        );
        user_role::create([
            'user_id' => 2,
            'role_id' => 2
        ]);
        //Профіль учня
        User::create(
            [
                'name' => 'student',
                'email' => 'student@example.com',
                'password' => Hash::make('student'), // Encrypt the password using
bcrypt
```

```
            ]
        );
        user_role::create([
            'user_id' => 3,
            'role_id' => 3
        ]);
    }
}
```

Лістинг SubjectsSeeder.php:

```
<?php

namespace Database\Seeders;

use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;
use App\Models\subject;

class SubjectsSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        subject::create([
            'name' => 'Математика 1 клас',
            'description' => 'Математика 1 клас (Нова українська школа)',
            'cover_image' => 'cover1.jpg',
        ]);

        subject::create([
            'name' => 'Математика 2 клас',
            'description' => 'Математика 2 клас (Нова українська школа)',
            'cover_image' => 'cover2.jpg'
        ]);
    }
}
```

Лістинг QuizSeeder.php:

```
<?php

namespace Database\Seeders;

use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;
use App\Models\quiz;
use App\Models\question;
use App\Models\answer;
```

```
class QuizSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        //Тести
        quiz::create([
            'name' => 'Основи математики 1',
            'description' => 'Цей тест перевірить ваші знання простих операцій у
математиці',
            'cover_image' => 'default.png',
            'subject_id' => '1',
            'user_id'=>'1'
        ]);
        quiz::create([
            'name' => 'Приклад тесту',
            'description' => 'Приклад тесту з усіма видами питань для перевірки
функціоналу системи',
            'cover_image' => 'default.png',
            'subject_id' => '2',
            'user_id'=>'2'
        ]);
    }
}
```

Лістинг QuestionSeeder.php:

```
<?php

namespace Database\Seeders;

use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;

use App\Models\question;

class QuestionSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        //Завдання
```

```
        //Тест_1
        question::create([
            'quiz_id' => '1',
            'question_text' => 'Скільки буде 2+2',
            'question_type_id' => '1'
        ]);
        question::create([
            'quiz_id' => '1',
            'question_text' => 'Розвязати рівняння 3x + 5 = 17',
            'question_type_id' => '1'
        ]);
        question::create([
            'quiz_id' => '1',
            'question_text' => 'Яка площа квадрата зі стороною довжини 6',
            'question_type_id' => '1'
        ]);
        question::create([
            'quiz_id' => '1',
            'question_text' => 'Спростіть вираз 2(4 + 3) - 5',
            'question_type_id' => '1'
        ]);
        question::create([
            'quiz_id' => '1',
            'question_text' => 'Яке значення числа π',
            'question_type_id' => '1'
        ]);

        //Тест 2
        question::create([
            'quiz_id' => '2',
            'question_text' => 'Одна відповідь на вибір (правильна відповідь:3)',
            'question_type_id' => '1'
        ]);
        question::create([
            'quiz_id' => '2',
            'question_text' => 'Багато відповідей на вибір (правильна відповідь:
1,3,4)',
            'question_type_id' => '2'
        ]);
        question::create([
            'quiz_id' => '2',
            'question_text' => 'Текстова відповідь (правильна відповідь: Win)',
            'question_type_id' => '3'
        ]);
    }
}
```

Лістинг AnswerSeeder.php:

```
<?php

namespace Database\Seeders;
```

```php
use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;

use App\Models\answer;

class AnswerSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        //Відповіді

        //--1 тест--

        //1 Питання
        answer::create([
            'question_id' => '1',
            'answer_text' => '3',
            'is_correct' => '0'
        ]);
        answer::create([
            'question_id' => '1',
            'answer_text' => '4',
            'is_correct' => '1'
        ]);
        answer::create([
            'question_id' => '1',
            'answer_text' => '5',
            'is_correct' => '0'
        ]);
        answer::create([
            'question_id' => '1',
            'answer_text' => '6',
            'is_correct' => '0'
        ]);

        //2 Питання
        answer::create([
            'question_id' => '2',
            'answer_text' => '3',
            'is_correct' => '0'
        ]);
        answer::create([
            'question_id' => '2',
            'answer_text' => '4',
            'is_correct' => '1'
```

```
]);
answer::create([
    'question_id' => '2',
    'answer_text' => '5',
    'is_correct' => '0'
]);
answer::create([
    'question_id' => '2',
    'answer_text' => '6',
    'is_correct' => '0'
]);

//3 Питання
answer::create([
    'question_id' => '3',
    'answer_text' => '12',
    'is_correct' => '0'
]);
answer::create([
    'question_id' => '3',
    'answer_text' => '18',
    'is_correct' => '0'
]);
answer::create([
    'question_id' => '3',
    'answer_text' => '24',
    'is_correct' => '0'
]);
answer::create([
    'question_id' => '3',
    'answer_text' => '36',
    'is_correct' => '1'
]);

//4 Питання
answer::create([
    'question_id' => '4',
    'answer_text' => '6',
    'is_correct' => '0'
]);
answer::create([
    'question_id' => '4',
    'answer_text' => '8',
    'is_correct' => '0'
]);
answer::create([
    'question_id' => '4',
    'answer_text' => '9',
    'is_correct' => '1'
]);
answer::create([
```

```
    'question_id' => '4',
    'answer_text' => '12',
    'is_correct' => '0'
]);

//5 Питання
answer::create([
    'question_id' => '5',
    'answer_text' => '2.71',
    'is_correct' => '0'
]);
answer::create([
    'question_id' => '5',
    'answer_text' => '3.14',
    'is_correct' => '1'
]);
answer::create([
    'question_id' => '5',
    'answer_text' => '3.50',
    'is_correct' => '0'
]);
answer::create([
    'question_id' => '5',
    'answer_text' => '4.20',
    'is_correct' => '0'
]);

//Тест 2

//1 Питання
answer::create([
    'question_id' => '6',
    'answer_text' => '1',
    'is_correct' => '0'
]);
answer::create([
    'question_id' => '6',
    'answer_text' => '2',
    'is_correct' => '0'
]);
answer::create([
    'question_id' => '6',
    'answer_text' => '3',
    'is_correct' => '1'
]);
answer::create([
    'question_id' => '6',
    'answer_text' => '4',
    'is_correct' => '0'
]);
```

```
        //2 Питання
        answer::create([
            'question_id' => '7',
            'answer_text' => '1',
            'is_correct' => '1'
        ]);
        answer::create([
            'question_id' => '7',
            'answer_text' => '2',
            'is_correct' => '0'
        ]);
        answer::create([
            'question_id' => '7',
            'answer_text' => '3',
            'is_correct' => '1'
        ]);
        answer::create([
            'question_id' => '7',
            'answer_text' => '4',
            'is_correct' => '1'
        ]);

        //3 Питання
        answer::create([
            'question_id' => '8',
            'answer_text' => 'Win',
            'is_correct' => '1'
        ]);
    }
}
```

Лістинг QuizResponseSeeder.php:

```
<?php

namespace Database\Seeders;

use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;
use App\Models\quiz_response;
use App\Models\response_answer;

class QuizResponseSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        //
```

```
    }
}
```

Лістинг quiz_take.js:

```
'use strict';

//Глобальні змінні
let log = console.log.bind(console),
  stream,
  recorder,
  chunks,
  media;

$(document).ready(function () {
  disableQuizControls();
  startRecording();
});
//Блокування елементів тестування на сторінці
function disableQuizControls() {
  document.getElementById('submit-quiz').disabled = false;

  const quizOptionsDivs = document.getElementsByClassName('quiz_options');

  // Loop through each 'options' div
  Array.from(quizOptionsDivs).forEach(optionsDiv => {
    // Get all child elements of the current 'options' div
    const quizOptions = optionsDiv.querySelectorAll('*');

    // Disable each quiz option
    quizOptions.forEach(option => {
      option.disabled = true;
    });

  });
}

//Розблокування елементів тестування на сторінці
function enableQuizControls() {
  document.getElementById('submit-quiz').disabled = false;

  const quizOptionsDivs = document.getElementsByClassName('quiz_options');

  // Loop through each 'options' div
  Array.from(quizOptionsDivs).forEach(optionsDiv => {
    // Get all child elements of the current 'options' div
    const quizOptions = optionsDiv.querySelectorAll('*');

    // Disable each quiz option
    quizOptions.forEach(option => {
      option.disabled = false;
    });

  });
}

//Підготовка до запису
function InitMedia() {

  return new Promise((resolve) => {
    media = {
      tag: 'video',
```

```
      type: 'video/webm',
      ext: '.webm',
      gUM: { video: true }
    };
    navigator.mediaDevices.getUserMedia(media.gUM).then(_stream => {
      stream = _stream;

      //Налаштування і запит доступу до медіа потоків
      recorder = new MediaRecorder(stream);
      recorder.ondataavailable = e => {
        chunks.push(e.data);
        if (recorder.state == 'inactive') { //Дії при припиненні запису відео
          stopMediaTracks(); //Припинити доступ до камери користувача
          sendData();  //Відправити відео
        }
      };
      log('got media successfully');
      resolve();
      enableQuizControls();
    }).catch(log)
  });
}
//Почати запис
function startRecording() {
  //За потреби налаштувати медіа потоки
  log('startRecording call');

  if (stream == null) {
    log('InitMedia call');
    InitMedia()
      .then(() => {
        log('After InitMedia call');
        //Почати запис
        chunks = [];
        recorder.start();
        log('After recorder start')
      });
  }
}

//Припинити запис
function stopRecording() {
  if (chunks != null) {
    stop.disabled = true;
    recorder.stop();
  }
  log("Recording stopped.")
}

function stopMediaTracks() {
  stream.getTracks().forEach(function (track) {
    track.stop();
  });
  log("Media access stopped.")
}

//Перетворюємо дані з форми в обєкт
function formDataToObject(formData) {
  const object = {};
  formData.forEach((value, key) => {
    object[key] = value;
  });
  return object;
```

```
}

function sendData() {
  log("Preparing data...")
  //Компіляція Blob
  log("Preparing video file...")
  let blob = new Blob(chunks, { type: media.type });
  blob.name = 'video' + media.ext;
  log("Video file ready.")
  //Формування запиту
  const quizForm = document.getElementById('quizForm'); //Обєкт форми
  const formDest = quizForm.getAttribute('action'); //Посилання з форми
  const formMethod = quizForm.getAttribute('method');
  log("Preparing form data...")
  const formData = new FormData(quizForm);  //Дані з форми
  log(quizForm)
  log(formData)
  log("Form data compiled.")
  //formData.append('file', blob); //Додати відео до форми
  //Так не можна бо дані з форми щоразу відправляються в запиті, таким чином
  //замість відправлення файлу по частинах ми його відправляємо 100500 разів в повному
обємі разом з його
  //розбитими частинами і перевантажуємо систему

  //Відправлення запиту

  //Завантаження v3
  //Переключення на Resumable для завантаження по частинах
  //Можливість завантаження файлів
  log("Preparing to send data...")
  let resumable = new Resumable({
    target: formDest,
    query: formDataToObject(formData),
    chunkSize: 1 * 1024 * 1024, // default is 1*1024*1024, this should be less than your
maximum limit in php.ini
    headers: {
      'Accept': 'application/json'
    },
    testChunks: false,
    throttleProgressCallbacks: 1,
  });

  //Почати завантаження коли є наявний файл
  resumable.on('fileAdded', function (file) {
    showProgress();
    resumable.upload()
  });

  //Подія оновлення прогресу завантаження
  resumable.on('fileProgress', function (file) {
    updateProgress(Math.floor(file.progress() * 100));
  });

  //Подія успішного завантаження
  resumable.on('fileSuccess', function (file, response) {
    log("File uploaded successfully");
    response = JSON.parse(response)
    log(response)
    //Перейти на сторінку результатів
    window.location.href = response;
  });

  //Помилка завантаження
```

```
  resumable.on('fileError', function (file, response) { // trigger when there is any
error
    alert('file uploading error.'.response);
  });
  log("Ready to send...")
  resumable.addFile(blob);
  log("Sending...")
}

let progress = $('.progress');

//Відображення полоски прогресу
function showProgress() {
  progress.find('.progress-bar').css('width', '0%');
  progress.find('.progress-bar').html('0%');
  progress.find('.progress-bar').removeClass('bg-success');
  progress.show();
}
//Оновлення полоски прогресу
function updateProgress(value) {
  progress.find('.progress-bar').css('width', `${value}%`)
  progress.find('.progress-bar').html(`${value}%`)
  log(`${value}%`);

  if (value === 100) {
    progress.find('.progress-bar').addClass('bg-success');
  }
}
//Приховати полоску прогресу
function hideProgress() {
  progress.hide();
}

//Обробка блокування доступу до медіа
function handleMediaAccessDenied() {
  log('User denied access to the camera');

  const message = 'Для проходження тесту будь ласка, увімкніть камеру та перезавантажте
сторінку';
  alert(message);
}
```

Лістинг analyze_vision.py:

```
import sys
import cv2
from gaze_tracking import GazeTracking

gaze = GazeTracking()

try:
    videoStream = cv2.VideoCapture(sys.argv[1]) #Отримати посилання на відео
except:
    print("Error: Video path argument is invalid or empty.")
    sys.exit()

frame_rate = videoStream.get(cv2.CAP_PROP_FPS)
print(frame_rate)   #1 - FPS

#Розміри матриці відеопотоку
```

```
width, height = videoStream.get(3), videoStream.get(4)

center_frames = 0
offCenter_frames = 0
blink_frames = 0

while True:
    #Взяти зображення з відеопотоку
    _success, frame = videoStream.read()

    #Якщо відео кадри закінчилися спинити програму
    if  (not _success):
        break

    #Аналізувати зображення
    gaze.refresh(frame)

    #Перевіриити положення очей
    if  gaze.is_blinking_both():
        blink_frames = blink_frames+1
    elif gaze.is_blinking_left():
        blink_frames = blink_frames+1
    elif gaze.is_blinking_right():
        blink_frames = blink_frames+1
    elif gaze.is_looking_center():
        center_frames = center_frames+1
    elif gaze.is_looking_right():
        offCenter_frames = offCenter_frames+1
    elif gaze.is_looking_left():
        offCenter_frames = offCenter_frames+1
    elif gaze.is_looking_up():
        offCenter_frames = offCenter_frames+1
    elif gaze.is_looking_down():
        offCenter_frames = offCenter_frames+1

#Обрахувати час за кількістю кадрів
center_seconds = center_frames/frame_rate
offCenter_seconds = offCenter_frames/frame_rate
blink_seconds = blink_frames/frame_rate

#Вивести дані аналізу
print(center_seconds)   #2  -   Секунди погляду у центрі
print(offCenter_seconds)    #3  -   Секунди погляду поза центром
print(blink_seconds)    #4  -   Секунди закритих очей

videoStream.release()
sys.exit()
```

Лістинг gaze_tracking.py:

```
from __future__ import division
import os
```

```
import cv2
import dlib
from .eye_full import Eye_full
from .calibration import Calibration

class GazeTracking(object):
    """
    Основний клас відстеження очей. Головними функціями є
    відстеження координат очей та напрямку погляду
    """

    def __init__(self):
        self.frame = None
        self.eye_left = None
        self.eye_right = None
        self.calibration = Calibration()

        # _face_detector is used to detect faces
        self._face_detector = dlib.get_frontal_face_detector()

        # _predictor is used to get facial landmarks of a given face
        cwd = os.path.abspath(os.path.dirname(__file__))
        model_path = os.path.abspath(os.path.join(cwd,
"trained_models/shape_predictor_68_face_landmarks.dat"))
        self._predictor = dlib.shape_predictor(model_path)

    @property
    def pupil_left_located(self):
        #Перевірити чи було лівий зрачок
        try:
            int(self.eye_left.visible.pupil.x)
            int(self.eye_left.visible.pupil.y)
            return True
        except Exception:
            return False

    @property
    def pupil_right_located(self):
        #Перевірити чи було правий зрачок
        try:
            int(self.eye_right.visible.pupil.x)
            int(self.eye_right.visible.pupil.y)
            return True
        except Exception:
            return False

    @property
    def both_pupils_located(self):
        #Перевірити чи було знайдено обидва зрачки

        return self.pupil_left_located and self.pupil_right_located
```

```
def _analyze(self):
    #Знайти обличчя та очі
    frame = cv2.cvtColor(self.frame, cv2.COLOR_BGR2GRAY)
    faces = self._face_detector(frame)

    try:
        #Створити окремі обєкти для кожного ока
        landmarks = self._predictor(frame, faces[0])
        self.eye_left = Eye_full(frame, landmarks, 0, self.calibration)
        self.eye_right = Eye_full(frame, landmarks, 1, self.calibration)

    except IndexError:
        self.eye_left = None
        self.eye_right = None

def refresh(self, frame):
    """Refreshes the frame and analyzes it.
    Arguments:
        frame (numpy.ndarray): The frame to analyze
    """
    self.frame = frame
    self._analyze()

def pupil_left_coords(self):
    #Координати лівого зрачка
    if self.pupil_left_located:
        x = self.eye_left.visible.origin[0] + self.eye_left.visible.pupil.x
        y = self.eye_left.visible.origin[1] + self.eye_left.visible.pupil.y
        return (x, y)

def pupil_right_coords(self):
    #Координати правого зрачка
    if self.pupil_right_located:
        x = self.eye_right.visible.origin[0] + self.eye_right.visible.pupil.x
        y = self.eye_right.visible.origin[1] + self.eye_right.visible.pupil.y
        return (x, y)

#
#Відносні координати зрачків до зображення
#
def pupil_in_frame_horizontal_ratio(self):
    """Повертаємо число від 0.0 до 1.0, що представляє собою
    відносну горизонтальну координату зрачків на зображенні.
    Ліва межа зображення це: 0.0, центр: 0.5 та права межа: 1.0
    """
    if self.both_pupils_located:
        #Відношення координат зрачків до довжини зображення
        frameSize = self.frame.shape[:2]

        #Глобальні координати зрачків
```

```
            pupil_left_x = self.eye_left.visible.pupil.x+self.eye_left.visible.origin[0]
            pupil_right_x =
self.eye_right.visible.pupil.x+self.eye_right.visible.origin[0]

            #Порівняти координати з позицією на зображенні
            pupil_left = pupil_left_x / (frameSize[1])
            pupil_right = pupil_right_x / (frameSize[1])

            #Вивести середнє значення
            ratio = (pupil_left + pupil_right) / 2
            return ratio

    def pupil_in_frame_vertical_ratio(self):
        """Повертаємо число від 0.0 до 1.0, що представляє собою
        відносну вертикальну координату зрачків на зображенні.
        Верхня межа зображення це: 0.0, центр: 0.5 та нижня межа: 1.0
        """
        if self.both_pupils_located:
            #Відношення координат зрачків до висоти зображення
            frameSize = self.frame.shape[:2]

            #Глобальні координати зрачків
            pupil_left_y = self.eye_left.visible.pupil.y+self.eye_left.visible.origin[1]
            pupil_right_y =
self.eye_right.visible.pupil.y+self.eye_right.visible.origin[1]

            #Порівняти координати з позицією на зображенні
            pupil_left = pupil_left_y / (frameSize[0])
            pupil_right = pupil_right_y / (frameSize[0])

            #Вивести середнє значення
            ratio = (pupil_left + pupil_right) / 2
            return ratio

    #Набір простих функцій для виведення позиції зрачків на екрані
    def is_located_left(self):
        #Зрачки у лівій частині зображення
        if self.both_pupils_located:
            return self.pupil_in_frame_horizontal_ratio() <= 0.35

    def is_located_right(self):
        #Зрачки у правій частині зображення
        if self.both_pupils_located:
            return self.pupil_in_frame_horizontal_ratio() >= 0.65

    def is_located_top(self):
        #Зрачки у правій частині зображення
        if self.both_pupils_located:
            return self.pupil_in_frame_vertical_ratio() <= 0.35

    def is_located_bottom(self):
```

```
        #Зрачки у правій частині зображення
        if self.both_pupils_located:
            return self.pupil_in_frame_vertical_ratio() >= 0.65

    def is_located_center(self):
        #Зрачки у центральній частині екрану
        if self.both_pupils_located:
            return not self.is_located_right() and not self.is_located_left() and not
self.is_located_top() and not self.is_located_bottom()


    #
    #Відносні координати зрачків до очей
    #
    def pupil_in_eye_horizontal_ratio(self):
        """Повертаємо число від 0.0 до 1.0, що представляє собою
        відносну горизонтальну координату зрачків на очах.
        Ліва межа очей це: 0.0, центр: 0.5 та права межа: 1.0
        """
        if self.both_pupils_located:
            #Відношення координат зрачків до довжини очей
            leftFrameSize = self.eye_left.visible.frame.shape[:2]
            rightFrameSize = self.eye_right.visible.frame.shape[:2]

            #Відносні координати зрачків
            pupil_left_x = self.eye_left.visible.pupil.x
            pupil_right_x = self.eye_right.visible.pupil.x

            #Порівняти координати з позицією в очах
            pupil_left = pupil_left_x / (leftFrameSize[1])
            pupil_right = pupil_right_x / (rightFrameSize[1])

            #Вивести середнє значення
            ratio = (pupil_left + pupil_right) / 2
            return ratio

    def pupil_in_eye_vertical_ratio(self):
        """Повертаємо число від 0.0 до 1.0, що представляє собою
        відносну вертикальну координату зрачків на очах.
        Верхня межа зображення це: 0.0, центр: 0.5 та нижня межа: 1.0
        """
        if self.both_pupils_located:
            #Розміри зображеннь повного ока
            leftFrameSize = self.eye_left.frame.shape[:2]
            rightFrameSize = self.eye_right.frame.shape[:2]

            #Відносні координати зрачків
            pupil_left_y = self.pupil_left_coords()[1] - self.eye_left.origin[1]
            pupil_right_y = self.pupil_right_coords()[1] - self.eye_right.origin[1]

            #Порівняти координати з позицією в очах
```

```
            pupil_left = pupil_left_y / (leftFrameSize[0])
            pupil_right = pupil_right_y / (rightFrameSize[0])

            #Вивести середнє значення
            ratio = (pupil_left + pupil_right) / 2
            return ratio

    #Набір простих функцій для виведення напрямку погляду
    def is_looking_left(self):
        #Зрачки у лівій частині очей
        if self.both_pupils_located:
            return self.pupil_in_eye_horizontal_ratio() <= 0.4

    def is_looking_right(self):
        #Зрачки у правій частині очей
        if self.both_pupils_located:
            return self.pupil_in_eye_horizontal_ratio() >= 0.6

    def is_looking_up(self):
        #Зрачки у правій частині очей
        if self.both_pupils_located:
            return self.pupil_in_eye_vertical_ratio() <= 0.675

    def is_looking_down(self):
        #Зрачки у правій частині очей
        if self.both_pupils_located:
            return self.pupil_in_eye_vertical_ratio() >= 0.78

    def is_looking_center(self):
        #Зрачки у центрі очей
        if self.both_pupils_located:
            return not self.is_looking_right() and not self.is_looking_left() and not
self.is_looking_up() and not self.is_looking_down()

    #
    #Перевірки чи закриті очі
    #
    def is_blinking_left(self):
        #Чи закрите ліве око
        if not self.pupil_left_located:
            return True
        #else:
        #    return self.eye_left.visible.blinking > 6.75

    def is_blinking_right(self):
        #Чи закрите праве око
        if not self.pupil_right_located:
            return True
        #else:
        #    return self.eye_right.visible.blinking > 6.75
```

```
    def is_blinking_both(self):
        #Чи закриті обидва ока
        return self.is_blinking_left() and self.is_blinking_right()

    #Повертаємо зображення з позначеними очима і зрачками
    def annotated_frame(self):
        frame = self.frame.copy()

        #Позначити ліве око прямокутником
        if(self.eye_left != None):
            color = (0, 0, 255)
            cv2.rectangle(frame, self.eye_left.origin, self.eye_left.endpoint, color, 1)
            if(self.eye_left.visible != None):
                color = (0, 255, 0)
                cv2.rectangle(frame, self.eye_left.visible.origin,
self.eye_left.visible.endpoint, color, 1)
                #Позначити правий зрачок плюсом
                if self.pupil_left_located:
                    color = (255, 0, 0)
                    x, y = self.pupil_left_coords()
                    cv2.line(frame, (x - 5, y), (x + 5, y), color)
                    cv2.line(frame, (x, y - 5), (x, y + 5), color)

        #Позначити ліве око прямокутником
        if(self.eye_right != None):
            color = (0, 0, 255)
            cv2.rectangle(frame, self.eye_right.origin, self.eye_right.endpoint, color,
1)
            if(self.eye_right.visible != None):
                color = (0, 255, 0)
                cv2.rectangle(frame, self.eye_right.visible.origin,
self.eye_right.visible.endpoint, color, 1)
                #Позначити правий зрачок плюсом
                if self.pupil_right_located:
                    color = (255, 0, 0)
                    x, y = self.pupil_right_coords()
                    cv2.line(frame, (x - 5, y), (x + 5, y), color)
                    cv2.line(frame, (x, y - 5), (x, y + 5), color)
        return frame
```

Лістинг calibration.py:

```
from __future__ import division
import cv2
from .pupil import Pupil

class Calibration(object):
    """
    This class calibrates the pupil detection algorithm by finding the
    best binarization threshold value for the person and the webcam.
    """
```

```
    def __init__(self):
        self.nb_frames = 20
        self.thresholds_left = []
        self.thresholds_right = []

    def is_complete(self):
        """Returns true if the calibration is completed"""
        return len(self.thresholds_left) >= self.nb_frames and
len(self.thresholds_right) >= self.nb_frames

    def threshold(self, side):
        """Returns the threshold value for the given eye.

        Argument:
            side: Indicates whether it's the left eye (0) or the right eye (1)
        """
        if side == 0:
            return int(sum(self.thresholds_left) / len(self.thresholds_left))
        elif side == 1:
            return int(sum(self.thresholds_right) / len(self.thresholds_right))

    @staticmethod
    def iris_size(frame):
        """Returns the percentage of space that the iris takes up on
        the surface of the eye.

        Argument:
            frame (numpy.ndarray): Binarized iris frame
        """
        frame = frame[5:-5, 5:-5]
        height, width = frame.shape[:2]
        nb_pixels = height * width
        nb_blacks = nb_pixels - cv2.countNonZero(frame)
        return nb_blacks / nb_pixels

    @staticmethod
    def find_best_threshold(eye_frame):
        """Calculates the optimal threshold to binarize the
        frame for the given eye.

        Argument:
            eye_frame (numpy.ndarray): Frame of the eye to be analyzed
        """

        average_iris_size = 0.54
        trials = {}

        for threshold in range(5, 100, 5):
            iris_frame = Pupil.image_processing(eye_frame, threshold)
            trials[threshold] = Calibration.iris_size(iris_frame)
```

```
        best_threshold, iris_size = min(trials.items(), key=(lambda p: abs(p[1] -
average_iris_size)))
        return best_threshold

    def evaluate(self, eye_frame, side):
        """Improves calibration by taking into consideration the
        given image.

        Arguments:
            eye_frame (numpy.ndarray): Frame of the eye
            side: Indicates whether it's the left eye (0) or the right eye (1)
        """
        threshold = self.find_best_threshold(eye_frame)

        if side == 0:
            self.thresholds_left.append(threshold)
        elif side == 1:
            self.thresholds_right.append(threshold)

    def reset(self):
        self.thresholds_left = []
        self.thresholds_right = []
```

Лістинг eye_full.py:

```
import math
import numpy as np
import cv2
from .pupil import Pupil
from .eye_visible import Eye_visible

class Eye_full(object):
    """
    Цей клас ізолює окреме око та брову над ним
    Потім ми створюємо окремий обєкт видимого ока
    """

    LEFT_EYE_POINTS = list(range(36, 42))+list(range(17,22))
    RIGHT_EYE_POINTS = list(range(42, 48))+list(range(22,27))

    @staticmethod
    def _middle_point(p1, p2):
        """Returns the middle point (x,y) between two points

        Arguments:
            p1 (dlib.point): First point
            p2 (dlib.point): Second point
        """
        x = int((p1.x + p2.x) / 2)
        y = int((p1.y + p2.y) / 2)
        return (x, y)
```

```
    def __init__(self, original_frame, landmarks, side, calibration):
        #Сторона ока
        self.side = side
        #Розміри початкового зображення
        self.width = None
        self.heigth = None
        #Обрізане зображення ока
        self.frame = None
        #Початкова точка зображення
        self.origin = None
        #Кінечна точка зображення
        self.endpoint = None

        #Точка центру ока
        self.center = None
        #Видима частина ока
        self.visible = None

        #Орієнтири
        self.landmark_points = None

        self._analyze(original_frame, landmarks, side, calibration)

    def _isolate(self, frame, landmarks, points):
        """Isolate an eye, to have a frame without other part of the face.

        Arguments:
            frame (numpy.ndarray): Frame containing the face
            landmarks (dlib.full_object_detection): Facial landmarks for the face region
            points (list): Points of an eye (from the 68 Multi-PIE landmarks)
        """
        region = np.array([(landmarks.part(point).x, landmarks.part(point).y) for point
in points])
        region = region.astype(np.int32)
        self.landmark_points = region

        # Applying a mask to get only the eye
        height, width = frame.shape[:2]
        black_frame = np.zeros((height, width), np.uint8)
        mask = np.full((height, width), 255, np.uint8)
        cv2.fillPoly(mask, [region], (0, 0, 0))
        eye = cv2.bitwise_not(black_frame, frame.copy(), mask=mask)

        # Cropping on the eye
        margin = 5
        min_x = np.min(region[:, 0]) - margin
        max_x = np.max(region[:, 0]) + margin
        min_y = np.min(region[:, 1]) - margin
        max_y = np.max(region[:, 1]) + margin
```

```
        self.frame = eye[min_y:max_y, min_x:max_x]
        self.origin = (min_x, min_y)
        self.endpoint = (max_x, max_y)

        center_x = (min_x + max_x) / 2
        center_y = (min_y + max_y) / 2

        self.height, self.width = self.frame.shape[:2]
        self.center = (center_x, center_y)

    def _eyebrow_ratio(self, landmarks, points):
        """Calculates a ratio that can indicate whether an eye is closed or not.
        It's the division of the width of the eye, by its height.

        Arguments:
            landmarks (dlib.full_object_detection): Facial landmarks for the face region
            points (list): Points of an eye (from the 68 Multi-PIE landmarks)

        Returns:
            The computed ratio
        """
        left = (landmarks.part(points[0]).x, landmarks.part(points[0]).y)
        right = (landmarks.part(points[3]).x, landmarks.part(points[3]).y)
        top = self._middle_point(landmarks.part(points[1]), landmarks.part(points[2]))
        bottom = self._middle_point(landmarks.part(points[5]),
landmarks.part(points[4]))

        eye_width = math.hypot((left[0] - right[0]), (left[1] - right[1]))
        eye_height = math.hypot((top[0] - bottom[0]), (top[1] - bottom[1]))

        try:
            ratio = eye_width / eye_height
        except ZeroDivisionError:
            ratio = None

        return ratio

    def _analyze(self, original_frame, landmarks, side, calibration):
        """Detects and isolates the eye in a new frame, sends data to the calibration
        and initializes Eye_visible object.

        Arguments:
            original_frame (numpy.ndarray): Frame passed by the user
            landmarks (dlib.full_object_detection): Facial landmarks for the face region
            side: Indicates whether it's the left eye (0) or the right eye (1)
            calibration (calibration.Calibration): Manages the binarization threshold
value
        """
        if side == 0:
            points = self.LEFT_EYE_POINTS
        elif side == 1:
```

```
            points = self.RIGHT_EYE_POINTS
        else:
            return

        self.eyebrow = self._eyebrow_ratio(landmarks, points)
        self._isolate(original_frame, landmarks, points)

        self.visible = Eye_visible(original_frame, landmarks, side, calibration)
```

Лістинг eye_visible.py:

```
import math
import numpy as np
import cv2
from .pupil import Pupil


class Eye_visible(object):
    """
    Цей клас ізолює видиму частину ока та створює обєкт зрачка
    """

    LEFT_EYE_POINTS = list(range(36, 42))
    RIGHT_EYE_POINTS = list(range(42, 48))

    @staticmethod
    def _middle_point(p1, p2):
        """Returns the middle point (x,y) between two points

        Arguments:
            p1 (dlib.point): First point
            p2 (dlib.point): Second point
        """
        x = int((p1.x + p2.x) / 2)
        y = int((p1.y + p2.y) / 2)
        return (x, y)

    def __init__(self, original_frame, landmarks, side, calibration):
        #Розміри початкового зображення
        self.width = None
        self.heigth = None
        #Обрізане зображення ока
        self.frame = None
        #Початкова точка зображення
        self.origin = None
        #Кінечна точка зображення
        self.endpoint = None

        #Точка центру ока
        self.center = None
        #Зрачок
        self.pupil = None
        #Кліпання ока
```

```
        self.blinking = None

        #Орієнтири
        self.landmark_points = None

        self._analyze(original_frame, landmarks, side, calibration)

    def _isolate(self, frame, landmarks, points):
        """Isolate an eye, to have a frame without other part of the face.

        Arguments:
            frame (numpy.ndarray): Frame containing the face
            landmarks (dlib.full_object_detection): Facial landmarks for the face region
            points (list): Points of an eye (from the 68 Multi-PIE landmarks)
        """
        region = np.array([(landmarks.part(point).x, landmarks.part(point).y) for point
in points])
        region = region.astype(np.int32)
        self.landmark_points = region

        # Applying a mask to get only the eye
        height, width = frame.shape[:2]
        black_frame = np.zeros((height, width), np.uint8)
        mask = np.full((height, width), 255, np.uint8)
        cv2.fillPoly(mask, [region], (0, 0, 0))
        eye = cv2.bitwise_not(black_frame, frame.copy(), mask=mask)

        # Cropping on the eye
        margin = 5
        min_x = np.min(region[:, 0]) - margin
        max_x = np.max(region[:, 0]) + margin
        min_y = np.min(region[:, 1]) - margin
        max_y = np.max(region[:, 1]) + margin

        self.frame = eye[min_y:max_y, min_x:max_x]
        self.origin = (min_x, min_y)
        self.endpoint = (max_x, max_y)

        center_x = (min_x + max_x) / 2
        center_y = (min_y + max_y) / 2

        self.height, self.width = self.frame.shape[:2]
        self.center = (center_x, center_y)

    def _blinking_ratio(self, landmarks, points):
        """Calculates a ratio that can indicate whether an eye is closed or not.
        It's the division of the width of the eye, by its height.

        Arguments:
            landmarks (dlib.full_object_detection): Facial landmarks for the face region
            points (list): Points of an eye (from the 68 Multi-PIE landmarks)
```

```
        Returns:
            The computed ratio
        """
        left = (landmarks.part(points[0]).x, landmarks.part(points[0]).y)
        right = (landmarks.part(points[3]).x, landmarks.part(points[3]).y)
        top = self._middle_point(landmarks.part(points[1]), landmarks.part(points[2]))
        bottom = self._middle_point(landmarks.part(points[5]),
landmarks.part(points[4]))

        eye_width = math.hypot((left[0] - right[0]), (left[1] - right[1]))
        eye_height = math.hypot((top[0] - bottom[0]), (top[1] - bottom[1]))

        try:
            ratio = eye_width / eye_height
        except ZeroDivisionError:
            ratio = None

        return ratio

    def _analyze(self, original_frame, landmarks, side, calibration):
        """Detects and isolates the eye in a new frame, sends data to the calibration
        and initializes Pupil object.

        Arguments:
            original_frame (numpy.ndarray): Frame passed by the user
            landmarks (dlib.full_object_detection): Facial landmarks for the face region
            side: Indicates whether it's the left eye (0) or the right eye (1)
            calibration (calibration.Calibration): Manages the binarization threshold
value
        """
        if side == 0:
            points = self.LEFT_EYE_POINTS
        elif side == 1:
            points = self.RIGHT_EYE_POINTS
        else:
            return

        self.blinking = self._blinking_ratio(landmarks, points)
        self._isolate(original_frame, landmarks, points)

        if not calibration.is_complete():
            calibration.evaluate(self.frame, side)

        threshold = calibration.threshold(side)
        self.pupil = Pupil(self.frame, threshold
```

Лістинг pupil.py:

```
import numpy as np
import cv2
```

```
class Pupil(object):
    """
    This class detects the iris of an eye and estimates
    the position of the pupil
    """

    def __init__(self, eye_frame, threshold):
        self.iris_frame = None
        self.threshold = threshold
        self.x = None
        self.y = None
        self.detect_iris(eye_frame)

    @staticmethod
    def image_processing(eye_frame, threshold):
        """Performs operations on the eye frame to isolate the iris

        Arguments:
            eye_frame (numpy.ndarray): Frame containing an eye and nothing else
            threshold (int): Threshold value used to binarize the eye frame

        Returns:
            A frame with a single element representing the iris
        """
        kernel = np.ones((3, 3), np.uint8)

        new_frame = cv2.bilateralFilter(eye_frame, 10, 15, 15)
        new_frame = cv2.erode(new_frame, kernel, iterations=3)
        new_frame = cv2.threshold(new_frame, threshold, 255, cv2.THRESH_BINARY)[1]

        return new_frame

    def detect_iris(self, eye_frame):
        """Detects the iris and estimates the position of the iris by
        calculating the centroid.

        Arguments:
            eye_frame (numpy.ndarray): Frame containing an eye and nothing else
        """
        self.iris_frame = self.image_processing(eye_frame, self.threshold)

        contours, _ = cv2.findContours(self.iris_frame, cv2.RETR_TREE,
cv2.CHAIN_APPROX_NONE)[-2:]
        contours = sorted(contours, key=cv2.contourArea)

        try:
            moments = cv2.moments(contours[-2])
            self.x = int(moments['m10'] / moments['m00'])
            self.y = int(moments['m01'] / moments['m00'])
        except (IndexError, ZeroDivisionError):
            pass
```

Додаток Б

Презентаційний матеріал

Мета та завдання

- **Обєкт**: процес проходження тестування знань за аналізом погляду у відеопотоці.
- **Предмет**: методи обробки зображень у відеопотоці, методи виявлення елементів обличчя на зображенні, методи проходження тестувань знань.
- **Метою роботи** є підвищення академічної доброчесності під час проходження тестувань за аналізом погляду шляхом обрахунку кількості часу, коли погляд знаходиться поза екраном.

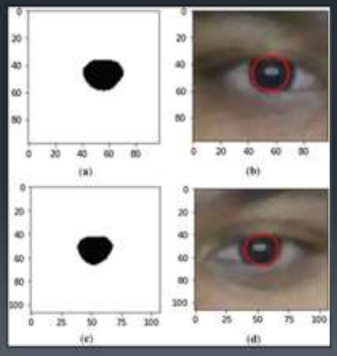
Завдання

- провести аналіз існуючих підходів які реалізують метод спостереження за очима під час тестування;
- запропонувати та реалізувати метод спостереження за очима;
- реалізувати аналіз відео потоку на стороні сервера під час проходження тестувань;
- реалізувати вебсистему тестування знань з вбудованим методом спостереження за очима.

3

Існуючі методи спостереження за поглядом



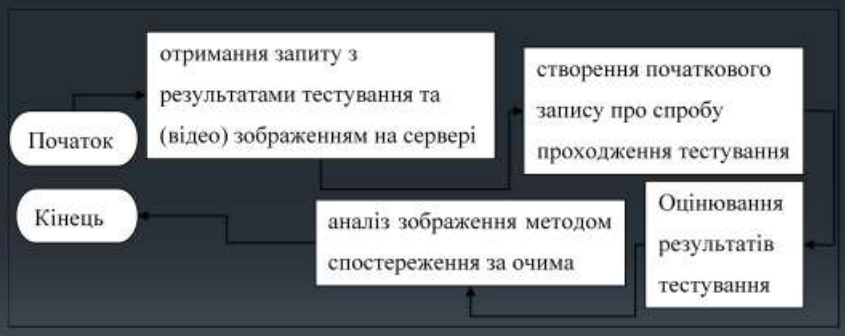

4


Метод спостереження за поглядом
Загальна схема методу спостереження за поглядом очей у відео потоці
Початок
(Блок 1) Запис відео зображення на клієнтській частині
(Блок 2) Обробка результатів тестування на стороні серверу
(Блок 3) Метод спостереження за очима
(Блок 4) Обробка результатів методу спостереження за очима
Кінець
5


Метод спостереження за поглядом
Схема роботи клієнтської частини вебзастосунку при тестуванні
Початок
завантаження сторінки тестування
початок запису (відео) зображення з камери клієнта
Кінець
відправка (відео) зображення та результатів тестування на сервер
проходження тестування користувачем
6

Метод спостереження за поглядом

Схема початкової обробки даних сервером



7

Метод спостереження за поглядом

Схема для виявлення напрямку погляду на відео зображенні

8

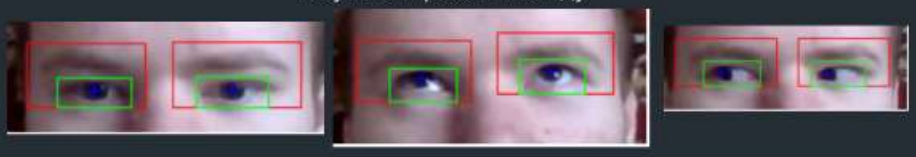
Метод спостереження за поглядом

Схема для отримання результату за аналізом погляду

9

Метод спостереження за поглядом

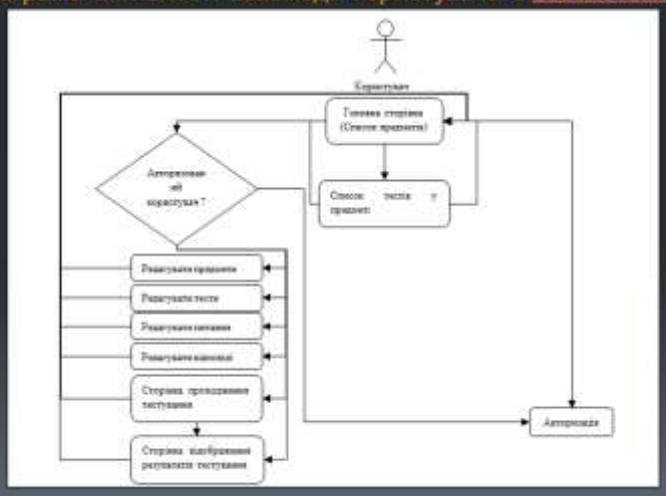
Результати роботи методу



10

Вебсистема тестування знань

Діаграма активності взаємодії користувача з вебсистемою



11

Вебсистема тестування знань

12

Висновки

- Розглянуто існуючі рішення для спостереження за очима
- Розроблено метод спостереження за очима для систем тестування знань
- Розроблена вебсистема з інтегрованим методом спостереження за очима

13

Дякую за увагу !

Wed Jun 12 10:39:28 EEST 2024, Петровський Сергій Степанович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 2.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 8%**

ID: 129882 Назва: КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА на тему Метод спостереження за очима (eye-tracking) для вебсистеми тестування знань Додано в БД: 2024-06-12 Автора: Богдан РОМАНОВ Керівники: Олександр БАРМАК Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	92195	1392	3967 (4%)	64 (5%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

Ім'я користувача:
Кафедра КН

ID перевірки:
1016351599

Дата перевірки:
12.06.2024 11:43:59 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
12.06.2024 11:47:55 EEST

ID користувача:
100005671

Назва документа: **КН-20-2 Романов_ЗАПИСКА**

Кількість сторінок: **70** Кількість слів: **14609** Кількість символів: **111845** Розмір файлу: **2.25 MB** ID файлу: **1016155269**

4.5%
Схожість

Найбільша схожість: **2.42%** з джерелом з бібліотеки (ID файлу: **1016149429**)

2.95% Джерела з Інтернету	462	Сторінка 72
3.09% Джерела з Бібліотеки	136	Сторінка 74

0% Цитат

Вилучення цитат вимкнено

Вилучення списку бібліографічних посилань вимкнено

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	3

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ КАФЕДРИ КОМП'ЮТЕРНИХ НАУК
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: *Метод спостереження за очима (eye-tracking) для вебсистеми тестування знань*

Автор: *студент групи КН-20-2 Романов Богдан Анатолійович*

Спеціальність: *122 – Комп'ютерні науки*

Освітня програма: *освітньо-професійна*

Науковий керівник: *д.т.н., проф. Олександр Бармак*

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	***відповідає***
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі Романова Богдана Анатолійовича, не є плагіатом, оскільки: запозичення розміщені в розділі огляду існуючих підходів, не описують безпосередньо авторську роботу і не стосуються її результатів; усі запозичення фрагментарні; до запозичень входять фрагменти що не мають авторства і містять поширені конструкції; серед запозичень знаходяться загальновідомі терміни та скорочення.

Обсяг запозичень, визначений системами виявлення збігів/ідентичності/схожості, складає:

- за системою Anti-Plagiarism – 2%;

- за системою Unicheck – 4.5 %.

Керівник роботи _____ *Олександр БАРМАК*

Гарант ОП _____ *Олександр МАЗУРЕЦЬ*

Завідувач кафедри КН _____ *Олександр БАРМАК*

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ

Кафедра комп'ютерних наук

РЕЦЕНЗІЯ

на кваліфікаційну роботу бакалавра

студента *гр. КН-20-2 Романова Богдана Анатолійовича*
за темою: *Метод спостереження за очима (eye-tracking) для вебсистеми тестування знань*

1. Актуальність обраної теми

Забезпечення академічної доброчесності при проведенні онлайн тестувань є актуальною задачею. Саме для на вирішення цієї задачі і спрямований, запропонований у роботі, метод спостереження за очима для вебсистеми тестування знань.

2. Повнота розкриття мети та завдань роботи

У кваліфікаційні роботі досягнута поставлена мета та повністтю виконанні завдання дослідження: досліджено існуючі рішення для спостереження за очима, розроблено метод спостереження за очима для систем тестування знань та проведено інтеграцію модуля спостереження за очима у розроблену вебсистему тестування знань.

3. Зміст кожного розділу роботи

У першому розділі проаналізовані існуючі рішення для систем спостереження за очима, визначена мета та завдання дослідження. У другому розділі розроблено метод спостереження за очима, у третьому розділі наведено практичну реалізацію методу спостереження за очима та інтеграцію його у вебсистему тестування знань.

4. Оцінка розробленої інформаційної системи, її практична цінність

Розроблена вебсистема успішно забезпечує функціонал оцінки часу, де погляд перебував поза екраном під час проходження тестувань. Метод спостереження за очима може бути інтегрований в різні вебресурси та використаний закладами освіти для підвищення рівня академічної доброчесності під час проходження онлайн тестувань.

5. Якість оформлення кваліфікаційної роботи бакалавра

Оформлення кваліфікаційної роботи бакалавра відповідає основним вимогам до оформлення наукових робіт.

6. Недоліки кваліфікаційної роботи бакалавра

Розроблена інформаційна система може бути покращена шляхом зменшення необхідного часу для аналізу погляду на зображенні та внесенням змін в дизайн інтерфейсу.

7. Загальний висновок (допускається чи не допускається до захисту), та оцінка на яку заслуговує кваліфікаційна робота.

Враховуючи рівень виконання та забезпечення усіх необхідних вимог, робота може бути допущена до захисту. Рекомендована оцінка ***«відмінно»***.

Рецензент _____ /Бездітний І.П./

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ

Кафедра комп'ютерних наук

ВІДГУК НАУКОВОГО КЕРІВНИКА

на кваліфікаційну роботу бакалавра

студента *гр. КН-20-2 Романова Богдана Анатолійовича*

за темою *Метод спостереження за очима (eye-tracking) для вебсистеми тестування знань*

1. Актуальність теми

Визначення рівня академічної доброчесності під час проходження онлайн тестувань за спостереженнями за очима є актуальною задачею. Сучасне використання автоматизованих систем тестування знань потребує функціоналу, який би дозволяв формувати висновки про академічну доброчесність шляхом аналізу напрямку погляду упродовж проходження тестування, порівнюючи проміжок часу де погляд був поза екраном з загальним часом затраченим на проходження тестування. Розробка такого метода є актуальною задачею комп'ютерних наук.

2. Відповідність роботи предметній області Стандарту спеціальності 122 Комп'ютерні науки

За стандартом, а саме описом предметної області, об'єктами вивчення та діяльності є математичні, інформаційні, імітаційні моделі реальних явищ, об'єктів, систем і процесів та методи і технології отримання, зберігання, обробки, передачі та використання інформації. Метою роботи саме є розробка методу підвищення академічної доброчесності під час проходження тестувань за аналізом погляду шляхом обрахунку кількості часу, коли погляд знаходиться поза екраном та розробка інтеграції у системі тестування знань зазначеного методу. При вирішенні поставленої задачі використано математичні моделі, методи та алгоритми розв'язання теоретичних і прикладних задач, що виникають при розробці вказаного методу. Тому результати виконання кваліфікаційної роботи бакалавра відповідають стандарту бакалавра спеціальності 122 – Комп'ютерні науки.

3. Професійні та особистісні якості бакалавра

При роботі над кваліфікаційною роботою бакалавра Романов Богдан Анатолійович проявив себе кваліфікованим фахівцем та дисциплінованим студентом, вчасно виконуючи завдання дослідження. Як в процесі написання пояснювальної записки, так і при розробці прикладного програмного забезпечення проявив достатні для

одержання успішного результату компетентності та результати навчання. Опанував професійні скіли за напрямком «Комп'ютерні науки» та достатньо значний софт скіл .

4. Ступінь самостійності під час виконання кваліфікаційної роботи

Одержані в роботі результати є наслідком особистої діяльності студента, який самостійно виконував всі поставлені задачі.

5. Ступінь оволодіння методами дослідження

При реалізації кваліфікаційної роботи показав достатній рівень компетентностей та володіння необхідними інструментами та обладнанням, методами, методиками та технологіями предметної області комп'ютерних наук.

6. Повнота та якість розкриття теми роботи

Тема роботи в повній мірі обґрунтована й розкрита, проведено аналіз відомих досліджень в межах обраної теми, поставлені завдання, розроблено відповідний метод та програмне забезпечення для валідації та верифікації запропонованого метода.

7. Логічність, послідовність, аргументованість, літературна грамотність викладення матеріалу

Структура роботи та послідовність викладення логічні та відповідають поставленій меті. Викладення матеріалу послідовне, аргументоване, літературно грамотне.

8. Можливість практичного застосування кваліфікаційної роботи бакалавра, окремих її частин

Розроблений у роботі метод та його програмна реалізація може бути використана закладами освіти для підвищення академічної доброчесності при проходженні онлайн тестувань.

9. Висновок про можливість допуску кваліфікаційної роботи бакалавра до захисту, на яку оцінку заслуговує робота

*Враховуючи високий рівень виконання та забезпечення усіх необхідних вимог, робота може бути допущена до захисту. Рекомендована оцінка «**відмінно**».*

Керівник _____ *д.т.н., проф. зав. каф. КН Олександр БАРМАК*