Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Моделі та програмні засоби дистанційного тестування з використанням технології Google Cloud Vision. Частина 2. Модуль клієнта»

Виконав: студент II курсу
групи ЗПІ-22м спеціальності
121 – Інженерія програмного забезпечення
Кирнасюк Є.С.

Керівник: к.т.н., доц. каф. ПЗ Майданюк В. П.
«11» грудня 2023 р.

Опонент: к.т.н., доц. каф. КІ Черняк О. І.
«11» грудня 2023 р.

Допущено до захисту
Завідувач кафедри ПЗ
д.т.н., проф. Романюк О. Н.
(прізвище та ініціали)
«11» грудня 2023 р.

ВНТУ – 2023

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

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
«19» вересня 2023 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ
СТУДЕНТУ
Кирнасюку Євгену Сергійовичу

1. Тема роботи – Моделі та програмні засоби дистанційного тестування з використанням технології Google Cloud Vision. Частина 2. Модуль клієнта.

Керівник роботи: Майданюк Володимир Павлович, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від «18» вересня 2023 року № 247.

2. Строк подання студентом роботи 5 грудня 2023 р

3. Вихідні дані до роботи: модель розробки – клієнтський модуль для здійснення тестування з фотоконтролем; Вхідні дані: середовище розробки WebStorm; каскадна модель розробки; мова програмування JavaScript/TypeScript; фреймворк Angular.

4. Зміст текстової частини: вступ; аналіз та обґрунтування вибору методу розробки та постановка задачі дослідження; розробка структури та алгоритмів роботи програмного продукту; розробка програмних компонент для додат тестування програми; висновки; список використаних джерел; додат економічна частина.

5. Перелік графічного матеріалу: титульний слайд; актуальність теми; мета, об'єкт та предмет дослідження; задачі дослідження; наукова новизна; практична цінність одержаних результатів; порівняльний аналіз аналогів; блок-схема алгоритму проходження тесту з фотоконтролем; блок-схема алгоритму роботи програмного додатку; структура графічного інтерфейсу головного вікна додатку; структура графічного інтерфейсу проходження тесту; структура графічного інтерфейсу створення тесту; структура графічного інтерфейсу вкладки логіну та реєстрації; тестування програми; апробація та публікації результатів роботи; фінальний слайд.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	виконанн прийняв
1-4	Майданюк В. П. к. т. н., доцент кафедри ПЗ	19.09.23	05.12.23
5	Причепа І.В., к.е.н., доцент кафедри ЕПВМ	27.11. 23	1.12.23

7. Дата видачі завдання 19 вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітк
1	Аналіз та обґрунтування вибору методу розробки та постановки задачі дослідження	20.09.23 – 25.09.23	вик
2	Розробка архітектури та алгоритмів програмного продукту	26.09.23 – 28.09.23	вик
3	Розробка програмного продукту	29.09. 23 – 7.10.23	вик
4	Тестування програми	8.10. 23 – 24.11.23	вик
5	Економічна частина	27.11. 23 – 1.12.23	вик
6	Оформлення пояснювальної записки та матеріалів до захисту	25.11. 23 – 1.12.23	вик

Студент _____ (підпис) **Кирнасюк Є.С.** (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ (підпис) **Майданюк В. П.** (прізвище та ініціали)

АНОТАЦІЯ

УДК. 004.912.032.26

Кирнасюк Є. С. Моделі та програмні засоби дистанційного тестування з використанням технології Google Cloud Vision. Частина 2. Модуль клієнта. Магістерська кваліфікаційна робота зі спеціальності 121 – інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2023.

На укр. мові. Бібліогр. : 31 назви ; рис. : 39; табл. 6.

У магістерській кваліфікаційній роботі проведено детальний аналіз стану програмного забезпечення для проведення тестування. Сформульовано мету досліджень – підвищення ефективності процесу тестування людей, забезпечення чесності результату тесту за допомогою технології Google Cloud Vision та фотоконтролю процесу тестування.

Запропоновано процес здійснення онлайн тестування користувача, на основі різнотипових тектово-візуальних тестових питань й фотоконтролем процесу тестування користувача у формі прикладного програмного інтерфейсу клієнтського модуля.

У магістерській кваліфікаційній роботі розроблено програмний засіб дистанційного тестування з використанням технології Google Cloud Vision. Модуль клієнта. Розроблений програмний засіб призначений для покращення процесу створення, перевірки достовірності та проходження онлайн тестів.

Даний програмний засіб написаний на мові JavaScript/TypeScript з використанням фреймворку Angular, та характеризується простотою розробки, швидкодією та адаптивністю.

Розроблено алгоритми для реалізації процесу онлайн тестування з фотоконтролем та технології Google Cloud Vision.

Ключові слова: Веб додаток, дистанційне навчання, онлайн тестування, фотоконтроль, Angular, JavaScript/TypeScript, Google Cloud Vision.

ABSTRACT

Kirnasyuk E. S. Models and software tools of remote testing using Google Cloud Vision technology. Part 2. The client module. Master's qualification work on specialty 121 - software engineering, educational program - software engineering. Vinnytsia: VNTU, 2023.

In Ukrainian speech Bibliographer: 31 titles; Fig. : 39; table 6.

In the master's qualification work, a detailed analysis of the state of the software for testing was carried out. The purpose of the research is formulated - to increase the efficiency of the process of testing people, to ensure the integrity of the test result using Google Cloud Vision technology and photo control of the testing process.

The process of online user testing is proposed, based on various types of text-visual test questions and photo control of the user testing process in the form of an application software interface of the client module.

The master's thesis developed a software tool for remote testing using Google Cloud Vision technology. Client module. The developed software tool is designed to improve the process of creating, validating and passing online tests.

This software tool is written in the JavaScript/TypeScript language using the Angular framework, and is characterized by ease of development, speed and adaptability.

Algorithms have been developed for the implementation of the online testing process with photo control and Google Cloud Vision technology.

Keywords: Web application, distance learning, online testing, photo control, Angular, JavaScript/TypeScript, Google Cloud Vision.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. У наш час майже будь-яка область з якою взаємодіє людина пов’язана з роботою на комп’ютері, в телефоні чи на іншій платформі. Без програмування сьогодні уявити сучасний світ неможливо, людина приймає участь в програмуванні найрізноманітніших речей, починаючи від ступенів космічних ракет для їх подальшого повернення на Землю до програмування розумних інструментів для вашої вбиральні [1].

Розробка веб застосунків на сьогодні є однією з найпоширеніших сфер у розробці програмного забезпечення. Веб застосунок (іноді веб додаток) – розподілений застосунок, в якому клієнтом виступає браузер, а сервером – веб сервер. Браузер може бути реалізацією так званих тонких клієнтів – логіка застосунку зосереджується на сервері, а функція браузера полягає переважно у зображенні інформації, завантаженої мережею з сервера, і передачі назад даних користувача. Однією з переваг такого підходу є той факт, що клієнти не залежать від конкретної операційної системи користувача, тому веб застосунки є міжплатформовими сервісами [2].

Технології віддаленого доступу знайшли своє використання і в дистанційному навчанні: і лекції можна читати онлайн і практики проводити і навіть здавати іспити та тести. Однак попри неймовірну зручність для учнів, при цьому виникають серйозна загроза шахраювання з їх боку, адже контроль викладача значно послаблений і часто він не може запобігти порушенням академічної доброчесності.

Розробка програмного забезпечення для поліпшення процесу онлайн тестування вимагає врахування кількох ключових аспектів. Одним із найважливіших є забезпечення безпеки та недопущення шахрайства. Методи аутентифікації та валідації можуть бути використані для перевірки особи, яка здає тест, та визначення чесності результатів.

Крім того, створення ефективного інтерфейсу користувача для тестування важливо забезпечити зручність та легкість використання. Це може включати в себе можливість вставки графіки, використання різних типів питань (текстових, мультимедійних, тощо) та інші функціональності, які полегшують процес тестування.

Для підвищення довіри до результатів тестування, важливо використовувати алгоритми оцінювання, які є об'єктивними і справедливими. Також можуть бути використані системи моніторингу та аналізу активності користувача під час тестування для виявлення підозрілих взаємодій чи шахрайської діяльності.

Розвиток такого програмного забезпечення також може включати інтеграцію з системами управління навчанням, щоб забезпечити зв'язок з іншими аспектами навчального процесу.

Узагальнюючи, створення програм для поліпшення онлайн тестування вимагає уважного підходу до безпеки, ефективного взаємодії з користувачем та об'єктивного оцінювання результатів, що сприятиме підвищенню довіри до онлайн тестів у навчальних закладах та компаніях.

Також, важливою частиною розробки програмного забезпечення для онлайн тестування є врахування потреб різних типів користувачів. Забезпечення адаптивності інтерфейсу для різних пристроїв та різних рівнів технічної грамотності може сприяти більш широкому використанню системи.

Додатково, важливо враховувати прозорість та доступність у процесі тестування. Забезпечення чіткої інформації щодо правил та умов тесту, а також надання допомоги чи пояснень у разі необхідності, може покращити досвід користувача.

У контексті дистанційного навчання, важливо також розглядати можливості для взаємодії між учнями та викладачами, навіть під час тестування. Це може включати функції обговорення завдань, спільне розв'язання проблем та інші способи підтримки спільноти.

Це і визначає актуальність задачі розробки програмного засобу призначеного для покращення процесу створення, перевірки достовірності та проходження онлайн тестів, які зможуть підвищити довіру до результатів тестування.

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

Мета та завдання дослідження. Метою роботи є підвищення якості дистанційного тестування та зменшення випадків порушення академічної доброчесності за рахунок розробки клієнтської частини тестувальної онлайн системи з фотоконтролем.

Основними задачами дослідження є:

– спроектувати архітектуру клієнтського модуля;

– здійснити варіантний аналіз і обґрунтування вибору засобів для реалізації програмного застосунку;

– розробити інтерфейс користувача клієнтської частини веб додатку; (інтерфейсу користувача);

– розробити клієнтський модуль для створення тесту;

– розробити клієнтський модуль для перегляду усіх доступних тестів;

– розробити клієнтський модуль для проходження тесту з фотоконтролем;

– розробити клієнтський модуль для взаємодії з сервером;

– розробити клієнтський модуль для надсилання результатів фото контролю серверу для його аналізу за допомогою Google Cloud Vision;

- розрахунок економічної доцільності розробки.

Об'єктом дослідження – процес здійснення дистанційного тестування з фото контролем.

Предмет дослідження – методи та програмні засоби розробки клієнтської частини тестувальної системи з фото контролем.

Методи дослідження. У процесі дослідження використовувались: теорія кодування зображень для стиснення результатів фотоконотролю, теорія алгоритмів і структур даних для розробки програмного забезпечення, методи побудови та взаємодії компонентів клієнт-серверних веб-систем та методи тестування для перевірки роботи створеного веб додатку; комп'ютерне моделювання для аналізу та перевірки отриманих теоретичних положень.

Наукова новизна отриманих результатів.

1. Подальшого розвитку отримав метод дистанційного тестування, у якому на відміну від існуючих, додано функції отримання та передачі на сервер результатів фотоконтролю, що дозволило підвищити достовірність тестування на 10 %.
2. Подальшого розвитку отримав метод побудови архітектури систем дистанційного тестування, у якому, на відміну від існуючих, використано архітектуру типу Rich client для тестування, що зменшує навантаження на сервер та економить трафік.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає в тому, що на основі отриманих у магістерській кваліфікаційній роботі теоретичних положень запропоновано алгоритми та розроблено клієнтський модуль тестувальної системи з фото контролем.

Особистий внесок здобувача. Усі наукові результати, викладені у магістерській кваліфікаційній роботі, отримані автором особисто. У друкованих працях, опублікованих у співавторстві, автору належать такі результати: розробка алгоритму проходження тесту з фотофіксацією [3]**.**

Апробація результатів роботи. Основні положення магістерської кваліфікаційної роботи доповідалися та обговорювалися на міжнародних та всеукраїнських конференціях: Міжнародна науково-практична Інтернет-конференції «Електронні інформаційні ресурси: створення, використання, доступ» (Вінниця, 2023) [4].

Публікації. Основні результати досліджень опубліковано в 1 науковій праці у матеріалах конференції.

1 АНАЛІЗ ТА ОБҐРУНТУВАННЯ ВИБОРУ МЕТОДУ РОЗРОБКИ ТА ПОСТАНОВКИ ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Аналіз стану проблеми

Протягом останніх років дистанційне навчання стало не лише обговорюваною темою, але й широко впровадженою практикою в освіті світу. Згадка про цей термін десять років тому викликала асоціації з повністю віддаленим навчанням, де очні зустрічі зі студентами вважалися рідкісними, що ускладнювало перевірку їхніх знань та автентичності. З того часу пройшло багато часу на налаштування дистанційних курсів та тестів [5].

В сучасний період багато університетів, коледжів та шкіл по всьому світу активно використовують веб та мобільні додатки для представлення інформації про свої освітні програми. Більшість з них спрямовані на надання інформації абітурієнтам, хоча лише деякі забезпечують можливість перевірки рівня знань та запобігання шахрайству під час тестів.

Введення веб-додатків, спеціально розроблених для освітніх установ, значно спростило б процес тестування студентів, дозволяючи не лише оцінювати їхні досягнення, але й переглядати статистику тестів, результати студентів та надавати фотодокументацію після завершення тестів. Це сприяло б більш об'єктивній та справедливій оцінці знань кожного студента.

Більше того, використання спеціалізованих веб-додатків для освітніх установ дозволило б значно розширити можливості тестування студентів. Зокрема, забезпечення функції перегляду статистики тесту дозволило б вчителям аналізувати ефективність певних питань та модулів, щоб оптимізувати навчальний процес.

Також, враховуючи важливість взаємодії та залучення студентів, веб-додаток міг би надати можливість обговорення результатів тестування та консультацій між вчителями та учнями. Це стимулювало б активну

комунікацію та взаємопорозуміння, що є важливими елементами успішного освітнього процесу.

Зазначеною ініціативою також можна було б сприяти розвитку інноваційних методів оцінювання, таких як використання інтерактивних завдань, фото-аналізу та інших технологічних інструментів, що допомагають зробити оцінювання більш цікавим та ефективним. Такий веб-додаток став би не лише інструментом тестування, але й платформою для вдосконалення освітнього процесу в цілому.

Впровадження веб-додатків для освітніх установ може також сприяти забезпеченню доступності навчального матеріалу та ресурсів для студентів з різних географічних регіонів. Це особливо актуально в контексті росту популярності дистанційного навчання, коли студенти можуть знаходитися в різних частинах світу.

Однією з переваг веб-додатків є їхня можливість персоналізації навчання. Такі додатки можуть адаптуватися до потреб конкретного студента, надаючи індивідуальні завдання та матеріали для оптимального засвоєння матеріалу. Це може покращити якість освіти та зробити її більш ефективною для кожного учня.

Крім того, важливим аспектом використання веб-додатків є забезпечення безпеки тестування та захисту від шахрайства. Технології аутентифікації та відслідковування активності можуть бути впроваджені для запобігання несанкціонованому доступу та недобросовісному поводженню під час тестування.

Необхідно також враховувати важливість інтеграції цих веб-додатків з існуючими платформами для управління освітнім процесом, щоб забезпечити зручний доступ до інформації для вчителів, адміністраторів та студентів.

В контексті обговорення ключових параметрів та функціональних можливостей веб-додатків для освітніх установ, особлива увага буде приділена аспектам, пов'язаним із забезпеченням безпеки та конфіденційності даних студентів. Важливо розглянути системи аутентифікації, шифрування даних та

заходи забезпечення приватності, щоб уникнути можливих порушень та зловживань.

Під час порівняльного аналізу аналогів важливо враховувати не лише технічні характеристики, але й відгуки користувачів, рівень підтримки та можливості майбутнього розширення. Це допоможе визначити повний потенціал інструменту та його придатність для конкретної освітньої програми чи закладу.

Отже, використання веб-додатків для тестування та оцінювання може значно підвищити ефективність та якість освітнього процесу, забезпечуючи сучасні інструменти для навчання та взаємодії між учасниками освітнього процесу.

1.2 Порівняльний аналіз аналогів

У цьому розділі будуть розглядатися ключові аспекти та функціональні можливості веб-додатків, спрямованих на використання в освітніх установах. Особлива увага буде приділена їхнім можливостям у забезпеченні автентичності та об'єктивності процесу оцінювання знань студентів. Важливим аспектом є також аналіз можливостей взаємодії між викладачами та студентами, що сприяє активній комунікації та розвитку новаторських методів оцінювання.

В рамках порівняльного аналізу подібних продуктів буде визначено найбільш ефективні рішення та виявлені можливості для вдосконалення дистанційного тестування в освітньому середовищі. Такий підхід допоможе не лише вибрати оптимальні інструменти для конкретного навчального закладу, але й розробити стратегії для подальшого розвитку та інновацій в галузі дистанційної освіти.

Додатки які схожі за функціоналом, що можуть виконувати потреби освітніх установ наступні: JetIQ, Google forms та Online Test Pad.

JetIQ – це внутрішня навчальна платформа Вінницького національного технічного університету, яка була створена для взаємодії між викладачів та студентів, де викладачі викладають матеріали та тести до них, а студенти проходячи їх надають рівень своїх знань [6]. Основним недоліком даної платформи є те що це внутрішня система для викладачів та студентів Вінницького національного технічного університету, тобто викладачі та студенти інших навчальних закладів не мають змоги використовувати її. Хоча при проходженні тести є невеликі засоби, які дозволяють зробити результат проходження тесту, максимально чесним, блокуючи студенту перехід між вкладками браузера, сповіщаючи його повідомленням та не зарахуванням відповіді (рисунок 1.1).

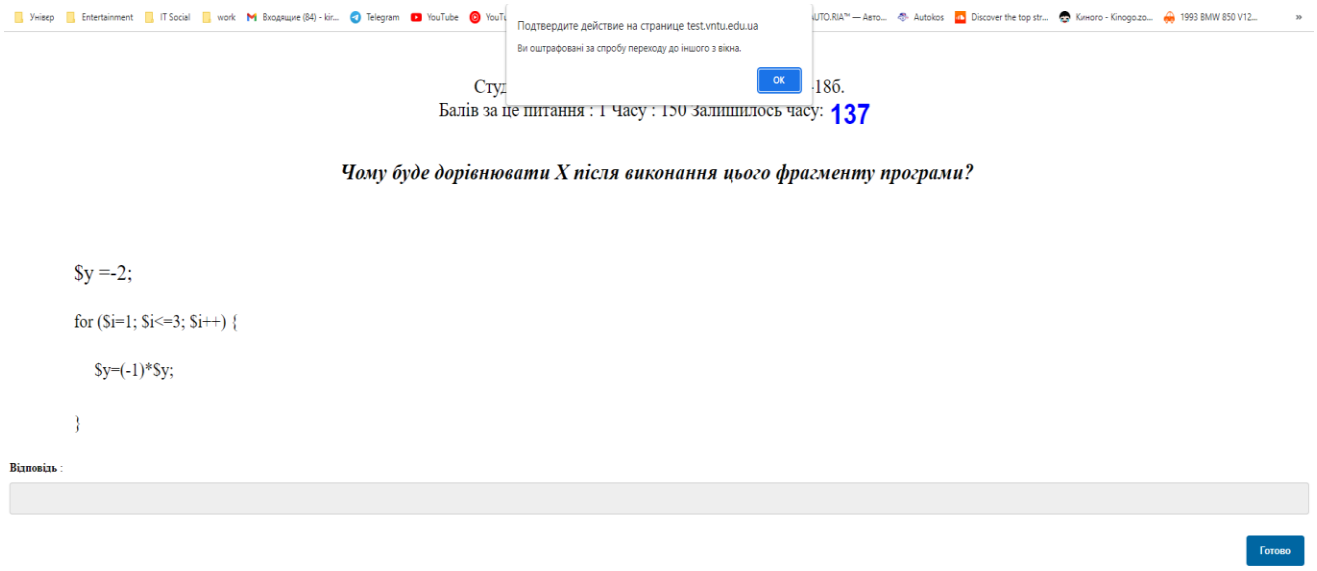


Рисунок 1.1 – Процес проходження тесту та виявлення шахрайства

Google Forms – веб додаток для створення та проходження опитування та тестів, який було створено Google та входить до складу безкоштовного вебпакету Google Docs Editors [7]. Даний веб додаток є загальнодоступним на відмінну від JetIQ, має велику кількість налаштувань та видів тестів (рисунок

1.2). Хоча на відмінну від JetIQ, Google Forms не мають функціоналу для виявлення шахрайства під час проходження тесту. Також Google Forms не має можливості задання часових рамок проходження тесту та конкретного питання, на відмінну від JetIQ та Online Test Pad.

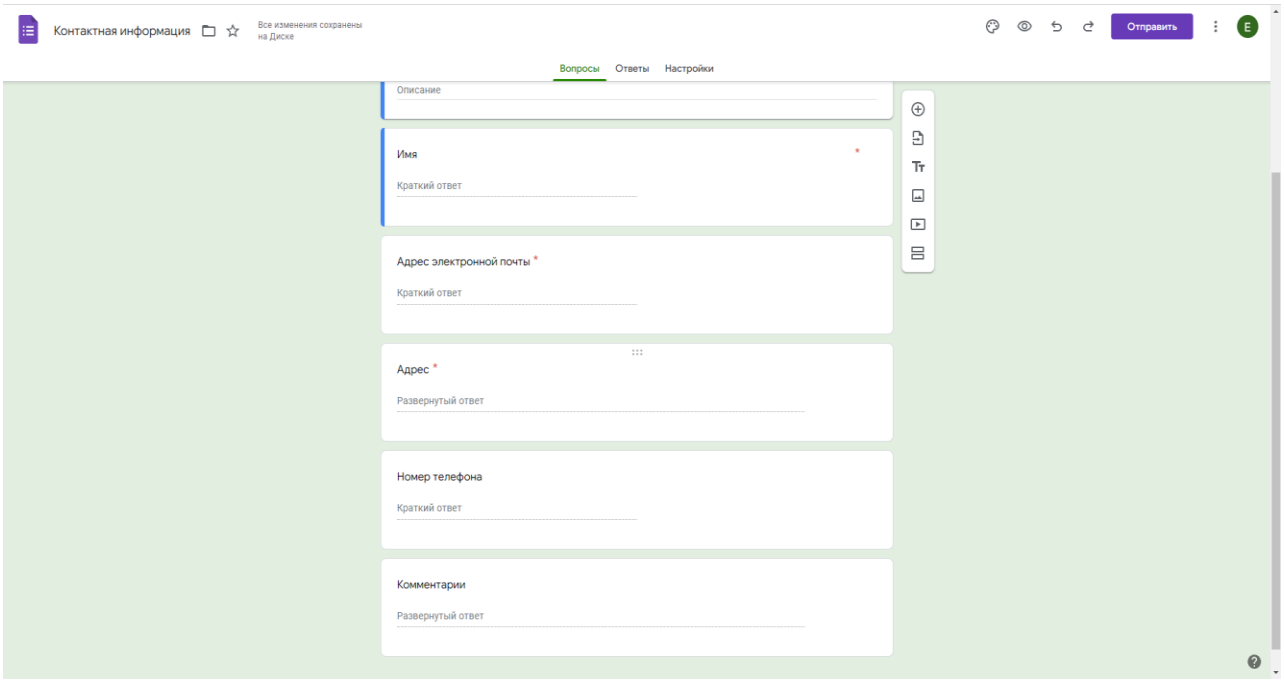


Рисунок 1.2 – Процес створення тесту Google Forms

Online Test Pad – простий та зручний сервіс для створення тестів та проведення тестування [8]. У конструкторі тестів передбачено велику кількість різних налаштувань тестів. У якому можна швидко та зручно створити дійсно унікальний тест під ваші цілі та завдання. Також доступний перегляд кожного результату, статистики відповідей та набраних балів з кожного питання, статистики з кожного результату. У таблиці представлені всі результати, реєстраційні параметри, відповіді на всі питання, які ви можете зберегти в Excel. Але недолік даного веб застосунку, як і в JetIQ та Google Forms, є відсутність фото чи відео контролю над проходженням тесту, але як і JetIQ він має можливість задавати час на проходження тесту та конкретного питання. Також дана платформа надає можливість проходження тестів не тільки у навчальних цілях, а і в розважальних (рисунок 1.3).

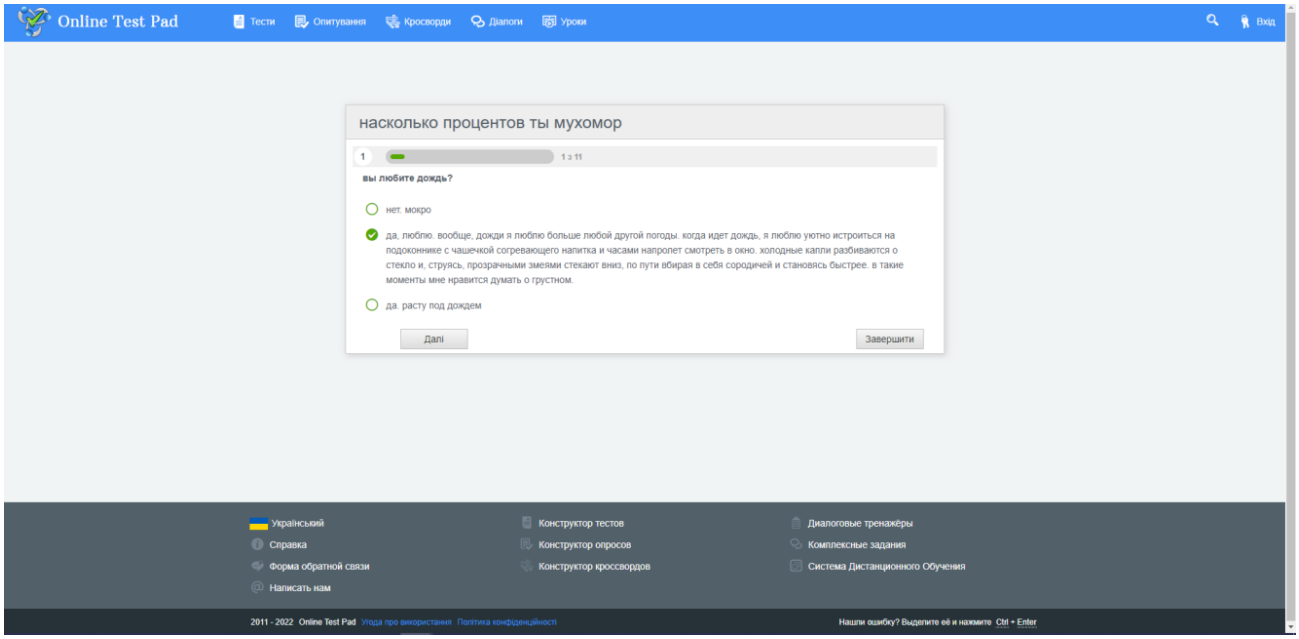


Рисунок 1.3 – Процес проходження розважального тесту на Online Test Pad

Отже, розглянувши усі аналоги визначено їх основні переваги та їх недоліки, на основі розглянутих додатків визначено основні функції для порівняння та порівняно їх з функціональними можливостями додатку, що буде розроблятися, та матиме назву – TestBuilder, , таблицю представлено у таблиці 1.1

Таблиця 1.1 – Порівняльна таблиця аналогів та розроблюваного додатку.

	JetIQ	Online Test Pad	Google Forms	TestBuilder
Загальнодоступний	–	+	+	+
Створення розважальних тестів	–	+	–	–
Фото контроль проходження тесту	–	–	–	+
Аналіз фото контролю	–	–	–	+

Продовження таблиці 1.1

	JetIQ	Online Test Pad	Google Forms	TestBuilder
Створення власних тестів	–	+	+	+
Статистика по проходження тесту	+	+	+	+
Часовий контроль проходження окремого питання	–	+	–	+
Орієнтованість на навчальні заклади	+	–	–	+
Результат:	2	5	3	7

Отже, проаналізувавши таблицю 1.1 визначено, що розроблюваний веб додаток, в межах потребах освітнього процесу є найбільш прийнятним для використання, головною перевагою є орієнтованість на навчальні заклади. Також розроблюваний додаток, візьме найкраще від представлених аналогів.

1.3 Вибір моделі життєвого циклу для розробки додатку

При розробці будь-якого програмного додатку розробник чи команда розробників обирає модель за якою буде розроблятися додаток. Такі моделі включать в себе процеси розробки, експлуатації та супровід програмного додатку, а також починаючи від визначення вимог до припинення використання програмного додатку.

Вибір моделі розробки є ключовим етапом перед створенням будь-якого програмного додатку і може значно вплинути на успішність проекту. Ось кілька причин, чому важливо ретельно обирати модель розробки:

– Функціональність. Різні моделі розробки можуть краще підходити для різних видів додатків. Наприклад, для великих та складних систем може бути ефективною модель розробки, орієнтована на водопад. Для проектів, які потребують гнучкості та постійних змін, краще підходить Agile.

– Методологія розробки. Різні моделі розробки передбачають різний спосіб розподілу завдань та взаємодії в команді. Важливо вибрати модель, яка найкраще враховує потреби вашого колективу та проекту.

– Тестування: Деякі методології, наприклад, Extreme Programming (XP) часто підкреслюють важливість тестування та автоматизації тестів. Інші можуть мати більший акцент на тестуванні на пізніших етапах розробки. Вибір моделі повинен враховувати потреби у якості програмного забезпечення.

– Керування ризиками. Деякі методології враховують інтегровані підходи до керування ризиками та забезпечення успішності проекту. Знання інструментів та прийомів в цьому контексті може допомогти забезпечити успішне виконання завдань та досягнення мети проекту.

– Швидкість розробки. Деякі моделі розробки спрямовані на швидкість виконання проекту, тоді як інші більше фокусуються на його довгостроковий успіх. Вибір повинен відповідати бізнес-вимогам і обмеженням термінів виконання.

– Взаємодія з клієнтом. Деякі методології активно включають клієнта в різні етапи розробки (наприклад, Scrum), що може бути важливим для проектів з великою кількістю невизначеності та змін.

Враховуючи ці аспекти, розробники можуть обрати модель розробки, яка найкраще відповідає конкретним умовам та завданням їхнього проекту, що в кінці кінців сприяє успіхові та якості програмного забезпечення.

Найбільш розповсюдженими моделями розробки на сьогодні є каскадна та спіральна моделі.

Каскадна модель представляє собою поетапне виконання кожного аспекту розробки без її повторного виконання у майбутньому. Тобто етап розробки виконується один раз і тільки один раз, це означає що кожен етап повинен бути ретельно виконаний та перевірений декілька раз перед початком виконання наступного етапу [9]. На рисунку 1.4 представлено приклад каскадної моделі життєвого циклу

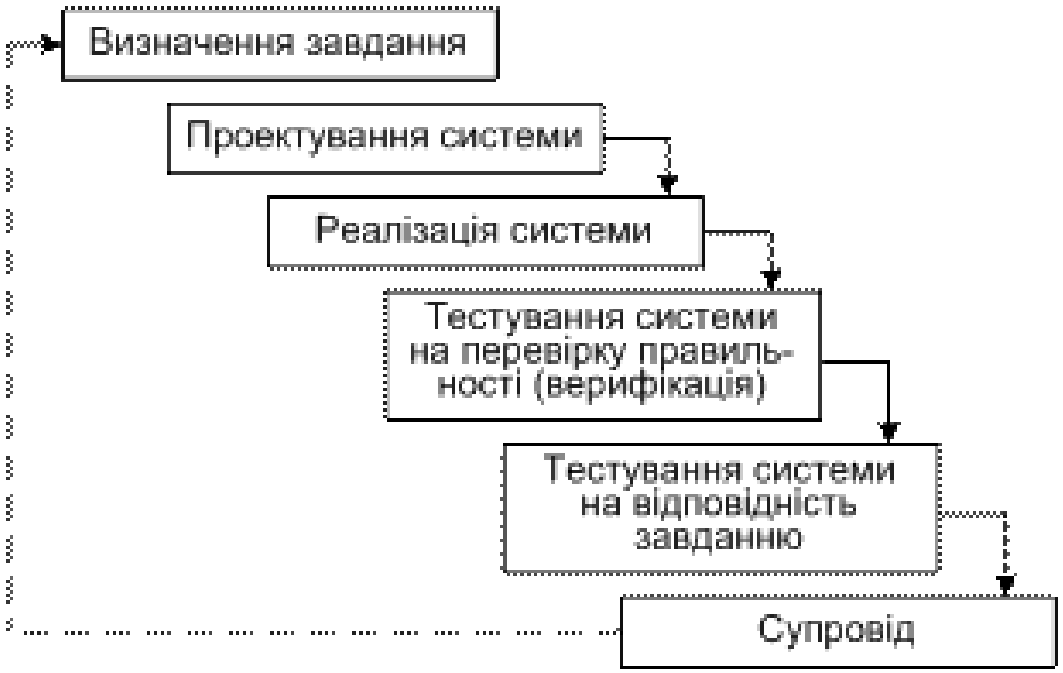


Рисунок 1.4 – Приклад каскадної моделі життєвого циклу програмного додатку

Спіральна модель представляє має протилежну від каскадної модель суть. При розробці програмного додатку з використанням спіральної моделі передбачається не повне виконання того чи іншого етапу розробки та можливості переходу до наступної. Спіральна модель все частіше використовується в компаніях ціль яких утримати нового клієнта, адже за таким життєвим циклом є можливість розробити основні модулю та продемонструвати їх замовникові [10].

Спіральна модель розробки передбачає можливість багаторазового повернення до попередніх етапів, внесенню змін до технічного завдання в

процесі розробки і так далі. Такий життєвий цикл є зручним для керівників компаній але є доволі неприємним для розробників, оскільки дуже часто розробникам доводиться повністю переписувати попередній код.

На рисунку 1.5 представлено спіральну модель життєвого циклу

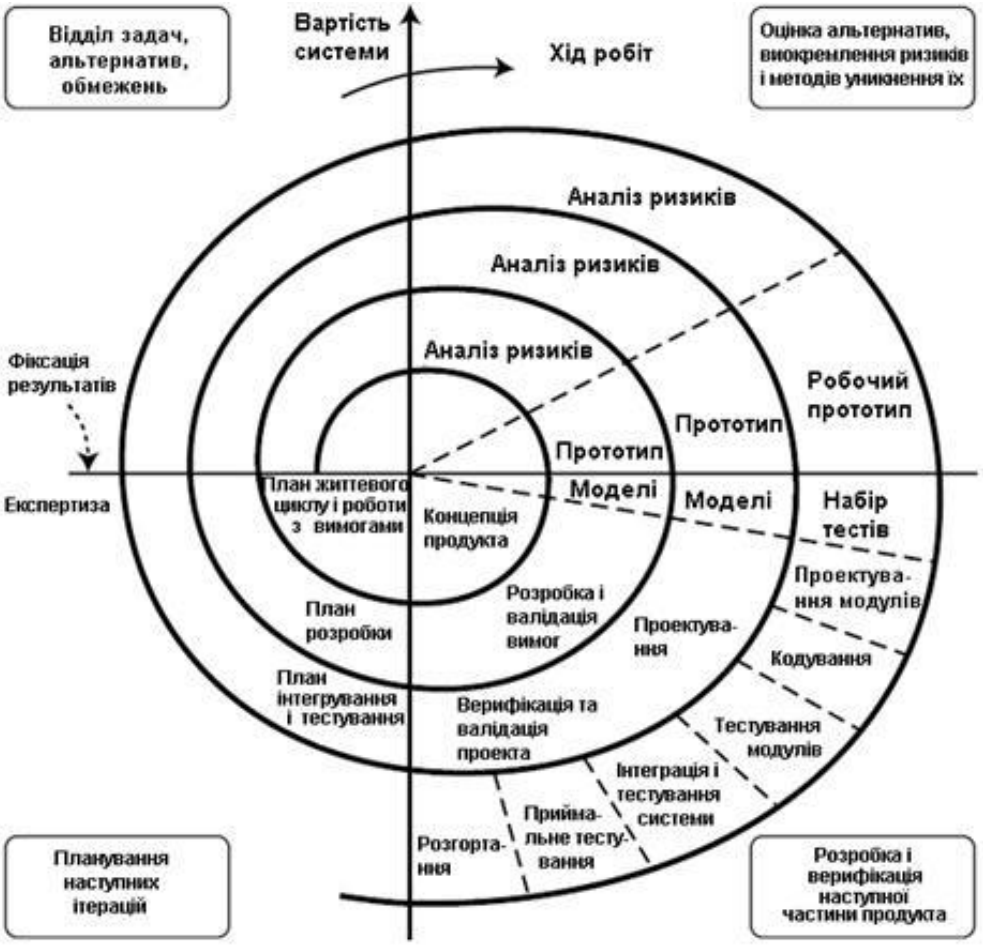


Рисунок 1.5 – Приклад спіральної моделі життєвого циклу програмного додатку

Отже, серед розглянутих вище моделей життєвого циклу найбільш прийнятним буде каскадна модель. Оскільки маючи чітке технічне завдання та невеликий за розмірами проект каскадна модель буде швидшою у виконанні ніж спіральна модель. Таким чином кожен етап розробки буде поетапним, повернення до попередніх етапів не передбачено.

1.4 Постановка задачі

Веб-додаток TestBuilder розроблено з урахуванням потреб сучасного освітнього середовища, надаючи викладачам універсальний інструмент для ефективного тестування та аналізу знань студентів. Однією з ключових функціональних можливостей є можливість створення та проходження тестів із фотофіксацією, що дозволяє забезпечити більш об'єктивну оцінку та автентичність результатів. Також перегляд фото звітів проходження тестів та їх аналіз Google Cloud Vision.

Веб застосунок TestBuilder не є системою лише для певного навчального закладу, а є універсальним інструментом для будь якого викладача, який хоче зробити зріз справжніх знань своїх студентів.

Додаток використовує потужні можливості Google Cloud Vision для аналізу фотографій, забезпечуючи високий рівень точності та достовірності. Зберігання даних про проходження тестів, перегляд історії та статистики тесту дозволяє вчителям ефективно відстежувати прогрес студентів та адаптувати навчальні матеріали відповідно.

Зазначено, що TestBuilder не обмежується конкретним навчальним закладом, але є універсальним інструментом для будь-якого викладача. Реалізація двох рівнів доступу — для користувачів, які проходять тести, та для власників тестів – надає зручність та гнучкість в управлінні процесом навчання.

Цей веб-додаток не лише полегшує процес тестування, але і відкриває нові можливості для викладачів у вигляді аналізу результатів, взаємодії та постійного вдосконалення навчального процесу.

Має бути реалізовано два рівні доступу:

– Перший рівень – користувач який хоче пройти тест. Він має доступ до тесту, його проходження та перегляду своїх результатів.

– Другий рівень – власник тесту. Даний користувач має доступ до редагування, перегляд результатів тесту усіх користувачів, які проходили тест, статистику по тесту та фото звіт проходження тесту.

Отже, визначено основні задачі, які повинен виконувати веб додаток TestBuilder, також визначено рівні доступу користувачів.

1.5 Висновки

У цьому розділі проведено аналіз сучасного стану проблеми та обґрунтовано важливість розробки веб-застосунку для проходження адаптивних тестів із фотоконтролем. Розглянуто та проаналізовано аналоги таких додатків, зокрема, JetIQ, Google Forms і Online Test Pad. Зазначені веб-додатки вирішують схожі завдання та мають функціонал, спрямований на тестування та навчання. Аналізуючи аналоги, встановлено, що розроблюваний веб-додаток виходить у лідери, особливо за урахуванням його спрямованості на навчальні заклади та включення функціоналу з фотоконтролем. Це надає йому перевагу порівняно із розглянутими аналогами.

Додаток, що розробляється, буде мати додатковий функціонал у порівнянні з розглянутими аналогами, зокрема, фотоконтроль проходження тестів, україномовний інтерфейс та буде доступним безкоштовно. Крім того, визначено конкретні завдання, які має виконувати веб-додаток, та розподілено рівні доступу для користувачів. Це забезпечить ефективне використання та відповідність потребам в освітньому середовищі. Враховуючи потреби сучасного освітнього процесу, веб-додаток повинен відповідати сучасним тенденціями у сфері дистанційного навчання та тестування. Зокрема, його інтеграція з фотоконтролем завдяки Google Cloud Vision дозволить не лише забезпечувати високу точність аналізу, але й гарантувати високий рівень безпеки та запобігати можливим спробам обману під час тестування.

Ключовим аспектом розробки буде забезпечення доступності та інтуїтивно зрозумілого інтерфейсу для обох рівнів користувачів. Викладачі, володіючи рівнем доступу для редагування та аналізу результатів, зможуть з

легкістю коригувати тести, вивчати статистику та ефективно взаємодіяти зі студентами.

Також додаток має не тільки покращити функціонал який наявний у аналогів, а і створити новий, такий як аналіз фотоконтролю за допомогою Google Cloud Vision, який надасть аналіз результатів фотоконтролю, допоможе зменшити кількість випадків порушення академічної доброчесності та полегшить роботу викладачам при перевірці результатів шляхом аналізу фотографій, які будуть зроблені під час проходження тесту, а саме вкаже що зображено на картинках, чи було використано книги або гаджети, чи була в кадрі одна особа чи декілька.

Отже, однією з переваг нового додатку є його універсальність, яка дозволяє використовувати його у будь-якому навчальному закладі. Це зробить його незамінним інструментом для вчителів, які прагнуть не лише оцінювати знання своїх студентів, але й допомагати їм у розвитку та усвідомленні власних успіхів та недоліків у навчанні.

2 РОЗРОБКА АРХІТЕКТУРИ ТА АЛГОРИТМУ РОБОТИ КЛІЄНТСЬКОЇ ЧАСТИНИ ДОДАТКУ

2.1 Розробка загальної архітектури додатку

Для ефективної розробки додатку TestBuilder, потрібно визначитись та побудувати його загальну архітектуру. Розробка загальної архітектури додатку перед його активною розробкою є важливою з кількох причин:

1. Чітке сприйняття мети додатку. Архітектура допомагає визначити основні цілі та завдання додатку. Вона надає зрозумілу структуру та визначає, як різні компоненти взаємодіють для досягнення цих цілей. Це сприяє уточненню та узгодженню очікувань між розробниками, менеджерами та замовниками.

2. Зменшення ризиків і помилок. Перед розпочатком активної розробки, створення загальної архітектури дозволяє ідентифікувати потенційні проблеми та ризики, які можуть виникнути на етапі реалізації. Це дозволяє зменшити ймовірність помилок та неправильних рішень в майбутньому.

3. Оптимізація ресурсів. Архітектура дозволяє ефективно розподіляти ресурси, визначаючи, які компоненти додатку виконують які завдання, та як вони взаємодіють. Це допомагає забезпечити оптимальне використання ресурсів, таких як час, пам'ять та обчислювальна потужність.

4. Забезпечення масштабованості. Загальна архітектура дозволяє врахувати можливості масштабування додатку, тобто його здатність ефективно функціонувати при збільшенні обсягу даних чи навантаження. Це важливо для забезпечення стабільності та продуктивності додатку у майбутньому.

5. Спрощення комунікації в команді. Загальна архітектура служить як загальний рамок для команди розробників. Вона сприяє однозначному розумінню структури та взаємодії компонентів, що полегшує комунікацію в команді та сприяє вирішенню поточних завдань.

6. Полегшення тестування та супроводу. Загальна архітектура дозволяє легше планувати тести, оскільки вона визначає, які частини додатку потрібно тестувати окремо, а які можна тестувати як єдину систему. Крім того, правильно розроблена архітектура полегшує супровід та внесення змін у майбутньому.

Отже, розробка загальної архітектури перед активною розробкою додатку є стратегічно важливим етапом, який сприяє успішному впровадженню та подальшому розвитку продукту.

Задля візуалізації архітектури програмного проєкту, прийнято використовувати схеми та діаграми: діаграма варіантів використання та діаграма послідовності.

Діаграма варіантів використання відображає взаємодію між користувачами та елементами продукту шляхом надання усіх можливих варіантів використання продукту. З її допомогою можна описати основні функції програми не вказуючи способи їх реалізації. Користувачами на цій діаграмі можуть бути безпосередньо люди, функції, програми або інші модулі, що мають вплив на роботу програми [11]. Варіантами використання визначається набір дій, що виконується системою при діалозі з користувачем.

Рисунок 2.1 – Діаграма варіантів використання

З діаграми зрозуміло, що користувач додатку може увійти до системи, після чого обрати режим роботи. В залежності від вибору користувача та обраного ним режиму роботи, система відповідним чином обробляє дані та відображає результати користувачеві.

Для відображення взаємодії користувача з системою та опису життєвого циклу структурних компонентів розроблено діаграму послідовності, яка зображена на рисунку 2.2.

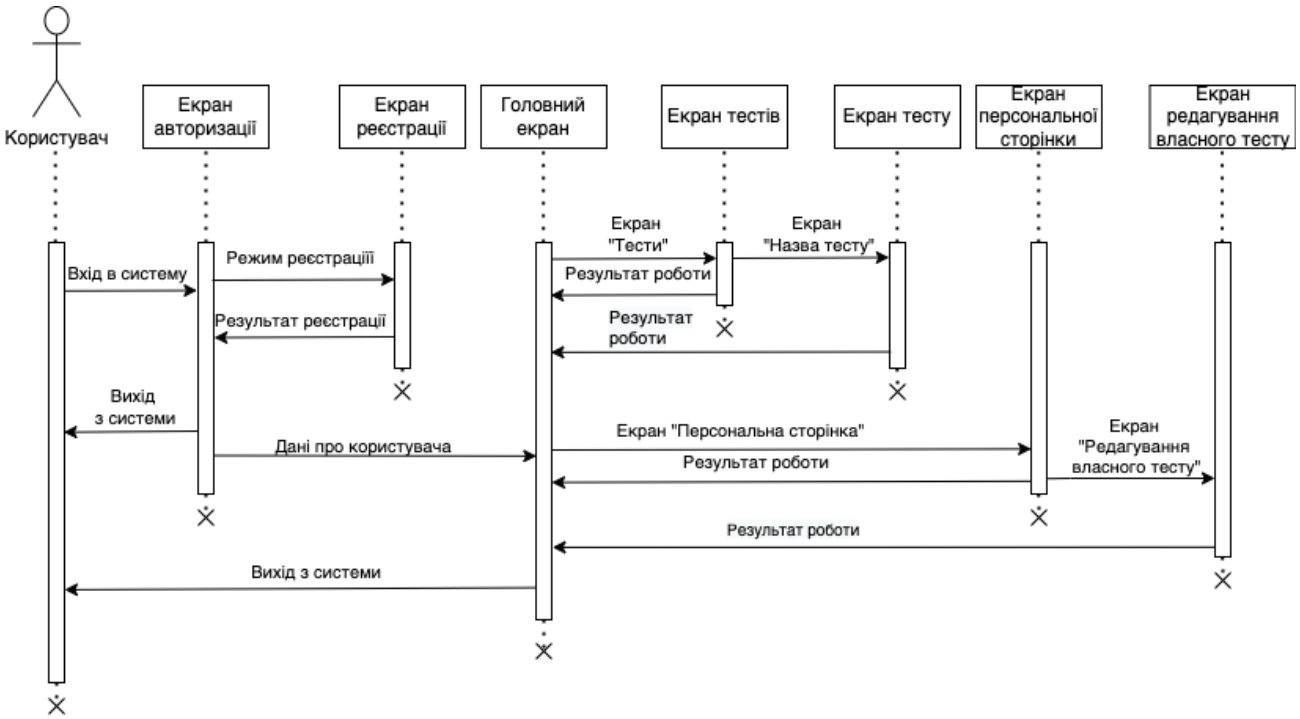


Рисунок 2.2 – Діаграма послідовності програмного додатку

Згідно розробленої діаграми послідовності можна чітко зрозуміти, які етапи обов’язково повинен пройти користувач, таким етапом є вікно авторизації або реєстрації користувача, після виконання першого етапу для користувача відкривається можливість обрати екран, зокрема екран тестів дозволяє обрати конкретний тест або перейти до екрану персональної сторінки і перейшовши до екрану редагування тесту, відредагувати його.

Для опису та демонстрації взаємодії модулів та компонентів програмного додатку між собою використовують структурну карту Константайна, вона в

першу чергу демонструє як рухаються потоки даних між модулями [12]. Розроблену структурну карту Константайна представлено на рисунку 2.3

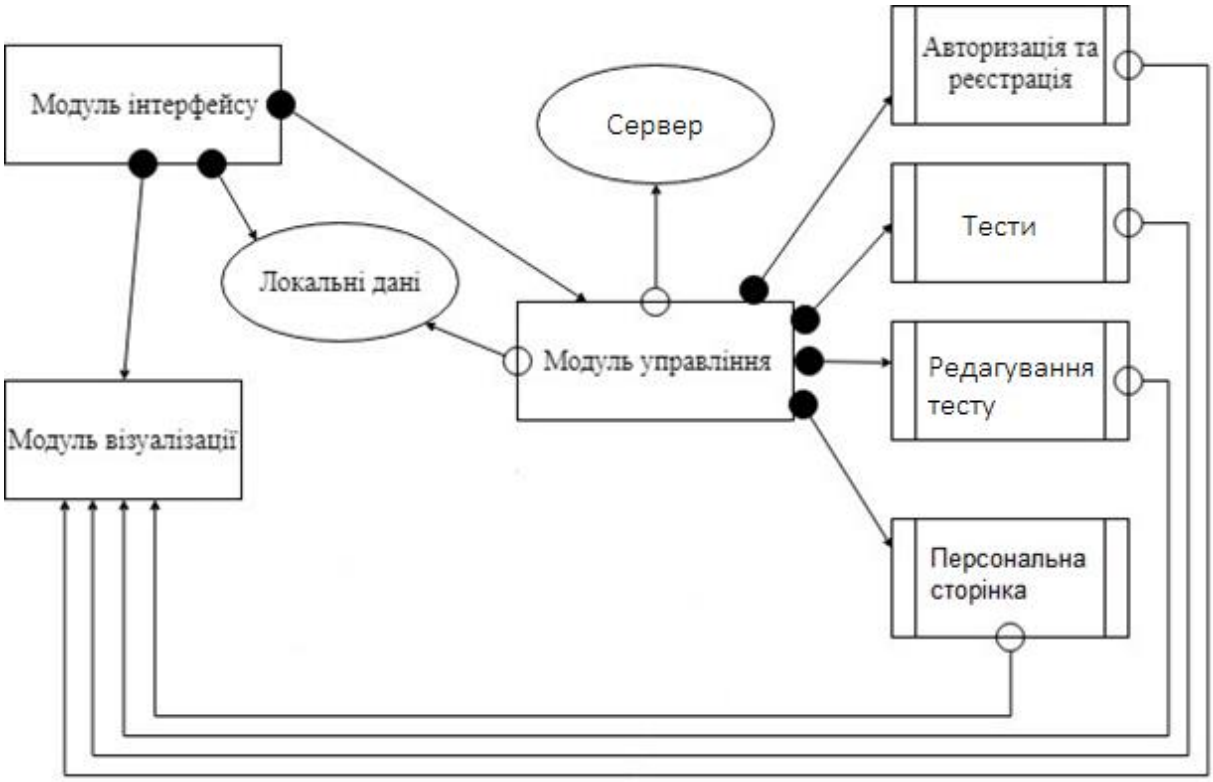


Рисунок 2.3 – Структурна карта Константайна для веб додатку

Програмний додаток складається з трьох основних модулів: управління, візуалізації та інтерфейсу. Модуль інтерфейсу пов’язаний зв’язком по управлінню з модулем візуалізації, локальними даними та модулем управління і відповідає за зовнішній вигляд вікон програми, а також реакцію на дії користувача. Модуль управління пов’язаний зв’язком по даних з локальними даними та серверною частиною. Цей модуль пов’язаний зв’язком по управлінню з основними підсистемами програми, які відповідають за її основні режими роботи. Використовуючи локальні дані та команди від інтерфейсу, він викликає одну з існуючих підсистем. Модуль візуалізації пов’язаний зв’язком по даних з усіма підсистемами додатку і відповідає за відображення вмісту інтерфейсу програми.

Отже, розроблені діаграми описують загальну архітектуру розроблюваного додатку і будуть використовуватись під час його програмної реалізації.

2.2 Розробка методу та схеми побудови архітектури типу Rich client

Будь-який додаток з яким буде взаємодіяти користувач повинен мати користувацький інтерфейс. Важлива не лише наявність інтерфейсу, а також важливою є якість виконання. Інтерфейс повинен бути зрозумілим та легким у користуванні, усі кнопки та надписи повинні відповідати модулю, до якого вони належать.

Для створення користувацького інтерфейсу буде використано принцип «Rich client». це термін, який використовується для опису програмного забезпечення, яке надає користувачам багатий набір функцій і можливостей, особливо в контексті графічного інтерфейсу користувача. «Rich client» може включати в себе різноманітні елементи, такі як графіка високої якості, інтерактивність, анімації та інші важливі функції, що роблять користування програмою більш ефективним та приємним. Однією з основних переваг «Rich clien» є здатність виконувати багато функцій локально, без необхідності постійного підключення до мережі. Це може полегшити доступ до даних та підвищити продуктивність, оскільки частину обробки можна виконувати на більш потужних локальних пристроях користувача.

«Rich client» – додатки часто використовуються у сферах, де важлива відзивчивість та можливості взаємодії, таких як графічні редактори, відеоігри, професійні програми для редагування мультимедіа та інші області, де важливий комфорт користувача та швидка обробка даних.

Саме тому даний підхід до розробки користувацьких інтерфейсів був обраний для розробки веб додатку для проходження онлайн тестів з фотоконтролем.

У випадку, якщо інтерфейс буде не якісним чи незрозумілим користувачу, досвід використання мобільним додатком буде поганим, що може зменшити бажання користувача повернутися у мобільний додаток чи зовсім змусить його видалити [13].

Для того, щоб розробити інтерфейси для кожного екрану спочатку потрібно визначити список екранів, які будуть доступні користувачеві.

Розроблюваний додаток буде складатися з наступних екранів:

1. Головна – головний екран виводить список усіх тестів.
2. Авторизація/Реєстрація – екран для вводу пошти/логіна та паролю.
3. Тест – екран який демонструє інформацію про тест та на якому відбувається процес його проходження.
4. Профіль – екран який виводить інформацію про користувача, список пройдених та створених ним тестів.
5. Редактор тесту – екран який виводить дані про тест, що був створений користувачем та інструменти для його редагування та створення.

Графічну схему інтерфейсу головного екрану мобільної інформаційної системи старости академічної групи представлено на рисунку 2.4.

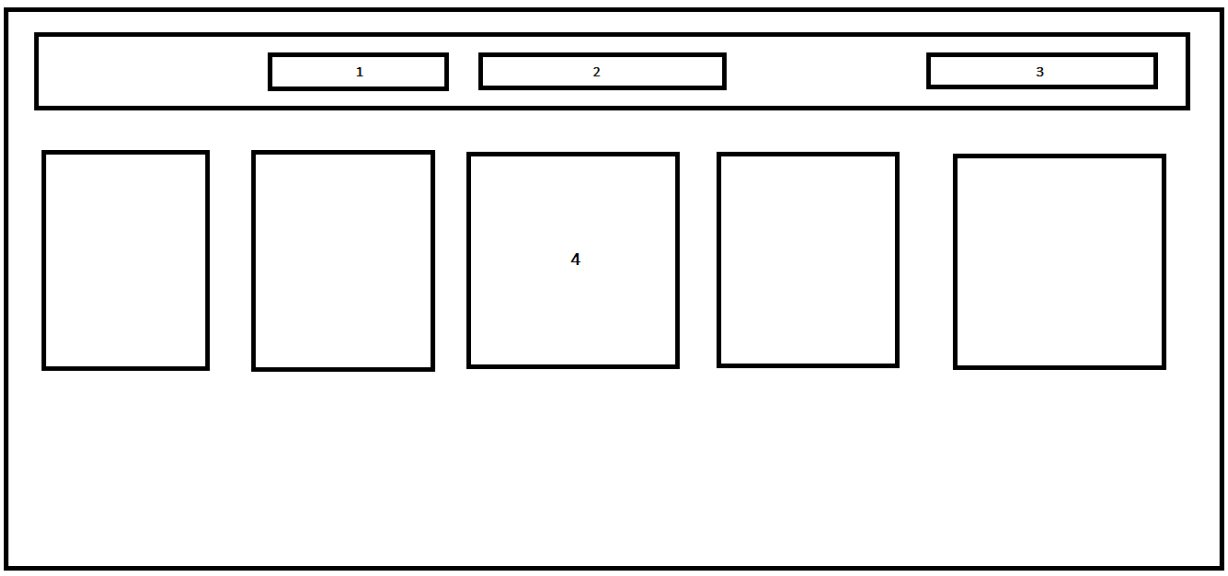


Рисунок 2.4 – Схема інтерфейсу головного екрану

Головний екран складається з наступних елементів:

1. Кнопка переходу на головний екран.
2. Кнопка переходу на сторінку користувача.
3. Кнопка переходу на сторінку авторизації.
4. Виведена інформація про тести.

Графічну схему екрану Авторизація представлено на рисунку 2.5

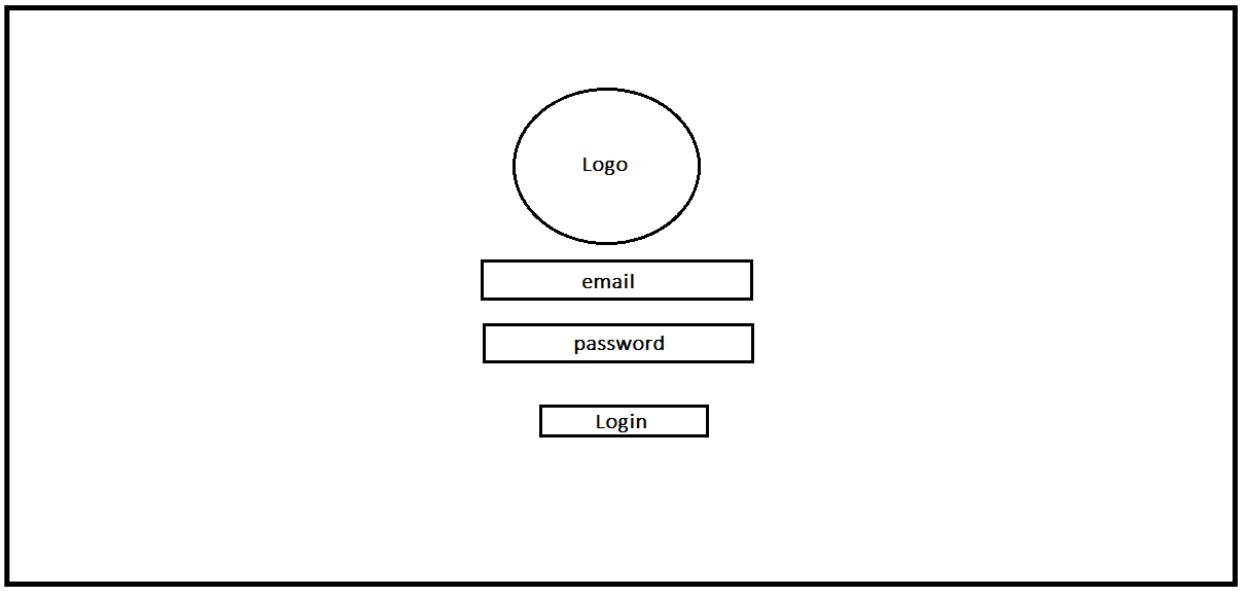


Рисунок 2.5 – Схема інтерфейсу авторизації та реєстрації в веб додатку

Важливим атрибутом веб додатку є екран з проходженням тесту, оскільки даний екран буде нести на собі головний функціонал розроблюваного веб додатку та користувач буде проводити найбільше часу на ньому

Графічну схему інтерфейсу проходження тесту представлено на рисунку 2.6.

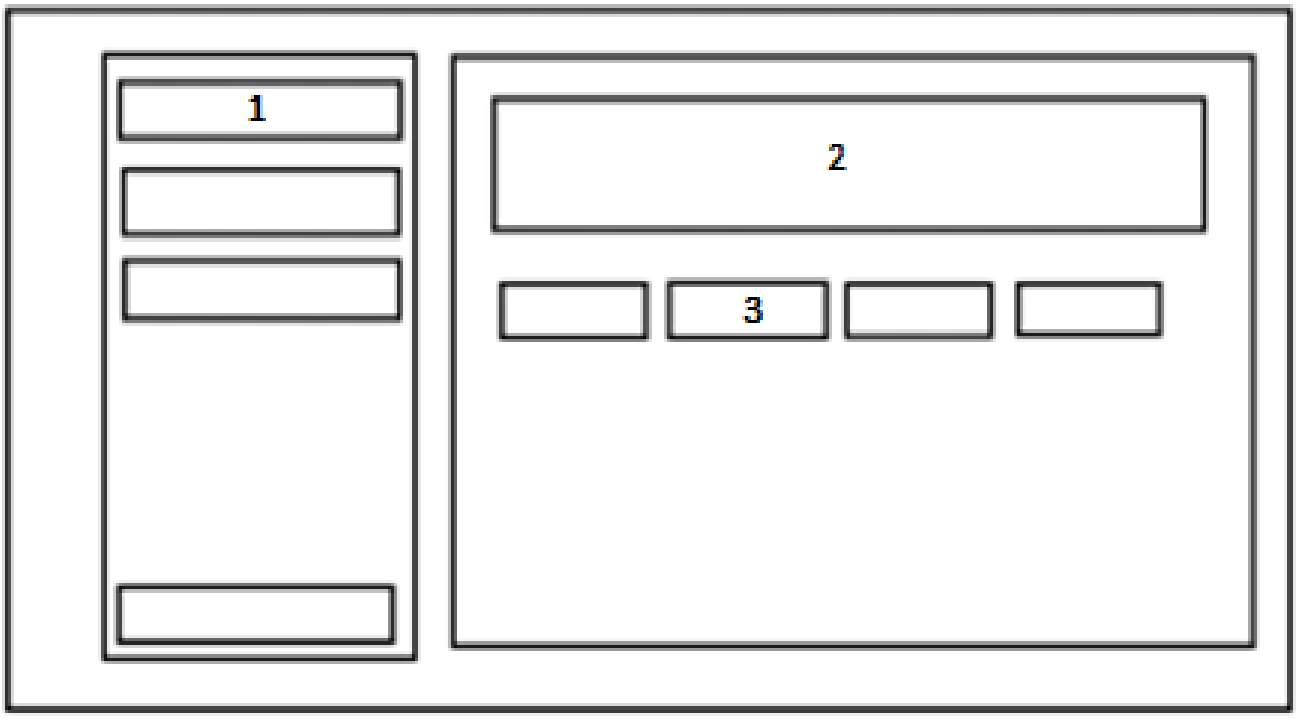


Рисунок 2.6 – Схема інтерфейсу проходження тесту

1. Назва питання.
2. Текст питання.
3. Варіанти відповіді на питання.

Також одним з важливих атрибутів веб додатку є екран з редагування та створення власного тесту, він має бути зручний у використанні, для швидкого редагування тесту та містити всі інструменти для створення та редагування питань. Ефективний і зручний у використанні екран редагування та створення тесту повинен забезпечувати простий доступ до всіх інструментів для редагування питань та тестового контенту. Це дозволяє користувачам швидко та з легкістю створювати та редагувати свої тести.

Забезпечення простоти, швидкості та повноти функцій у редагуванні тестів робить цей екран ключовим для зручності користувачів.

Дана схема інтерфейсу зображена на рисунку 2.7.

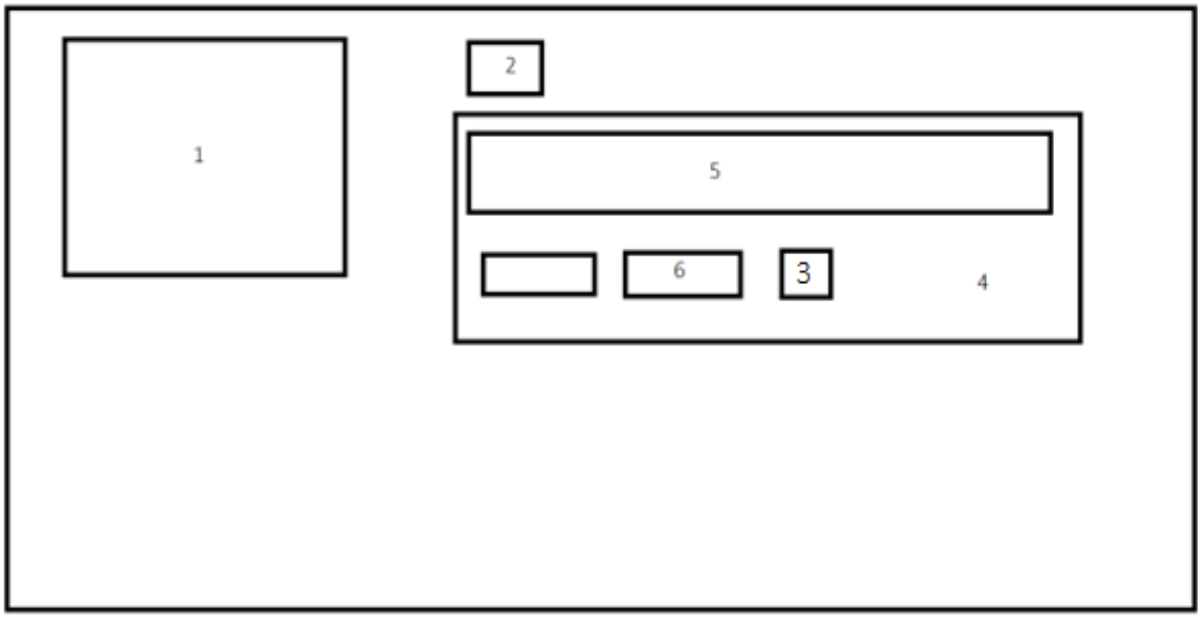


Рисунок 2.7 – Схема інтерфейсу редагування та створення тесту

1. Назва та короткий опис тесту.
2. Кнопка додання нового питання.
3. Кнопка додання варіанту відповіді.
4. Поле редагування та створення відповіді.
5. Текст запитання.

Отже, проведено розроблено схеми користувацьких інтерфейсів веб додатка. В процесі розробки були створені схеми для всіх користувацьких екранів, що є критичним кроком перед фактичною імплементацією програмного продукту.

Важливість розробки схем користувацьких екранів перед розробкою додатка важко переоцінити. Схеми надають можливість заздалегідь визначити та оптимізувати взаємодію користувача з програмою, розташування елементів інтерфейсу, порядок дій та виведення інформації. Це дозволяє уникнути потенційних помилок та непорозумінь у подальшому використанні додатка.

Розробка схем для користувацьких екранів є ефективним інструментом для забезпечення високої якості продукту, забезпечуючи зручність взаємодії з ним для кінцевого користувача. Цей етап дозволяє заздалегідь виявити та

виправити потенційні проблеми в інтерфейсі, а також підвищує шанси на успішне впровадження програмного продукту на ринку.

Таким чином, розробка схем користувацького інтерфейсу перед розробкою додатка є важливим етапом, що сприяє підвищенню ефективності та якості роботи програмного продукту, а також забезпечує задоволення потреб користувачів.

2.3 Розробка блок-схеми алгоритму роботи додатка

Наступним кроком буде розробка алгоритму роботи програмного додатку, на основі якого буде писатися програмний код.

Створення блок-схем алгоритму перед розробкою додатка є важливим етапом у процесі розробки програмного забезпечення. Ось деякі причини, чому це важливо:

1. Комунікація між командами. Блок-схеми надають зручний та універсальний спосіб для комунікації між розробниками, менеджерами та іншими учасниками проекту. Вони створюють зрозумілу інструкцію для всіх сторін щодо того, яким чином повинен працювати додаток.

2. Аналіз та вдосконалення алгоритмів. Блок-схеми дозволяють аналізувати алгоритми на ранніх етапах розробки. Вони допомагають виявити можливі проблеми та недоліки в логіці програми, що дозволяє виправити їх перед тим, як вони стануть серйозними проблемами.

3. Оцінка часових та ресурсних затрат. Розробка блок-схем дозволяє приблизно оцінити, скільки часу та ресурсів буде витрачено на виконання кожного етапу роботи. Це допомагає забезпечити реалістичні терміни та бюджет для проекту.

4. Орієнтація в проекті. Блок-схеми можуть служити важливим інструментом для орієнтації в проекті. Вони допомагають розробникам та

іншим учасникам швидше зрозуміти, як працює програма, та легше взаємодіяти з кодом.

5. Зменшення ризиків. Аналіз блок-схем дозволяє ідентифікувати можливі ризики та труднощі на етапі проектування. Це дозволяє зменшити ймовірність помилок та неполадок під час реалізації.

6. Стандартизація розробки. Використання блок-схем сприяє стандартизації процесу розробки. Це важливо, особливо у великих командах, де декілька людей може брати участь у створенні одного додатка.

7. Документація і навчання. Блок-схеми стають важливою частиною документації проекту. Вони надають зрозумілу та легко читабельну інструкцію для розробників, тестувальників та інших учасників проекту. Крім того, ці блок-схеми можуть використовуватися для навчання нових членів команди або для підготовки документації для користувачів.

8. Тестування та валідація. Блок-схеми можуть бути використані для планування тестів та валідації програми. Це дозволяє визначити, які частини програми потрібно перевірити, а також сприяє створенню ефективних тестових кейсів.

9. Зручність рефакторингу. Якщо під час розробки виникають зміни в логіці програми або потрібно внести покращення, блок-схеми можуть слугувати допоміжним інструментом для швидкого розуміння взаємозв'язків між різними частинами коду. Це полегшує рефакторинг та збереження структурної цілісності додатка.

10. Зручність рефакторингу. Якщо під час розробки виникають зміни в логіці програми або потрібно внести покращення, блок-схеми можуть слугувати допоміжним інструментом для швидкого розуміння взаємозв'язків між різними частинами коду. Це полегшує рефакторинг та збереження структурної цілісності додатка.

Блок–схема алгоритму роботи програмного додатку представлено на рисунку 2.8.

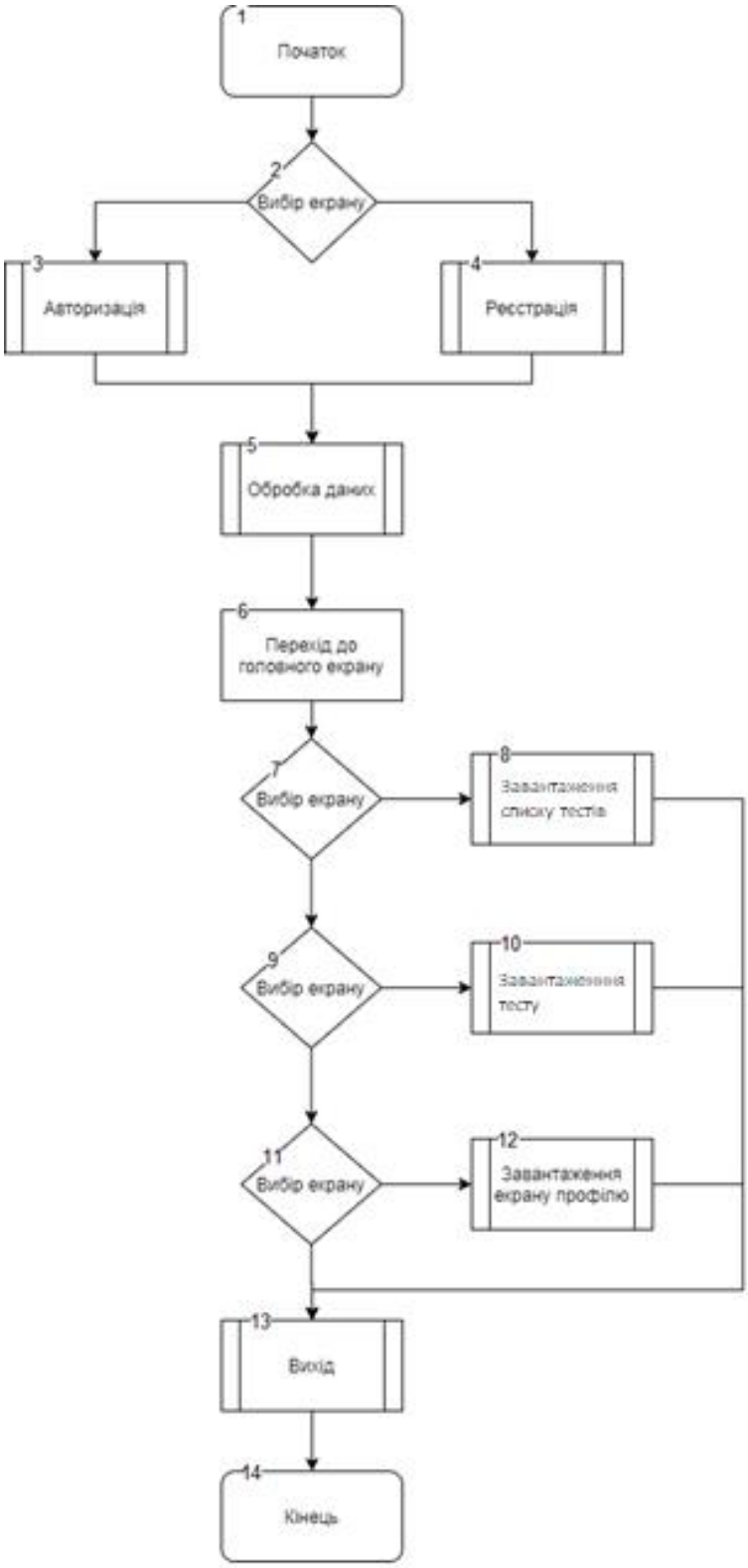


Рисунок 2.8 – Загальний алгоритм роботи програмного додатку

Отже, блок-схеми алгоритму роботи додатка є ефективним інструментом для покращення комунікації, аналізу та управління ризиками під час розробки програмного забезпечення.

2.4 Розробка методу та блок-схеми алгоритму роботи фотоконтролю

Фотоконтроль є ефективним методом в системі контролю за процесом тестування, а його реалізація в тестувальній системі виглядає особливо перспективною. З урахуванням стрімкого розвитку технологій, зокрема HTML5 та високорівневих API JavaScript, відкриваються нові можливості для використання апаратних ресурсів пристроїв [14].

Особливий акцент у тексті робиться на використанні фотоконтролю в розробці тестувальної системи. Реалізація архітектури системи фотоконтролю включає можливість проведення тестування та збереження зображень фотоконтролю на сервері. Важливим аспектом є можливість перегляду результатів тестування та фотоконтролю для викладачів.

Окрім того, для оптимальної роботи з фотоконтролем, використано сервіс "PhotoRegister". Цей сервіс спеціалізується на обробці та зберіганні результатів фотофіксації. Зокрема, звертається до сервера за допомогою модулю для передачі фотографій користувачів, які проходять тест, на сервер. При цьому використання модулів для взаємодії із сервером та роботи з фотоконтролем робить веб-додаток ще більш функціональним та ефективним.

Зазначається також важливість використання Google Cloud Vision у контексті фотоконтролю. Це надає системі високу точність у розпізнаванні та аналізі зображень, що є ключовим для забезпечення об'єктивності та достовірності фотоконтролю в системі тестування.

Для забезпечення безпеки та конфіденційності даних у системі фотоконтролю використовується шифрування при передачі та зберіганні зображень на сервері. Також враховується важливість визначення прав доступу до фотоконтролю для різних користувачів системи, забезпечуючи відповідність з правилами конфіденційності та обов'язковості.

У контексті освітніх інституцій важливим стає також інтеграція системи фотоконтролю з іншими педагогічними платформами та системами управління навчанням. Це дозволяє автоматизувати процеси оцінювання та моніторингу

навчального процесу, забезпечуючи вчителям та адміністраторам зручний доступ до інформації про продуктивність студентів.

Крім того, система фотоконтролю може бути розширена функціональністю розпізнавання облич, що дозволяє автоматично ідентифікувати студентів під час тестування. Це не лише полегшує процес фотоконтролю, але і робить його більш точним та швидким.

Загалом, використання фотоконтролю в тестувальній системі є перспективним та важливим кроком у напрямку покращення надійності та об'єктивності процесу оцінювання студентів. Технології HTML5, JavaScript та інші високорівневі API відкривають широкі можливості для реалізації таких систем, а їхнє поєднання з сервісами, такими як "PhotoRegister" та Google Cloud Vision, додає ефективності та надійності у процесі тестування.

На рисунку 2.9 зображене схематичне зображення розробленої архітектури.

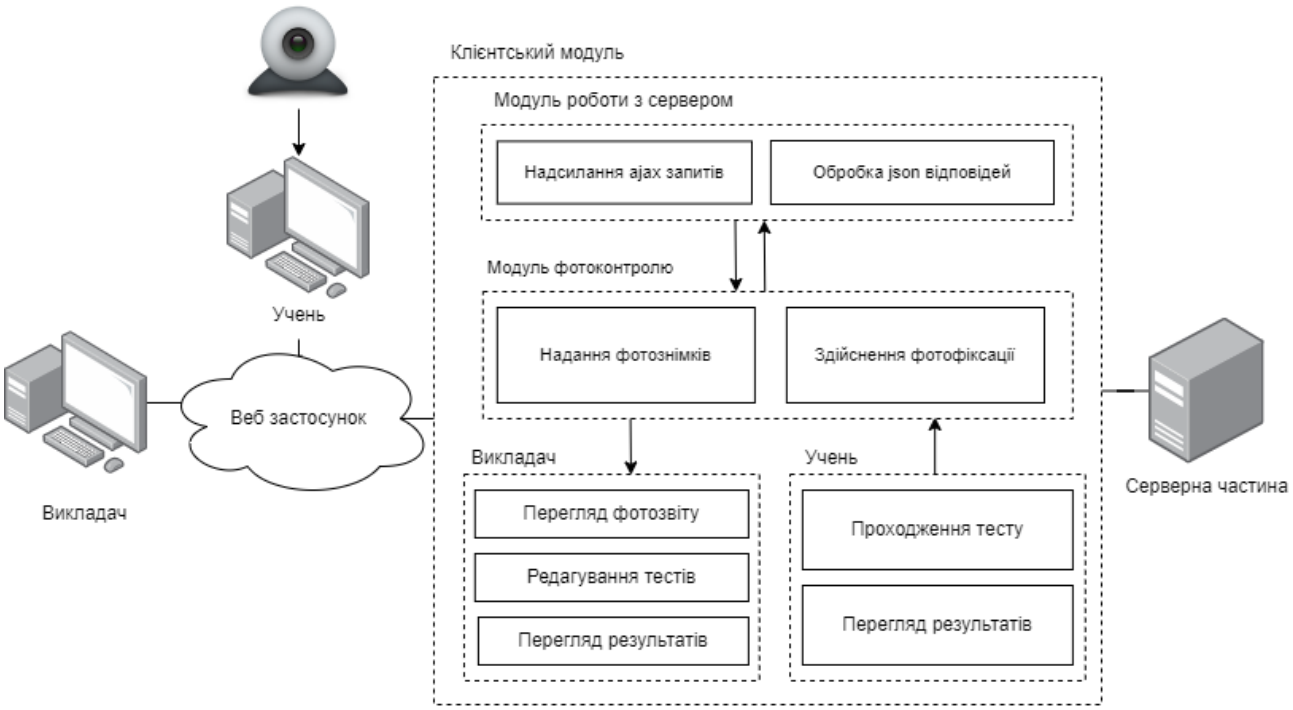


Рисунок 2.9 – Розроблена архітектура клієнтського модулю фотоконтролю

Наступним кроком буде розробити алгоритм проходження тесту та здійснення фотофіксації. Блок-схема алгоритму проходження тесту та здійснення фотофіксації представлено на рисунку 2.10. Фотофіксація здійснюється шляхом створення фотографій з веб камери користувача у певний період часу. Таким чином користувач не знає в який саме момент часу здійснюється фотофіксація.

Зображення зберігається у форматі png та відправляється на сервер шляхом переводу його в послідовність байтів, таким чином воно надсилається швидше.

Перед початком тестування система ініціює процес фотофіксації, сповіщаючи користувача про включення веб-камери для забезпечення об'єктивності та достовірності фотоконтролю. За допомогою високорівневих API JavaScript, система активує веб-камеру та запускає таймер, який регулює частоту створення фотографій.

Процес тестування та фотофіксації відбувається паралельно. Кожен етап тестування пов'язаний з певним інтервалом часу, протягом якого здійснюється фотофіксація. Це виконується для того, щоб забезпечити адекватний охоплення процесу тестування та уникнути можливості підготовки до фотофіксації.

Отримані зображення зберігаються у форматі PNG, який забезпечує високу якість та стисліть без втрат. Кожне зображення перетворюється в послідовність байтів для швидкого та ефективного передавання на сервер.

На сервері фотографії приймаються та зберігаються в безпечному місці, а інформація про час їхнього створення фіксується. Після завершення тестування система може здійснювати автоматичний аналіз фотографій за допомогою Google Cloud Vision для перевірки наявності шахрайства або нечесності.

Забезпечуючи паралельний процес тестування та фотофіксації, система мінімізує можливість маніпуляцій з боку користувача та забезпечує точність та надійність в системі фотоконтролю.

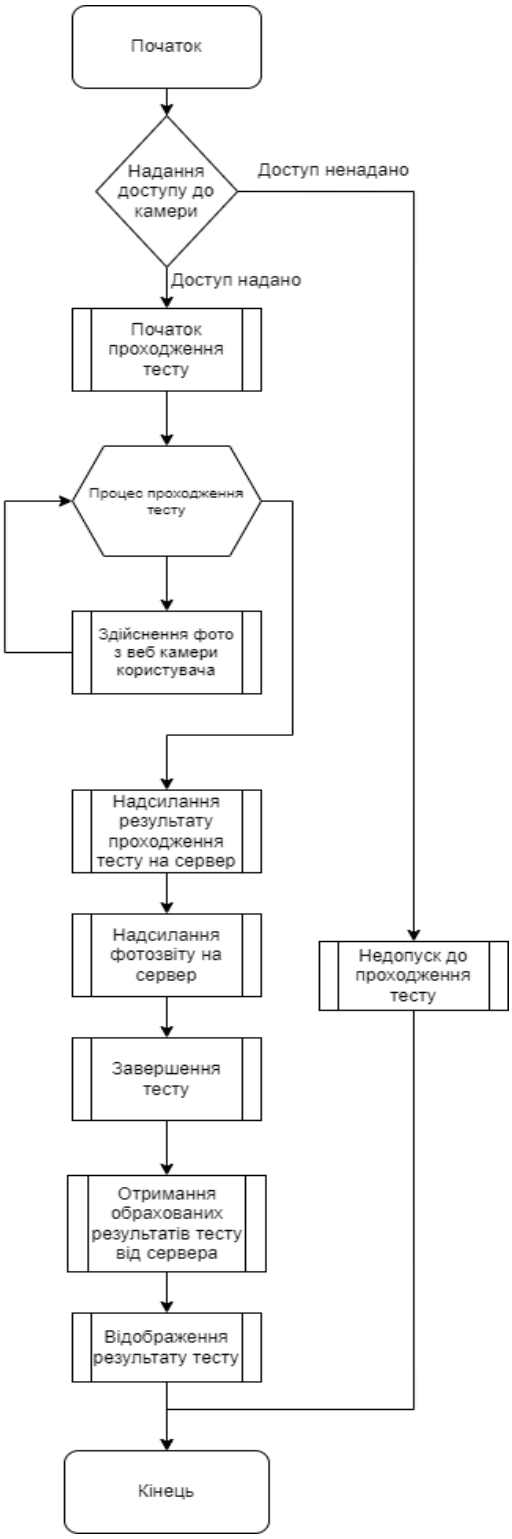


Рисунок 2.10 – Загальний алгоритм проходження тесту з фотоконтролем

Отже, у даному підрозділі розроблено загальний алгоритм блок-схеми алгоритму роботи фотоконтролю та архітектури клієнтського модулю фотоконтролю, використовуючи ці схеми і буде проводитися розробка додатку.

2.5 Висновки

У цьому розділі розроблено загальну архітектуру веб додатку, розроблено діаграми послідовності, діаграми варіантів та структурну карту Константайна. Розроблено схеми користувацького інтерфейсу для кожного екрану присутнього в веб додатку. Також побудовано загальний алгоритм та архітектуру роботи програмного додатку та проходження тесту з фотофіксацією. Для кращого розуміння та управління веб-додатком, була створена структурна карта Константайна, яка відображає взаємодію між основними компонентами системи та їхніми залежностями. Ця карта дозволяє легко визначити основні елементи архітектури та їхні функціональні зв'язки.

Алгоритм проходження тесту з фотофіксацією включає в себе взаємодію користувача з інтерфейсом, активацію фотоконтролю, таймер для фотофіксації та передачу зображень на сервер для зберігання та аналізу. Кожен з етапів має свої власні обробники та контрольні точки, що забезпечують правильну роботу та безпеку проходження тесту.

Схеми користувацького інтерфейсу для кожного екрану відображають практичний та зрозумілий дизайн для кінцевого користувача, забезпечуючи йому зручність та легкість взаємодії з системою.

Загальна архітектура веб-додатку розрахована на ефективну роботу з фотоконтролем, використовуючи високорівневі технології та сервіси, такі як "PhotoRegister" та Google Cloud Vision, для забезпечення надійності та об'єктивності процесу тестування.

Отже, розробленої архітектурою, діаграмами та схемами забезпечується повнота та структурованість веб-додатку, що робить його ефективним та легким у розумінні та управлінні.

3 РОЗРОБКА КОДУ КЛІЄНТСЬКОГО МОДУЛЮ З ФОТОКОНТРОЛЕМ

3.1 Варіантний аналіз і обґрунтування вибору мови програмування

Вибір мови програмування – комплексне й складне питання, від якого залежать подальші процеси розробки, вибору інструментів та бібліотек, а в деяких випадках і засобів для тестування.

Вибір мови програмування є критичним етапом у процесі проектування програмного додатку і може суттєво впливати на успіх проекту. Ось деякі причини, чому це важливо:

1. Функціональність. Різні мови програмування придатні для різних завдань. Деякі мови підходять для веб-розробки, інші - для мобільних додатків чи обчислювальних завдань. Вибір мови повинен відповідати функціональним вимогам вашого проекту.

2. Спільнота та Екосистема. Деякі мови мають великі та активні спільноти розробників, а також розвинуті інструменти та бібліотеки. Це полегшує вирішення проблем та забезпечує підтримку.

3. Продуктивність. Деякі мови можуть бути більш продуктивними для конкретних завдань чи для конкретного розробника. Важливо врахувати, наскільки швидко можна створити функціональний код та як він ефективний.

4. Масштабованість. Якщо передбачається зростання обсягів даних чи користувачів, важливо вибрати мову, яка легко масштабується та має підтримку для паралельного виконання завдань.

5. Безпека. Деякі мови мають вбудовані механізми безпеки, що дозволяє уникати деяких типових помилок програмування, таких як переповнення буфера чи витоки пам'яті.

6. Вартість розробки та підтримки. Вибір мови може вплинути на вартість розробки та підтримки. Деякі мови мають більше доступних розробників, що може знизити вартість найму та підтримки.

7. Сумісність. Важливо враховувати сумісність з існуючим програмним забезпеченням та інфраструктурою.

Отже, проведемо аналіз проведемо аналіз загальновідомих мов програмування, які можуть бути використанні для написання веб-додатку.

JavaScript – мультипарадигменна мова програмування. Підтримує об'єктно-орієнтований, імперативний та функціональний стилі. Є реалізацією специфікації ECMAScript. JavaScript зазвичай використовується як вбудована мова для програмного доступу до об'єктів програм. Найширше застосування знаходить у браузерах як мову сценаріїв для надання інтерактивності веб-сторінкам [15].

TypeScript – мова програмування, представлена Microsoft у 2012 році і позиціонується як засіб розробки веб-додатків, що розширює можливості JavaScript. TypeScript є сумісним з JavaScript і компілюється в останній. Фактично, після компіляції програму на TypeScript можна виконувати у будь-якому сучасному браузері або використовувати разом із серверною платформою Node.js. Код експериментального компілятора, що транслює TypeScript JavaScript, поширюється під ліцензією Apache. Його технологія ведеться в громадському репозиторії через обслуговування GitHub. TypeScript відрізняється від JavaScript можливістю явного статичного призначення типів, підтримкою використання повноцінних класів (як у традиційних об'єктно-орієнтованих мовах), а також підтримкою підключення модулів, що покликане підвищити швидкість розробки, полегшити читання, рефакторинг та повторне використання коду, допомогти здійснювати пошук помилок на етапі розробки та компіляції, і, можливо, прискорити виконання програм [16].

C++ – компільована, статично типізована мова програмування загального призначення. Підтримує такі парадигми програмування, як процедурне програмування, об'єктно-орієнтоване програмування, узагальнене програмування. Мова має багату стандартну бібліотеку, яка включає поширені контейнери і алгоритми, введення-виведення, регулярні висловлювання, підтримку багатопоточності та інші можливості. C++ поєднує властивості як

високорівневих, і низькорівневих мов. У порівнянні з його попередником - мовою C - найбільшу увагу приділено підтримці об'єктно-орієнтованого та узагальненого програмування [17].

Java – суворо типізована об'єктно-орієнтована мова програмування загального призначення, розроблена компанією Sun Microsystems (надалі придбаною компанією Oracle). Розробка ведеться спільнотою, організованою через Java Community Process; мова та основні технології, що її реалізують, поширюються за ліцензією GPL. Права на торгову марку належать корпорації Oracle. Програми Java зазвичай транслюються у спеціальний байт-код, тому вони можуть працювати на будь-якій комп'ютерній архітектурі, для якої існує реалізація віртуальної Java-машини [18].

Таблиця 3.1 – Порівняння мов програмування

Критерій	C++	Java	JavaScript/TypeScript
Швидкодія розробленого ПЗ	+	+	+
Синтаксис та зручність розробки	-	-	+
Менеджмент залежностей (бібліотек)	-	-	+
Кросплатформність	-	+	+
Широкий вибір IDE та інших інструментів	-	+	+

Продовження таблиці 3.1

Ручна динамічна робота з пам’яттю	+	-	-
Багато поточність	+	+	-
Підсумок	3	4	5

За результатами порівняння найкраще для розробки програмного забезпечення для управління конфігураціями підходить JavaScript/TypeScript. Ця мова програмування популярна серед Frontend інженерів та за допомогою неї розроблено більшість сучасних веб додатків.

3.2 Обґрунтування вибору середовища розробки

Вибір середовища розробки перед початком безпосередньої розробки є критичним етапом у процесі створення програмного продукту. Це важливо з числа різних причин:

1. Ефективність розробки. Вибір оптимального середовища може значно полегшити робочий процес і зробити розробку ефективнішою. Розробники можуть швидше писати код, використовуючи потужні інструменти та функції, які надає середовище.

2. Сумісність. Обране середовище повинно бути сумісним із тими технологіями, які ви плануєте використовувати у своєму проекті. Наприклад, якщо ви плануєте розробку веб-додатка, то важливо вибрати середовище, яке підтримує потрібні мови програмування та фреймворки.

3. Зручність відладки. Деякі середовища роблять відлагодження (debugging) значно простішим. Важливо мати доступ до інструментів для відстеження помилок та ефективного виправлення їх.

4. Масштабованість. Якщо ваш проект планується масштабувати в майбутньому, важливо вибрати середовище, яке підтримує розвиток та розширення проекту без великих зусиль.

5. Спільна робота. Якщо ви працюєте в команді, важливо вибрати середовище, яке дозволяє легко спільно працювати, ведучи контроль версій, обмінюючись кодом і координуючи роботу над проектом.

6. Безпека. Залежно від типу проекту і сфери використання, важливо враховувати питання безпеки та вибирати середовище, яке дозволяє вам ефективно впроваджувати заходи забезпечення.

7. Спільнота та підтримка. Популярні середовища мають активні спільноти користувачів та широку базу знань, що полегшує вирішення проблем і забезпечує доступ до нововведень.

8. Вартість. Вартість використання певних середовищ може вплинути на бюджет проекту. Важливо розглядати витрати на ліцензії, інфраструктуру та інші витрати.

Вибір оптимального середовища розробки перед початком проекту допоможе забезпечити успішну та продуктивну розробку програмного продукту.

Для розробки клієнтського коду непотрібно використовувати особливих середовищ розробки, клієнтський код можна писати і в блокноті, тому при виборі середовища розробки потрібно звертати увагу на комфорт при написанні коду, на можливість підключення модулів та інших функцій, які можуть допомогти при написанні програмного коду, для розгляду візьмемо наступні популярні рішення: Visual Studio Code, WebStorm.

Visual Studio Code – звичайний редактор коду, який має простий за можливостями вбудований дебагер та не може похизуватися вбудованою кількістю плагінів, які можуть спростити написання коду але надає можливість власноруч наповнити редактор потрібними плагінами та модулями, головною перевагою є маленька використовуваної оперативної пам’яті комп’ютера [19]. Він має багатомовний інтерфейс користувача та підтримує ряд мов

програмування, підсвітку синтаксису, IntelliSense, рефакторинг, налагодження, навігацію за кодом, підтримку Git та інші можливості. Багато можливостей Visual Studio Code недоступні через графічний інтерфейс, часті вони використовуються через палітру команд або JSON-файли (наприклад, користувацькі). Палітра команд являє собою подібне до командного рядка, яка викликається комбінацією клавіш.

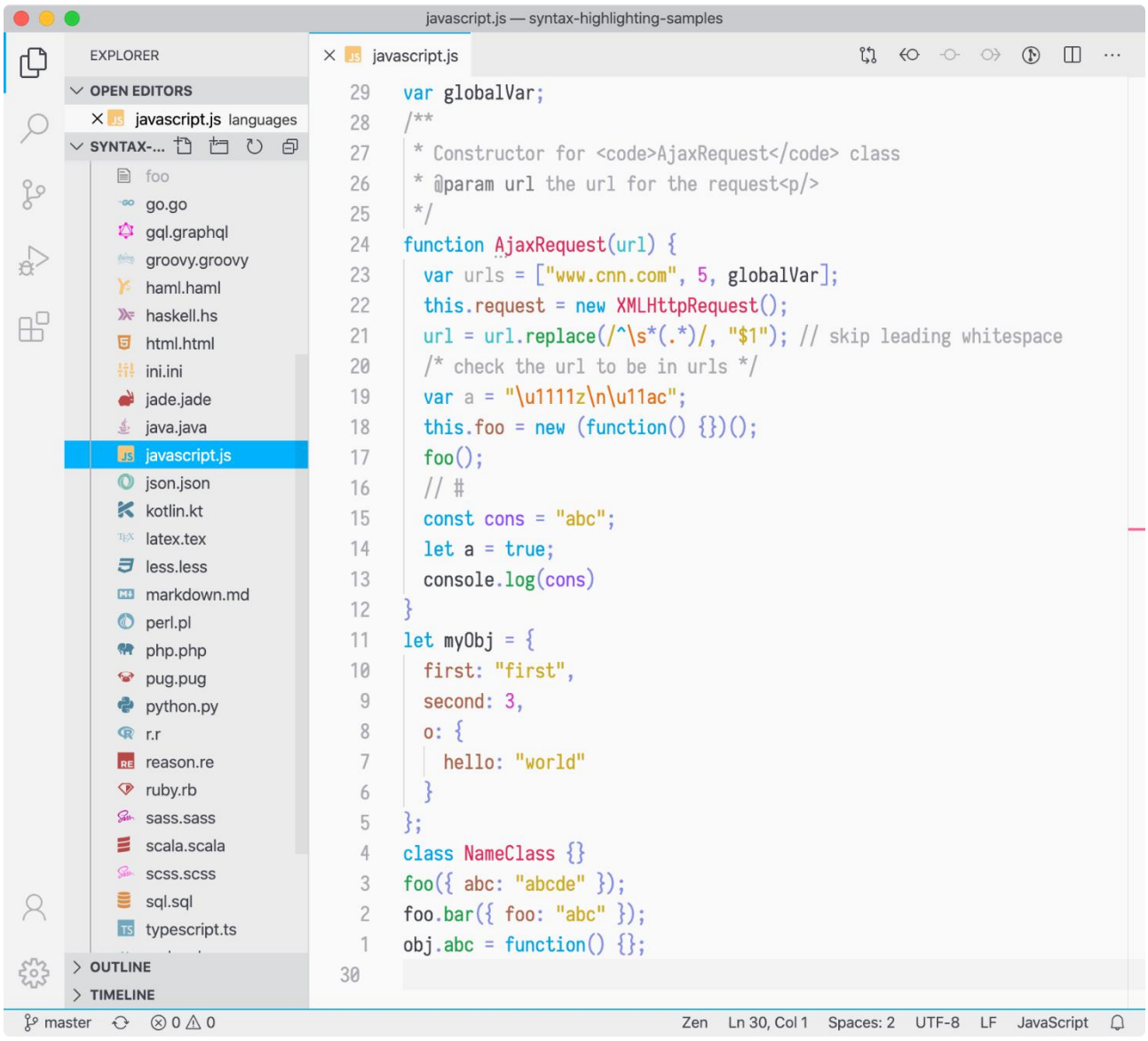


Рисунок 3.1 – Приклад інтерфейсу Visual Studio Code

WebStorm – на відміну від VS Code в даному середовищі розробки найпотрібніші плагіни одразу інтегровані в середовище, без потреби

налаштовувати. Також WebStorm підтримує синтаксис багатьох фреймворків для створення клієнтських частин веб додатків. Але WebStorm є платним редактором коду на відміну від VS Code і використовує більшу кількість оперативної пам’яті під час роботи [20].

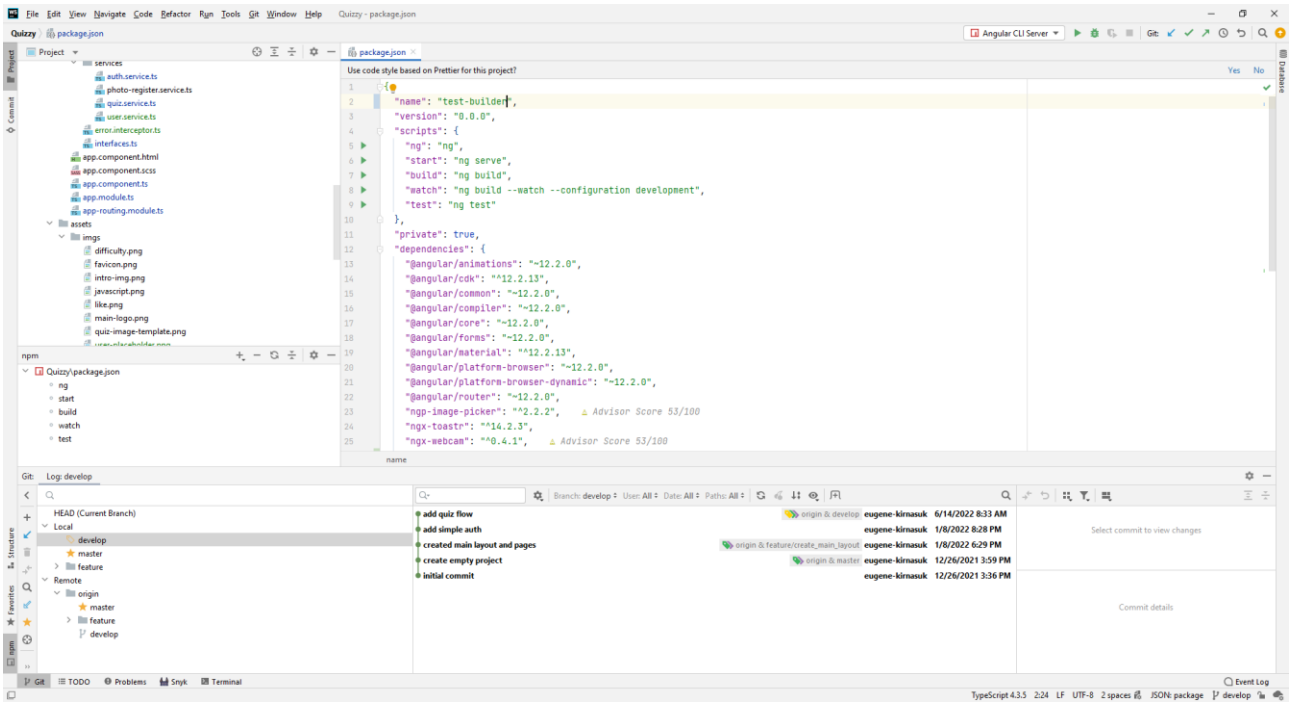


Рисунок 3.2 – Приклад інтерфейсу WebStorm

Отже, в даному підрозділі розглянуто базові відомості про IDE та проведено аналіз популярних середовищ розробки для мови програмування TypeScript/JavaScript та фреймворку Angular. Досліджено WebStorm та Visual Studio Code. За результатами порівняння найкраще для розробки програмного продукту підходить WebStorm.

3.3 Обґрунтування вибору сервісу Google Cloud Vision

Google Cloud Vision – це служба компанії Google, яка надає можливість використання розпізнавання образів та комп'ютерного зору у ваших програмах

та проектах. Цей сервіс пропонує API для розв'язання завдань, пов'язаних з обробкою та розпізнаванням зображень [21].

Порівнюючи Google Cloud Vision з аналогами, такими як Amazon Rekognition та Microsoft Azure Computer Vision, можна виділити кілька ключових аспектів:

Функціональність:

- Google Cloud Vision: Має широкий спектр функціональностей для розпізнавання облич, аналізу зображень, локалізації об'єктів, класифікації зображень і іншого.
- Amazon Rekognition: Також надає розширені можливості розпізнавання облич, об'єктів та тексту на зображеннях.
- Microsoft Azure Computer Vision: Забезпечує аналіз зображень, виявлення об'єктів, розпізнавання тексту і тематичний аналіз.

Точність:

- Google Cloud Vision: Відомий своєю високою точністю завдяки використанню технологій глибокого навчання.
- Amazon Rekognition: Теж демонструє добрі результати точності завдяки використанню подібних технологій.
- Microsoft Azure Computer Vision: Відомий своєю точністю в розпізнаванні об'єктів і тексту.

Інтеграція:

- Google Cloud Vision: Легко інтегрується з іншими сервісами Google Cloud.
- Amazon Rekognition: Інтегрується з іншими сервісами AWS, що спрощує використання для користувачів Amazon Web Services.
- Microsoft Azure Computer Vision: Інтегрується з іншими послугами Azure, забезпечуючи комплексний підхід до хмарних обчислень.

Вартість:

- Google Cloud Vision: Має свою систему тарифів, що може бути конкурентноспроможною, але вартість використання може залежати від конкретних вимог користувача.

Amazon Rekognition: Також пропонує тарифи в межах AWS, і вартість може варіюватися залежно від обсягу використання.

- Microsoft Azure Computer Vision: Має свою систему ціноутворення, яка може відрізнятися від інших хмарних платформ.

Захист і конфіденційність:

- Google Cloud Vision, Amazon Rekognition, Microsoft Azure Computer Vision: Усі три сервіси приділяють велику увагу захисту даних та конфіденційності користувачів.

Загалом, Google Cloud Vision є потужним інструментом для обробки та аналізу зображень, володіє великою функціональністю та надійністю.

Важливо відзначити, що Google Cloud Vision використовує глибокі нейронні мережі для розпізнавання образів. Це означає, що алгоритми базуються на великому обсязі навчальних даних та потужних моделях глибокого навчання, що дозволяє їм високоефективно розпізнавати різні об'єкти та характеристики на зображеннях. Алгоритм роботи Google Cloud Vision зображено рисунок 3.3.

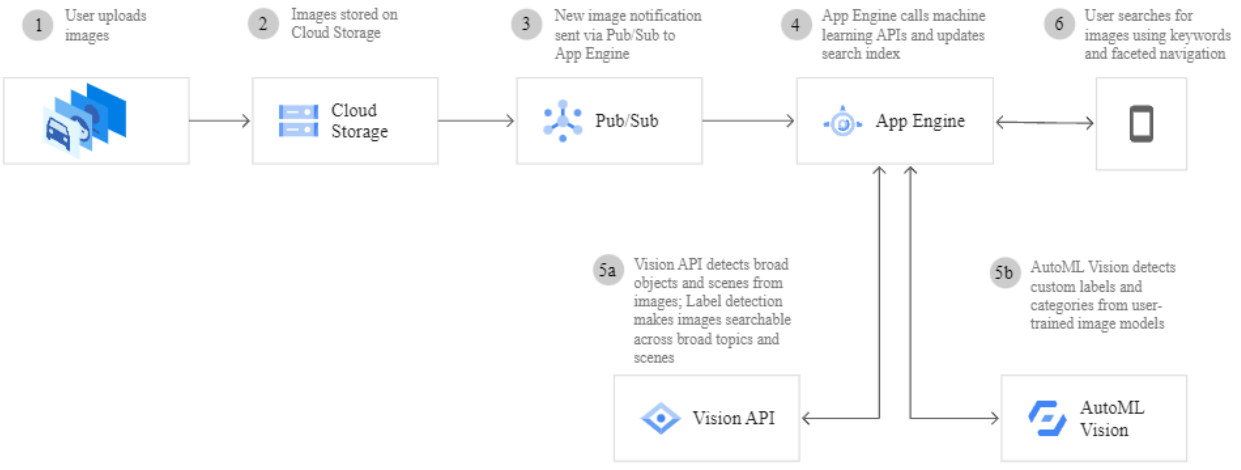


Рисунок 3.3 – Алгоритм роботи Google Cloud Vision

Алгоритм роботи Google Cloud Vision полягає в кількох основних етапах:

1) Надсилання запиту. Програма чи додаток, що використовує Google Cloud Vision, надсилає HTTP-запит до Vision API, вказуючи опції конфігурації та зображення, яке потрібно обробити.

2) Обробка на сервері. Google Cloud Vision API отримує запит та обробляє зображення. Під час обробки відбувається розпізнавання об'єктів, тексту, облич, етикет і інших вказаних характеристик.

3) Генерація відповіді. API повертає відповідь у форматі JSON, яка містить результати аналізу зображення. Ця відповідь містить різні поля з інформацією, яку запитали, наприклад, об'єкти на зображенні, витягнутий текст, визначені обличчя та інше.

4) Інтерпретація відповіді. Програма обробляє отриману відповідь і використовує результати розпізнавання зображення відповідно до своїх потреб. Наприклад, для відображення інформації про розпізнані об'єктів на зображенні, використовувати текст для подальшого аналізу чи взаємодії з користувачем.

Отже, використання Google Cloud Vision може значно спростити і поліпшити обробку зображень у проекті, дозволяючи ефективно використовувати потужності машинного зору без необхідності глибокого розуміння та реалізації алгоритмів машинного навчання.

3.4 Розробка модулю клієнта

Кожен модуль візуалізації (компонента) складається з візуальної частини HTML та CSS та функціоналу JavaScript що і використовує фреймворк Angular, за допомогою якого відбувається взаємодія з користувачем. Для прикладу можна взяти реалізацію створення тесту, де було використано для візуальної частини компонети HTML та CSS (рис.3.4), функціональної частини компоненти та для сервісу, в якому була реалізована уся логіка, що пов’язана з створенням та редагуванням тесту, виконано на мові програмування TypeScript рисунок 3.5 та 3.6 [22].

Angular – написаний на TypeScript front-end фреймворк з відкритим кодом, який розробляється під керівництвом Angular Team у компанії Google, а також спільнотою приватних розробників та корпорацій. Angular – це AngularJS, який був переосмислений та перероблений тією ж командою розробників. Angular використовує принципи сервіco-орієнтованого та компонентного програмування [23].

Компонентне програмування – це узагальнення ООП, орієнтоване на повторне використання програмних компонентів – незалежних від мови програмування самостійно реалізованих програмних об’єктів, які забезпечують виконання певної сукупності сервісів і представлених як контейнери з доступом до них через інтерфейс. Для інтеграції компонентів в кінцеві програми використовують окрім контейнерів такі типові засоби, як патерни та каркаси [24]. Приклад модулю візуалізації (компонента) зображено на рисунку 3.4.

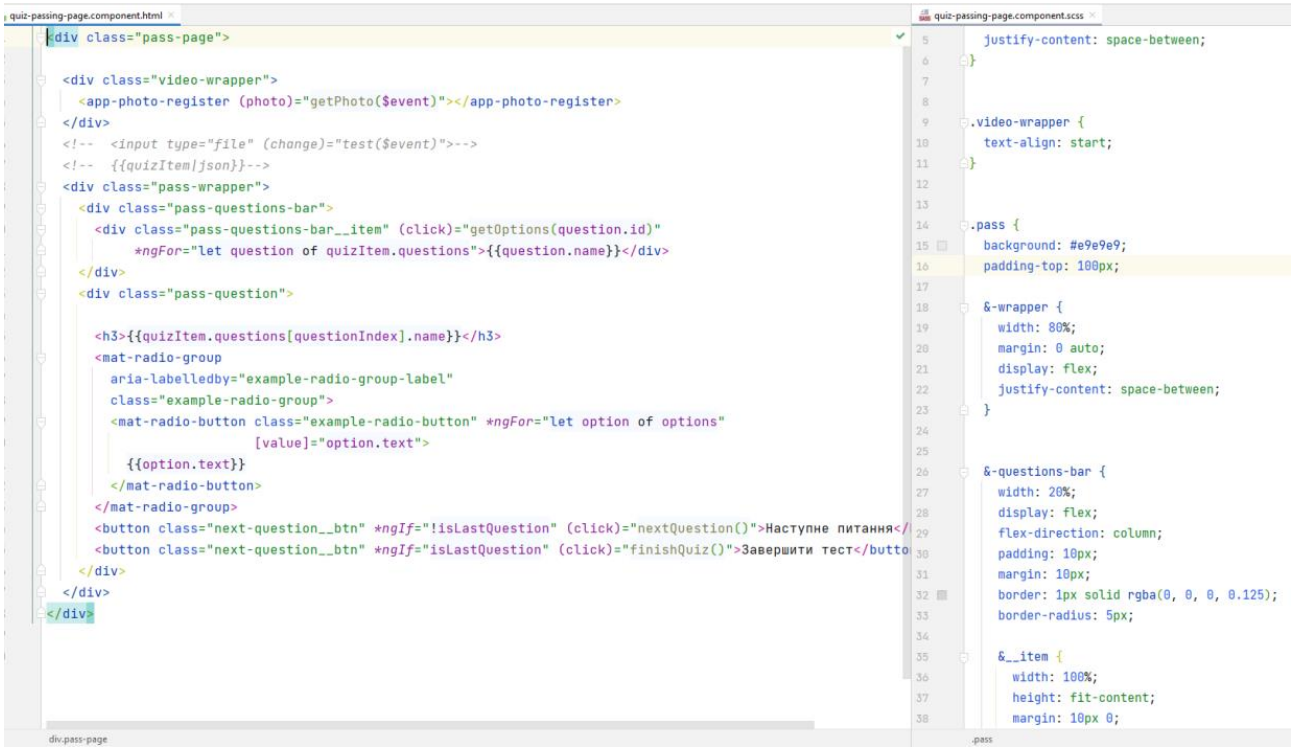


Рисунок 3.4 – Модуль візуалізації (компонента)

Сервісно-орієнтована архітектура (СОА) — це підхід до розробки програмного забезпечення, при якому програма розбивається на невеликі, незалежні сервіси, що можуть взаємодіяти один з одним. Кожен сервіс виконує конкретну функцію і має власний інтерфейс, що дозволяє іншим сервісам взаємодіяти з ним через зазначені протоколи. [25].

Основні принципи СОА включають:

1. Розподілений характер. Сервіси можуть розташовуватися на різних серверах та взаємодіяти через мережу.

2. Незалежність сервісів. Кожен сервіс може розвиватися та оновлюватися незалежно від інших. Це полегшує розвиток, тестування та вдосконалення окремих компонентів системи.

3. Інтерфейси. Сервіси спілкуються за допомогою чітко визначених інтерфейсів, що зменшує залежність між ними.

4. Повторне використання. Завдяки розбиттю програмного забезпечення на окремі сервіси, їх можна повторно використовувати в інших проектах або в різних частинах одного проекту.

5. Легкість масштабування. Систему можна легко масштабувати, додаючи або вилучаючи окремі сервіси за необхідності.

6. Управління життєвим циклом. СОА дозволяє керувати життєвим циклом різних сервісів незалежно, включаючи їх розгортання, моніторинг та відмову.

Використання СОА дозволяє покращити гнучкість системи, полегшити розвиток та управління, а також сприяє високій повторній використовуваності коду. Цей підхід особливо корисний у великих та складних програмних проектах, де різні частини системи можуть бути розроблені та підтримуватися різними командами [26].

Приклад алгоритму функціонального модулю (сервіс) зображено на рисунку 3.5.

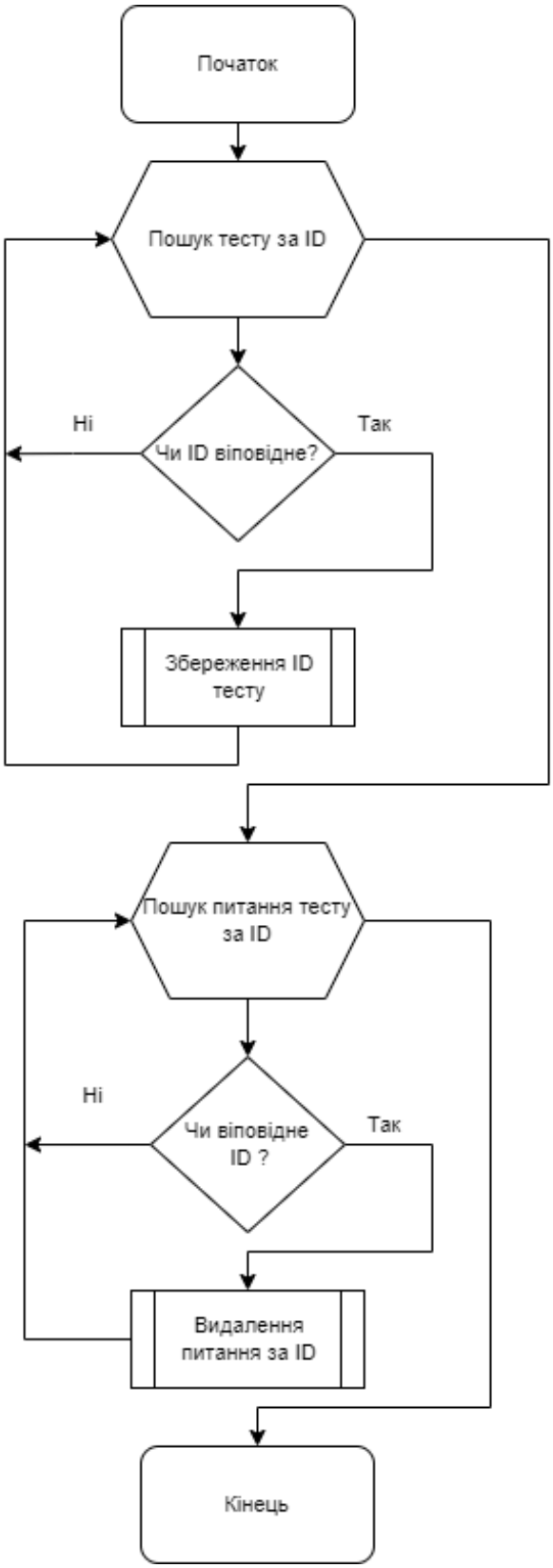
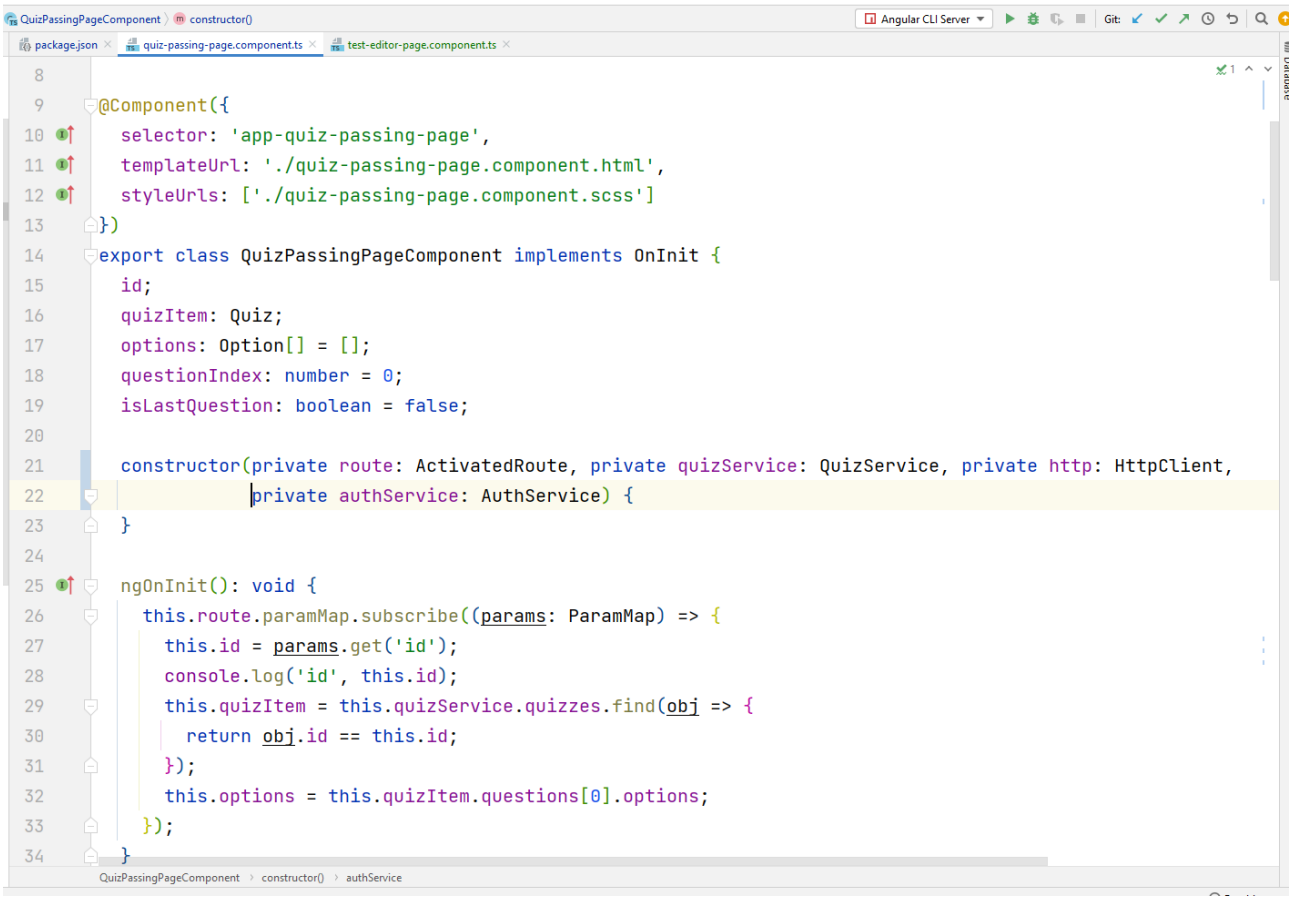


Рисунок 3.5 – Алгоритм функціонального модулю (сервіс), метод видалення питання з тесту

Також розроблено програмну компоненту для фотоконтролю під час проходження тесту. Дана компонента має функціональний та модуль візуалізації, в якій прописаний основний функціонал фотоконтролю. Програмна компонента для фотоконтролю в системі проходження тесту включає в себе функціональний та візуальний модулі, що забезпечують повноту та ефективність фотофіксації під час тестування.

Приклад функціонального модулю (компонента) зображено на рисунку 3.6.



```
@Component({
  selector: 'app-quiz-passing-page',
  templateUrl: './quiz-passing-page.component.html',
  styleUrls: ['./quiz-passing-page.component.scss']
})
export class QuizPassingPageComponent implements OnInit {
  id;
  quizItem: Quiz;
  options: Option[] = [];
  questionIndex: number = 0;
  isLastQuestion: boolean = false;

  constructor(private route: ActivatedRoute, private quizService: QuizService, private http: HttpClient,
              private authService: AuthService) {
  }

  ngOnInit(): void {
    this.route.paramMap.subscribe((params: ParamMap) => {
      this.id = params.get('id');
      console.log('id', this.id);
      this.quizItem = this.quizService.quizzes.find(obj => {
        return obj.id == this.id;
      });
      this.options = this.quizItem.questions[0].options;
    });
  }
```

Рисунок 3.6 – Функціональний модуль (компонента)

Візуальний модуль включає графічний інтерфейс, який надає користувачу інформацію та взаємодію з фотоконтролем. Це може включати в себе індикатор стану фотоконтролю, повідомлення користувача та інші елементи, що полегшують взаємодію та розуміння процесу фотофіксації під час тестування.

Візуальний модуль графічного інтерфейсу є ключовою частиною системи фотоконтролю, надаючи користувачам інтуїтивно зрозуміле та зручне середовище для взаємодії з процесом тестування. Основні елементи цього модулю включають:

Індикатор стану фотоконтролю. Графічний елемент, який вказує користувачеві поточний стан фотоконтролю. Наприклад, відображення, чи виконується фотофіксація, чи система готова до нового тесту.

Повідомлення користувача. Графічні сповіщення або повідомлення, які надають інформацію користувачу про важливі події або стани фотоконтролю. Наприклад, повідомлення про успішну фотофіксацію або про можливі помилки.

Елементи керування. Кнопки, перемикачі та інші елементи, які дозволяють користувачеві взаємодіяти з системою. Наприклад, кнопки для ініціювання фотофіксації, зупинки процесу або виклику додаткових опцій.

Візуалізація процесу фотофіксації. Графічні ефекти або анімації, які допомагають користувачеві розуміти та відстежувати хід фотофіксації. Наприклад, прогрес-смуги, анімовані іконки або інші візуальні елементи.

Меню налаштувань. Графічне меню, що дозволяє користувачам налаштовувати параметри фотоконтролю за їхніми власними потребами. Це може включати налаштування якості фотографій, вибір режимів роботи та інші опції.

Інтерактивні елементи. Графічні елементи, які реагують на взаємодію користувача, наприклад, відзначення областей торкання на сенсорному екрані або анімації під час наведення курсора.

Ці компоненти спільно створюють дружелюбний та ефективний інтерфейс для користувачів, що використовують фотоконтроль. При розробці важливо враховувати дизайн та ергономіку, щоб забезпечити оптимальний досвід користувача.

Така програмна компонента є важливою частиною системи та забезпечує надійність та об'єктивність фотоконтролю, що робить її ефективною для використання в системі тестування.

Приклад модулю візуалізації (компонента) зображено на рисунку 3.7

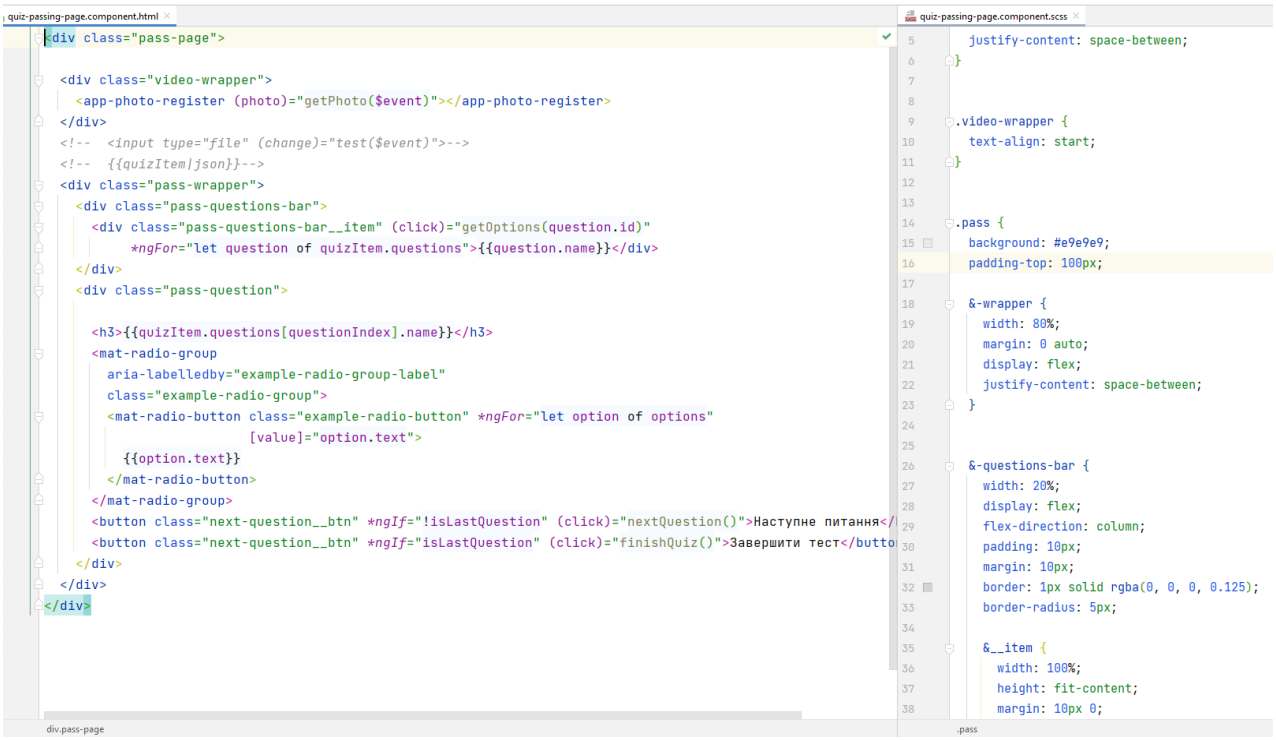


Рисунок 3.7 – Модуль візуалізації (компонента)

Функціональний модуль компоненти – це частина програмної системи або програмного продукту, яка виконує конкретну функцію чи завдання. В іншому сенсі, це невелика самостійна одиниця програмного забезпечення, яка відповідає за обробку конкретного фрагмента функціоналу. Функціональний модуль є ключовою частиною програмного забезпечення, і його основна мета полягає у виконанні конкретних завдань чи функцій. Основні характеристики функціональних модулів включають:

– Самостійність. Функціональний модуль може бути самостійним, що означає, що він може виконувати свої функції незалежно від інших модулів. Це дозволяє легко підтримувати, тестувати і розширювати систему.

– Відповідальність за конкретну функцію. Кожен функціональний модуль відповідає за обробку конкретного аспекту функціоналу системи. Це допомагає забезпечити чіткість і структурованість коду.

– Інтерфейси. Функціональний модуль може мати визначений інтерфейс для взаємодії з іншими частинами програми чи системи. Це може включати визначення вхідних та вихідних параметрів, а також способи комунікації з іншими модулями.

– Повторне використання. Функціональні модулі часто створюються з метою повторного використання. Це дозволяє їх використовувати в різних частинах системи чи навіть в інших проектах.

– Тестованість. Оскільки функціональні модулі виконують конкретні завдання, їх можна легко тестувати для перевірки правильності їх роботи.

Функціональні модулі є важливою частиною модульного програмування, де програма розбивається на менші, самостійні одиниці для полегшення розробки та обслуговування коду. Це також сприяє підтримці принципів солідності та модульності в програмному проекті.

Одним із підходів до побудови програмних систем є використання компонентної архітектури, де програма розбивається на компоненти, кожен з яких відповідає за виконання конкретної задачі або функції. Функціональний модуль є однією з таких компонент.

Приклад функціонального модулю (компонента) зображено на рисунку 3.8

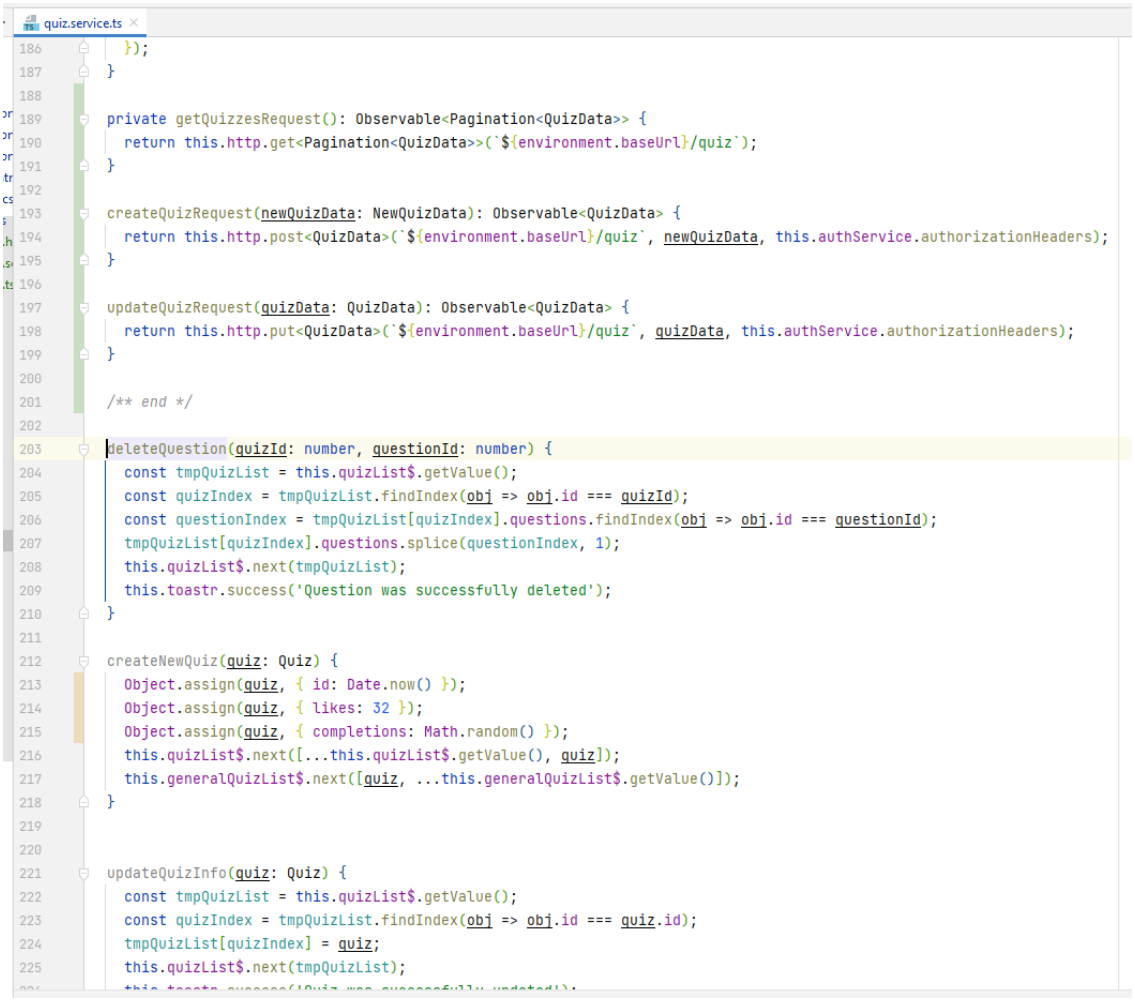


Рисунок 3.8 – Приклад функціонального модулю (сервіс)

Отже, у даному підрозділі описано програмну реалізацію веб додатку та використанні архітектурні підходи.

3.5 Висновки

Вибір мови програмування – ключовий етап у розробці програмного продукту, оскільки це визначає не тільки технічні характеристики, але й зручність розробки та підтримки коду в подальшому. У даному розділі проведено варіантний аналіз трьох популярних мов програмування: C++, Java та JavaScript/TypeScript. C++ відома своєю високою продуктивністю та широким спектром застосувань, особливо у високоефективних системах.

Однак, може вважатися менш зручною для веб-розробки та має вищий поріг входу для новачків. Java, з іншого боку, володіє великою кросплатформенністю та широким спільнотовим сприйняттям. Проте, для деяких завдань вона може виявитися занадто об'ємною та має високі вимоги до пам'яті. JavaScript/TypeScript надають багатий інструментарій для веб-розробки та дозволяють легко взаємодіяти з веб-елементами. TypeScript, як надмножина JavaScript, додає статичну типізацію, що полегшує роботу з великими проектами. Обрана мова підходить для наших потреб через свою широку популярність, легкість використання та швидкість розробки.

Після визначення мови програмування, ми дослідили ринок інтегрованих середовищ розробки (IDE) для JavaScript/TypeScript. Два основних варіанти, які були проаналізовані, це WebStorm та Visual Studio Code.

WebStorm від JetBrains є потужним та повнофункціональним інструментом, спеціалізованим на роботі з веб-технологіями. Він має багато вбудованих інструментів для підтримки JavaScript/TypeScript, а також велику кількість розширень. Visual Studio Code, розроблений Microsoft, є легким та розширюваним редактором коду, який також має потужні можливості для розробки на JavaScript/TypeScript. Він відкритий для розширень, що дозволяє вибрати лише необхідні інструменти. Обрано Webstorm через його широкий функціонал, легкість використання та активну спільноту розробників. Це забезпечить комфортну та продуктивну роботу при розробці нашого програмного продукту на основі JavaScript/TypeScript.

У даному розділі проведено варіантний аналіз і обґрунтування вибору мови програмування. Проаналізовано переваги й недоліки C++, Java та JavaScript/TypeScript. Для розробки програмного продукту обрано мову JavaScript/TypeScript. Досліджено ринок популярних IDE для цієї мови програмування, проаналізовано середовища розробки WebStorm та Visual Studio Code.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Тестування програмного забезпечення

Тестування програмного забезпечення (ПЗ) є важливою складовою розробки програм, спрямованою на забезпечення їхньої якості та надійності. Цей процес включає в себе ряд дій і стратегій для перевірки відповідності програмних продуктів вимогам та виявлення помилок та недоліків, а також перевіряє на відповідність додатку вимогам специфікації.

Основними етапами тестування:

- планування тестування: розробка стратегії тестування, визначення обсягу та виділення ресурсів.
- створення тестових сценаріїв: розробка сценаріїв, які описують тести для різних частин програмного продукту.
- виконання тестів: запуск тестів, реєстрація результатів та аналіз відповідності очікуванням.
- виправлення помилок: якщо тести виявляють помилки, розробники виправляють їх, і процес тестування повторюється.

Основними цілями тестування можуть бути наступні:

- перевірка ПЗ на відповідність програмного продукту на виході відносно нормативних, бізнес, технічних, функціональних вимог та вимог користувачів;
- виявлення технічних помилок/багів з подальшим їх усуненням (Quality Control);
- оцінка зручності, продуктивності, безпеки, локалізації, сумісності та встановлення системи.

Різні види тестування використовуються для перевірки різних аспектів програмного продукту. Нижче наведено деталізований огляд деяких основних видів тестування.

1. Модульне тестування (Unit Testing). Це перевірка окремих модулів або компонентів програмного продукту. Кожен модуль тестується ізольовано від інших, щоб переконатися, що вони виконують свої функції правильно. Це дозволяє виявляти та виправляти помилки на ранніх етапах розробки.

2. Інтеграційне тестування (Integration Testing). На цьому етапі перевіряється взаємодія між різними модулями або компонентами. Мета полягає в тому, щоб виявити помилки в інтерфейсах та забезпечити, що різні частини системи коректно співпрацюють під час взаємодії.

3. Системне тестування (System Testing). На цьому етапі перевіряється весь програмний продукт як єдина система. Тестуються всі функції та характеристики програми, включаючи взаємодію з іншими системами та виконання в умовах, схожих на реальні.

4. Функціональне тестування (Functional Testing). Це перевірка того, чи виконує програма очікувані функції. Включає тестування введення та виведення даних, роботу алгоритмів та реакцію програми на різні сценарії введення.

5. Юзабіліті тестування (Usability Testing). Цей вид тестування оцінює, наскільки легко користувач може взаємодіяти з програмним продуктом. Оцінюються аспекти, такі як інтерфейс, навігація та загальна зручність використання.

6. Відмовостійкість (Reliability) тестування. Тестується стійкість програми до відмов, тобто її здатність продовжувати працювати правильно в умовах помилок або невірного введення.

7. Тестування продуктивності (Performance Testing. Перевірка ефективності програми, включаючи швидкість відгуку, завантаження та витрату ресурсів (пам'яті, процесорного часу).

8. Безпека (Security) тестування. Оцінка вразливостей програми та виявлення можливих механізмів атаки. Тестується захищеність від несанкціонованого доступу та збереження конфіденційності даних.

Загальний підхід до тестування:

1. Ручне та автоматизоване тестування. Деякі тести можна виконати вручну, а інші можна автоматизувати за допомогою спеціальних інструментів.
2. Цикл розробки та тестування. Тестування вбудоване в усі етапи розробки, щоб виявляти та виправляти помилки якнайраніше.
3. Постійне вдосконалення. Процес тестування є динамічним і постійно вдосконалюється на основі отриманих результатів.

Використання різних видів тестування допомагає забезпечити, що програмний продукт відповідає вимогам, є стабільним та надійним, та забезпечує задоволення користувачів.

Веб додаток – це веб-сайт, на якому розміщені сторінки з частково або повністю несформованим вмістом. Остаточно вміст сторінки сформується тільки після того, як відвідувач сайту запросить сторінку з веб-сервера. У зв’язку з тим що остаточний вміст сторінки залежить від запиту, створеного на основі дій користувача, така сторінка називається динамічною. Тому Веб додаток ще називають клієнт-серверний додаток, бо логіка додатку зосереджена на сервері, а інтернет браузер лише відповідає за відображення інформації завантаженої з сервера.

Веб-тестування – це практика тестування програмного забезпечення для тестування веб-сайтів або веб-додатків на наявність потенційних помилок. Це повне тестування веб-додатків перед тим, як опублікувати.

Веб-систему потрібно перевірити повністю від кінця до кінця, перш ніж вона почне працювати для кінцевих користувачів.

Виконуючи тестування веб-сайтів та веб додатків, організація може переконатися, що веб-система працює належним чином і може бути прийнята користувачами в режимі реального часу. Дизайн та функціональність інтерфейсу є головними елементами тестування веб-сайтів.

Тестування програмного забезпечення є необхідним етапом у розробці програмних продуктів. Воно допомагає виявити помилки та покращити якість

продукту перед його випуском на ринок. Застосування різних видів тестування та ефективне управління цим процесом є ключовими для досягнення успіху в області розробки програмного забезпечення.

Отже протестуємо роботу веб додатку загалом. Відкриємо головну сторінку веб додатку у різних браузерах а саме: Google Chrome, Microsoft Edge, щоб порівняти коректне відображення стилів.

Тестування веб-додатків в різних браузерах є важливою частиною процесу розробки для забезпечення сумісності та оптимальної роботи додатка на різних платформах. Врахування різних браузерів, таких як Google Chrome, Mozilla Firefox, Microsoft Edge, Safari та інші, дозволяє забезпечити позитивний користувацький досвід для всіх користувачів.

Головне меню додатку в браузерах зображено на рисунках 4.1 та 4.2.

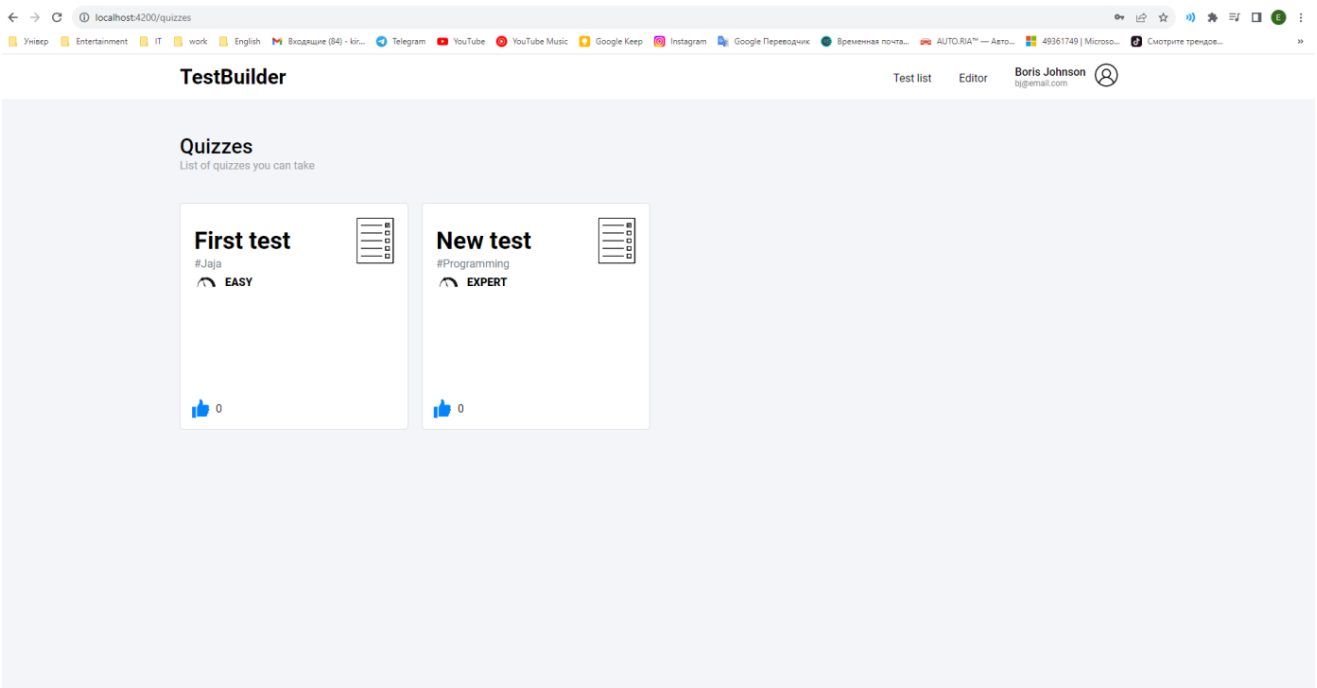


Рисунок 4.1 – Головне меню додатку в браузері Google Chrome

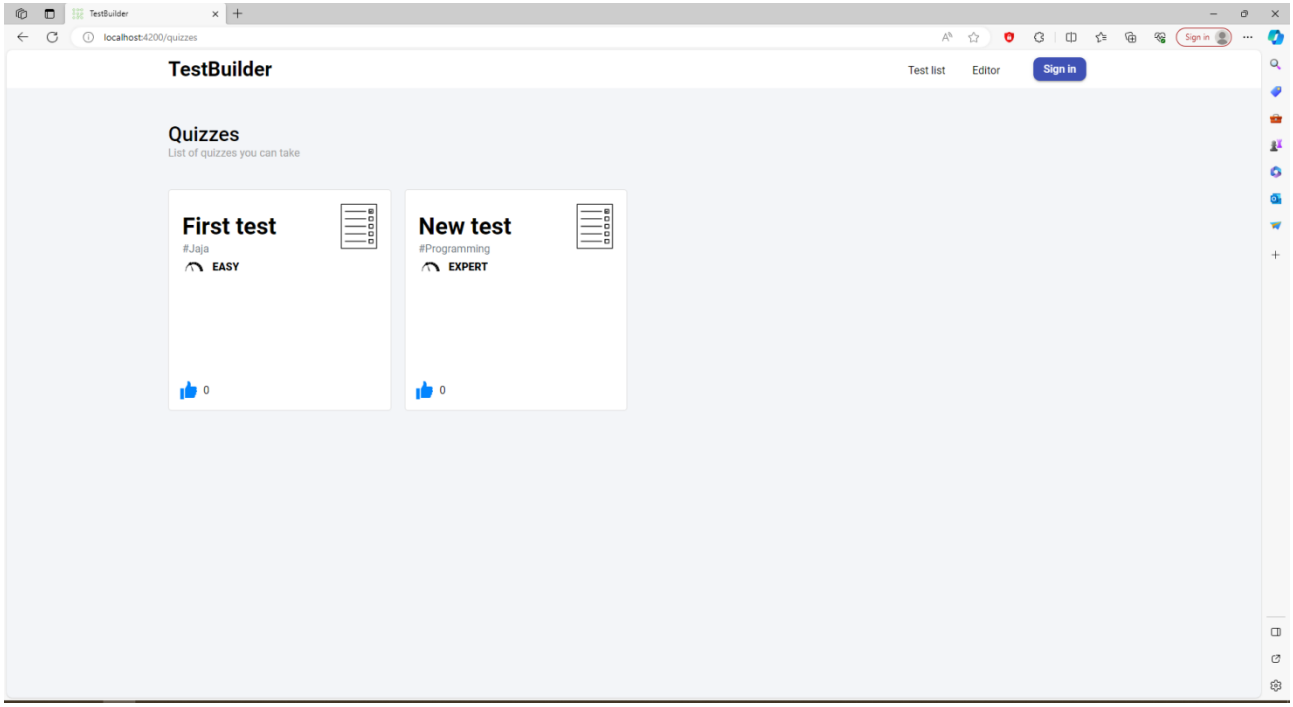


Рисунок 4.2 – Головне меню додатку в браузері Microsoft Edge

Протестуємо процес створення та редагування тестів. Розглянемо активну та не активну форму для редагування тесту рисунок 4.3 та 4.4.

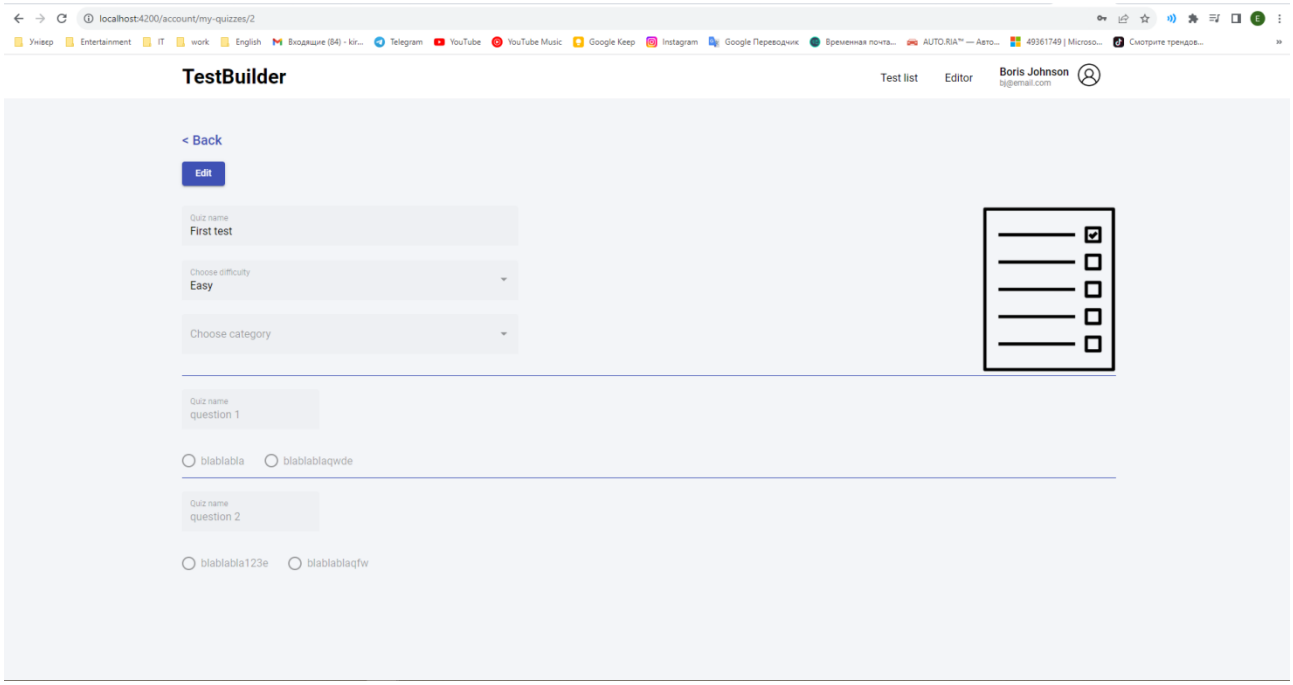


Рисунок 4.3 – Сторінка редагування тесту в неактивному стані

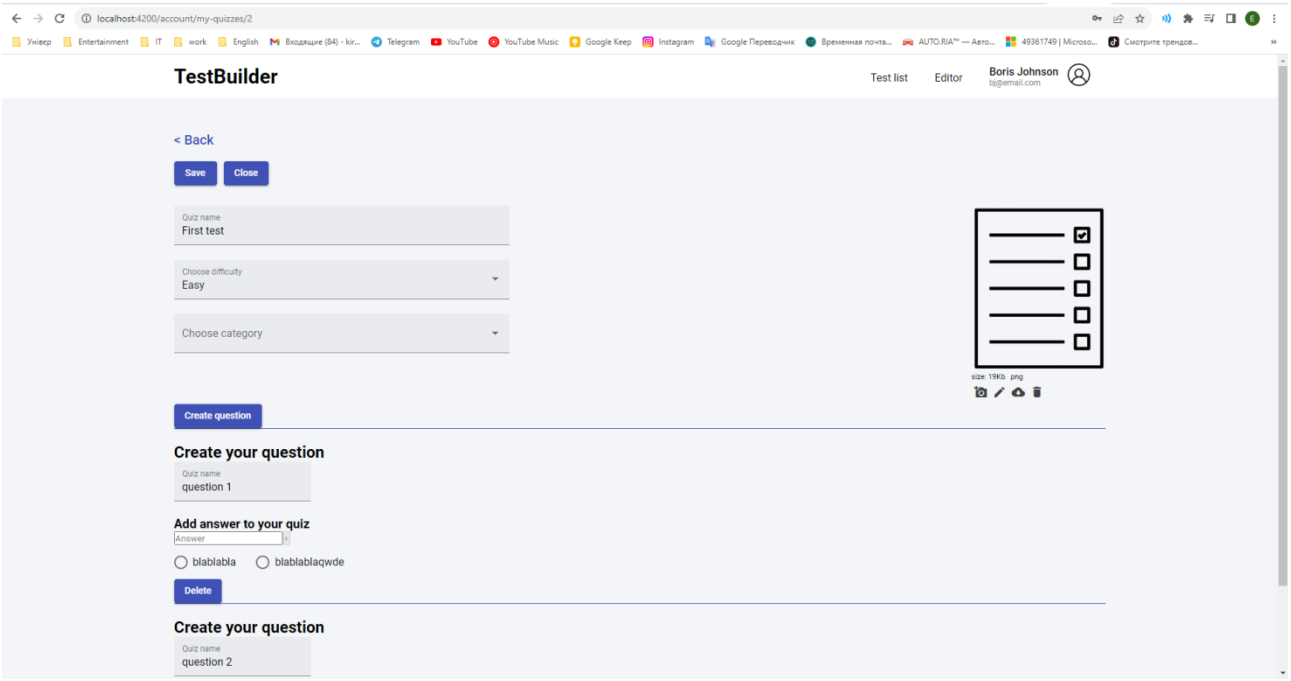


Рисунок 4.4 – Сторінка редагування тесту в активному стані

На сторінці редагування тесту в активному стані можна побачити додаткові елементи керування тестом, де можна додати або видалити питання та відповіді до нього. Також можна відредагувати сам тест, змінивши його категорію, назву, картинку та складність тесту. Додамо ще одне питання та три варіанта відповіді до нього рисунок 4.5.

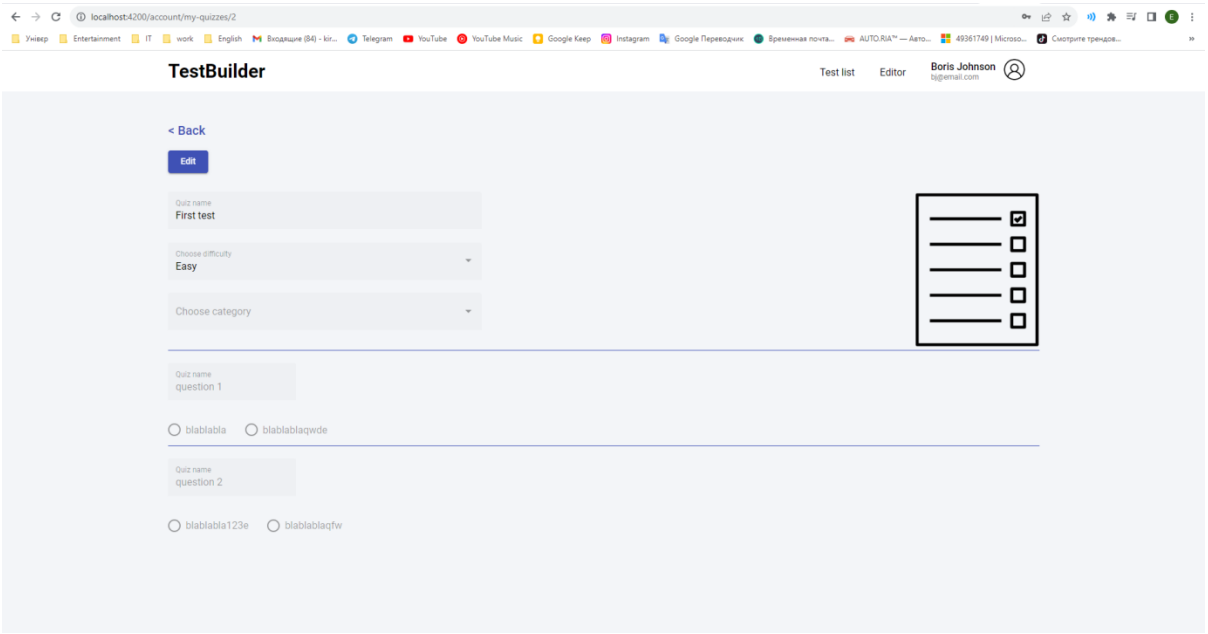


Рисунок 4.5 – Результат додання нового питання

Протестуємо створення нового тесту, додамо до нього два питання і по три відповіді до кожного та додамо тематичну картинку рисунок 3.6.

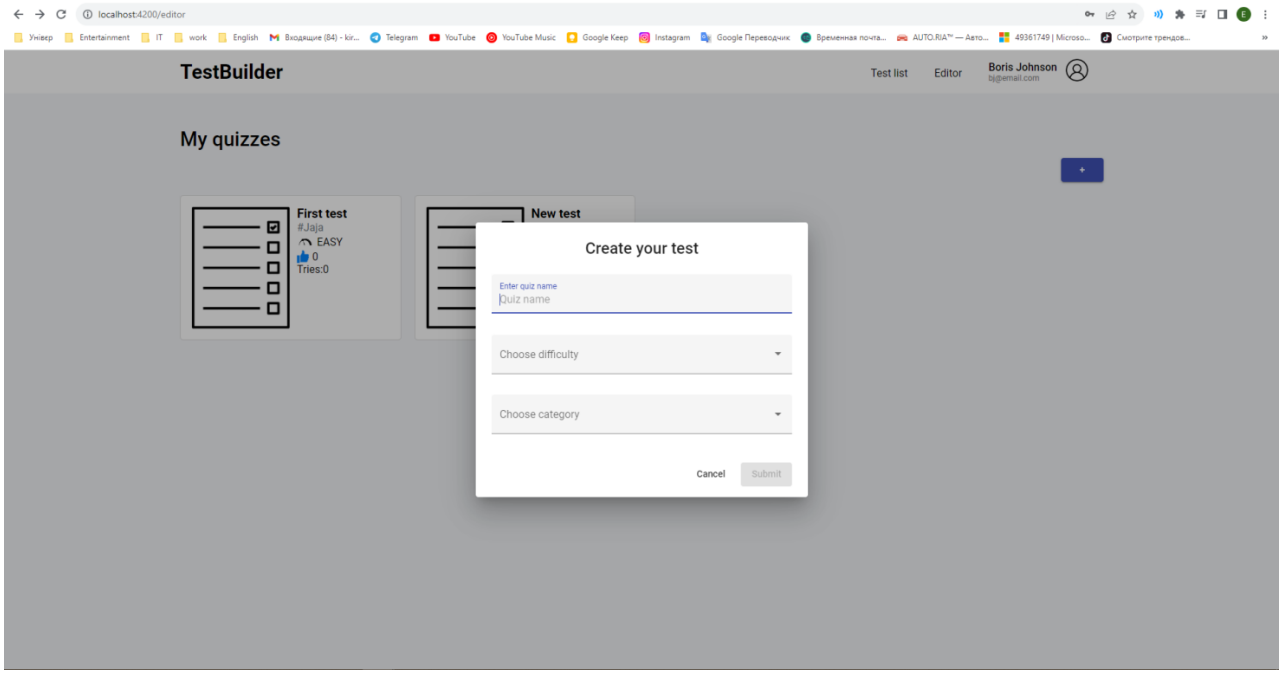


Рисунок 4.6 – Результат заповнення форми для створення тесту

Додамо питання і відповіді до нього та виберемо правильні відповіді до питання.

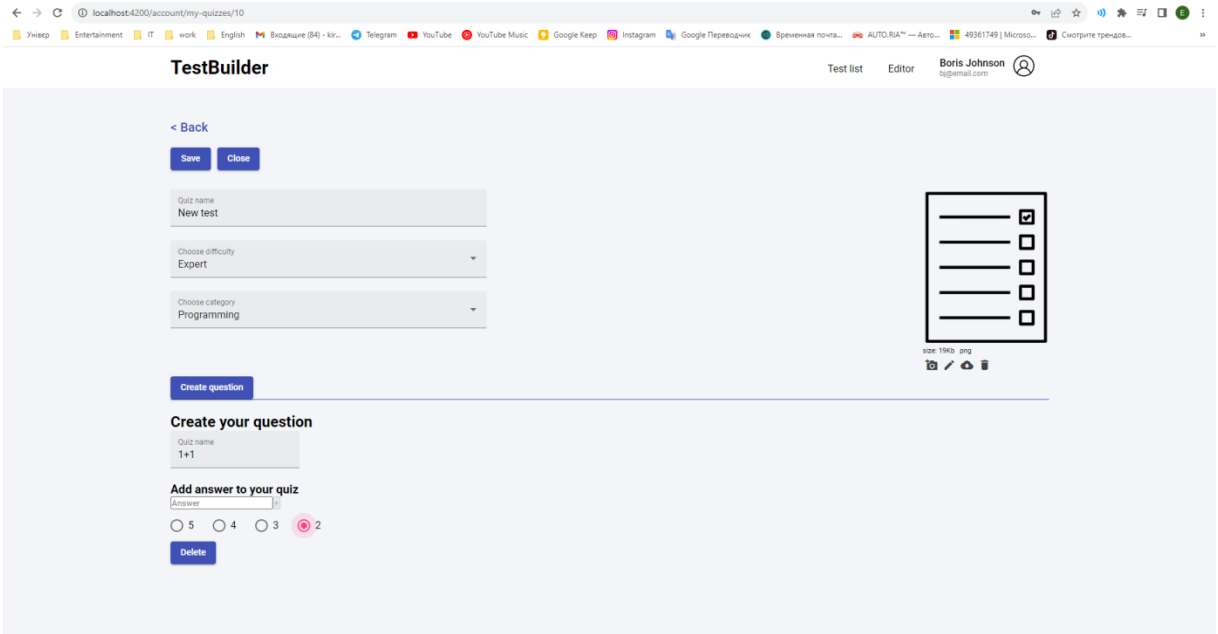


Рисунок 4.7 – Результат додання двох питань

Збережемо наш результат та побачимо повідомлення що все збережено успішно та перевіримо чи тест з’явився в основному списку тестів рисунок 4.8 та 4.9.

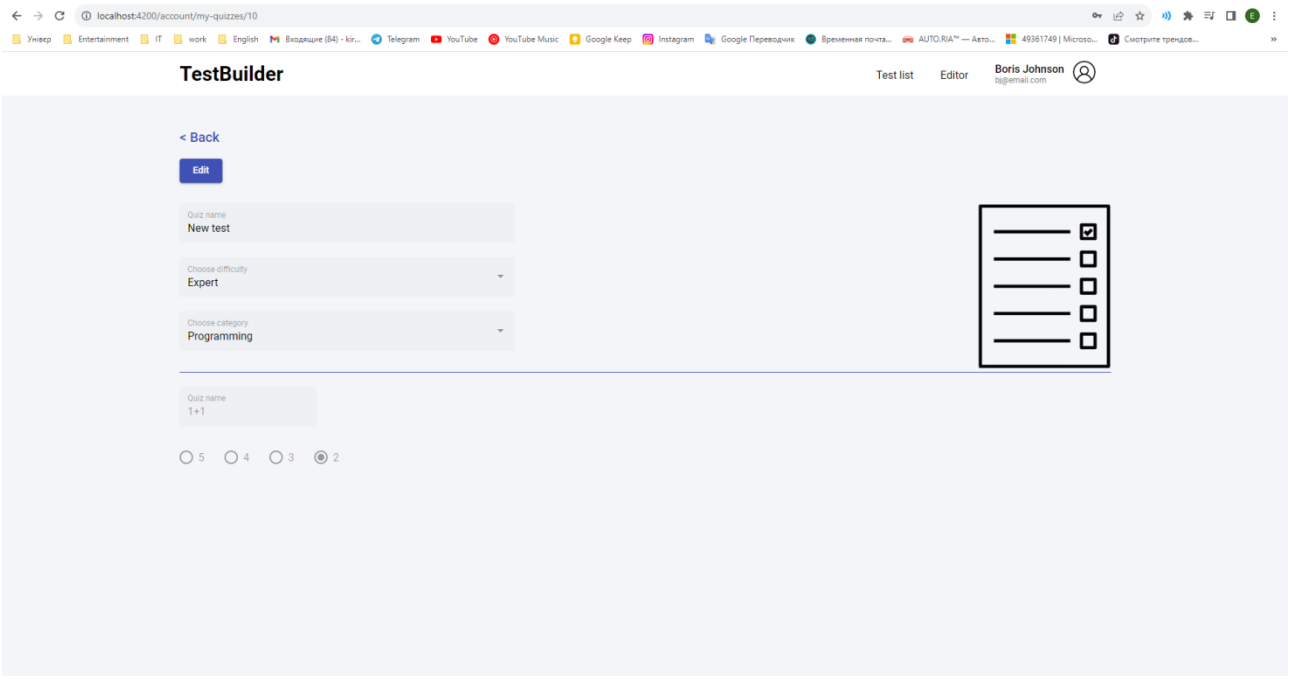


Рисунок 4.8 – Результат успішного редагування тесту

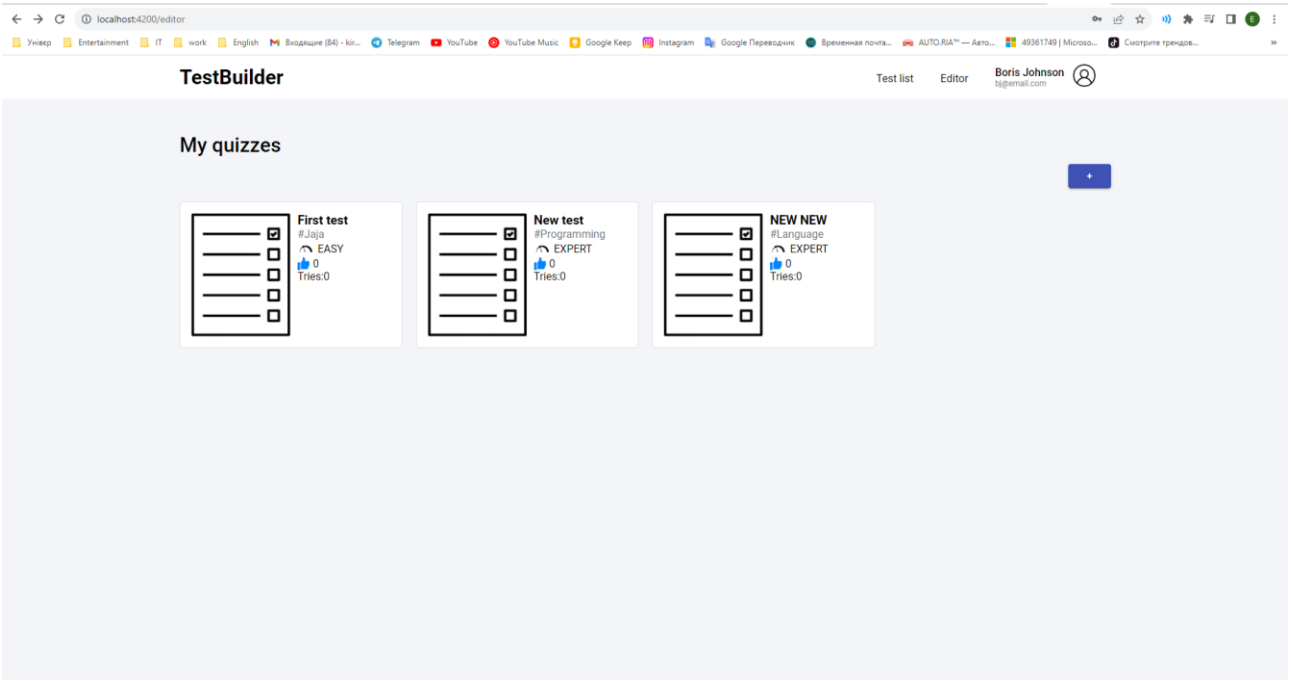


Рисунок 4.9 – Новостворений тест в основному списку тестів

Модульне тестування, іноді блочне тестування або юніт-тестування (англ. unit testing) – процес у програмуванні, що дозволяє перевірити на коректність окремі модулі вихідного коду програми, набори з одного або більше програмних модулів разом із відповідними керуючими даними, процедурами використання та обробки. Ідея полягає в тому, щоб писати тести для кожної нетривіальної функції чи методу. Це дозволяє досить швидко перевірити, чи не привела чергова зміна коду до регресії, тобто до появи помилок у вже відтестованих місцях програми, а також полегшує виявлення та усунення таких помилок. Наприклад, оновити бібліотеку, що використовується в проекті, до актуальної версії можна в будь-який момент, прогнавши тести і виявивши несумісності [27].

Також написано простий Unit тест для тестування одної з найважливіших компенет для фотофіксації. Лістинг та результат тесту зображено на рисунках 4.10 та 4.11.

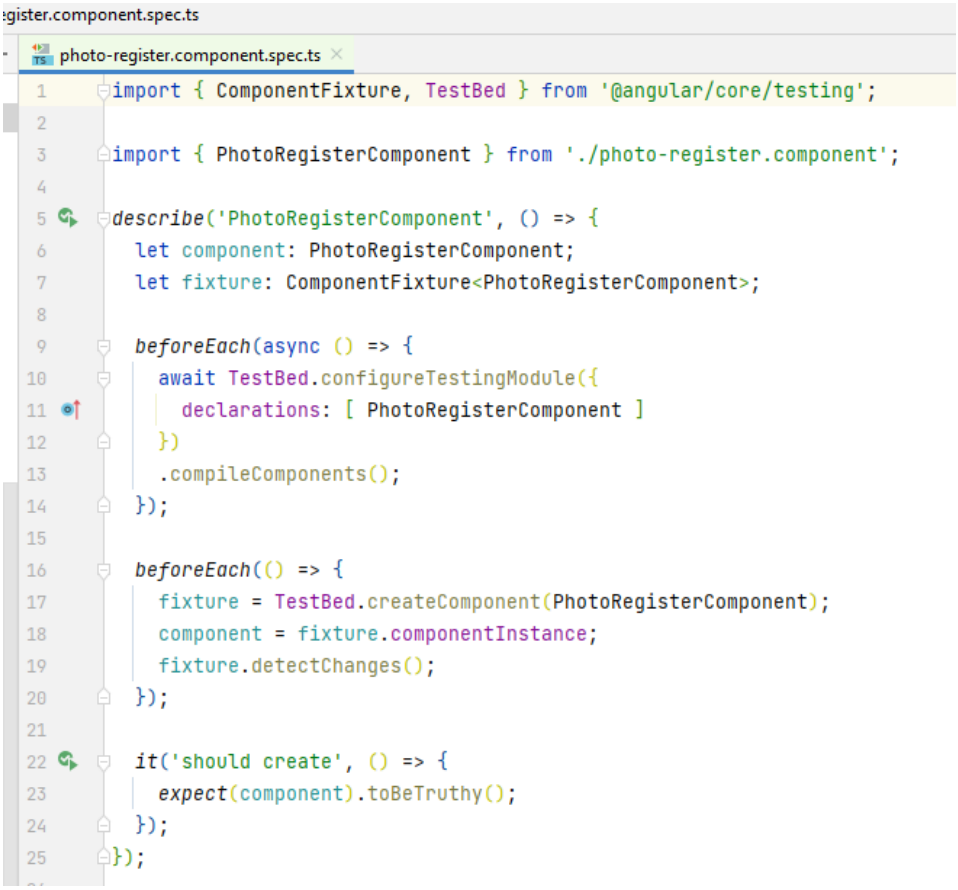


Рисунок 4.10 – Лістинг Unit тесту компоненти фотофіксації

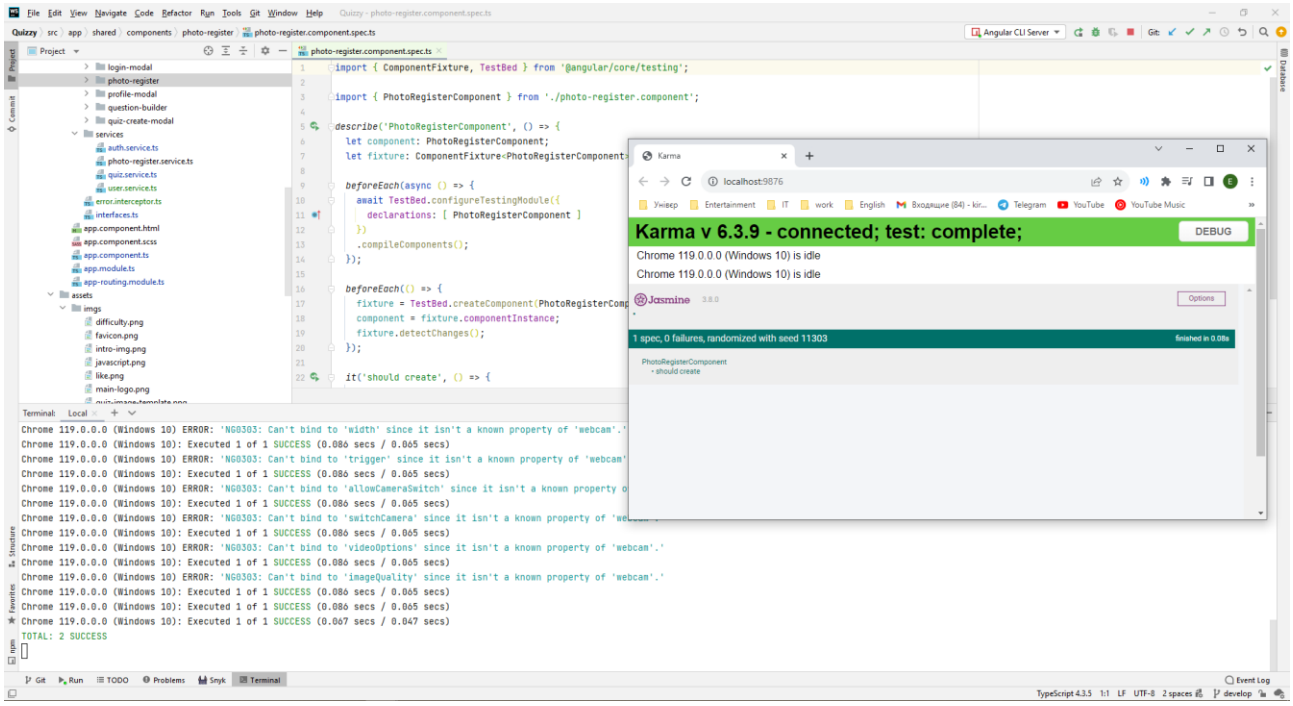


Рисунок 4.11 – Результат виконання Unit тесту компоненти фотофіксації

Отже, протестований веб додаток немає критичних багів, основний функіонал працює коректно та додаток відображається одноково в різних браузерах.

4.2 Інструкція користувача

Інструкція користувача призначена для того, щоб навчити користувача працювати з програмним додатком. Оскільки програмний веб додаток є браузерним то він не потребує інсталяції стороннього програмного забезпечення та може бути запущеним на будь якій платформі, будь-то Windows, Linux чи MacOS. Для роботи веб додатку потрібно мати тільки браузер який підтримує роботу з JavaScript [28].

Розробка інструкції користувача для програмного додатку є важливим етапом в процесі створення програмного продукту. Ось кілька причин, чому це важливо:

1) Спрощення використання – інструкції надають користувачам ясний огляд та крок за кроком роз'яснення щодо користування додатком. Це допомагає спростити процес для новачків та зменшити ймовірність помилок.

2) Ефективніше навчання – інструкції дозволяють користувачам ефективно навчатися функціоналу додатку. Вони можуть використовувати їх для швидкого вивчення основ або для глибшого розуміння складніших функцій.

3) Менше запитань від коистувачів – детальні та зрозумілі інструкції можуть зменшити кількість запитань, які користувачі можуть мати щодо використання програмного додатку. Це допомагає уникнути непорозумінь та сприяє більш гладкому користуванню продуктом.

4) Покращення вражень користувача – гарна інструкція може позитивно вплинути на враження користувачів від додатку. Користувачі, які швидко розуміють, як користуватися продуктом, ймовірніше залишаться задоволеними та вірними користувачами.

5) Забезпечення безпеки та правильного використання – для деяких програм можуть існувати певні правила безпеки чи конфіденційності, які користувач повинен дотримуватися. Інструкції можуть надати необхідну інформацію та попередження для запобігання можливих проблем.

6) Зменшення витрат на підтримку – якщо користувачі можуть знайти відповіді на свої питання у власному розумінні за допомогою інструкцій, це може знизити необхідність великої кількості звернень до служби підтримки.

Узагальнюючи, розробка якісної інструкції користувача є важливим кроком для забезпечення успішного впровадження програмного додатку та задоволення користувачів.

Запустивши веб додаток, відкривається головна сторінка, на якій зображено список доступних тестів . Приклад головної сторінки веб додатку для проходження тестів з фотоконтролем зображено на рисунку 4.12.

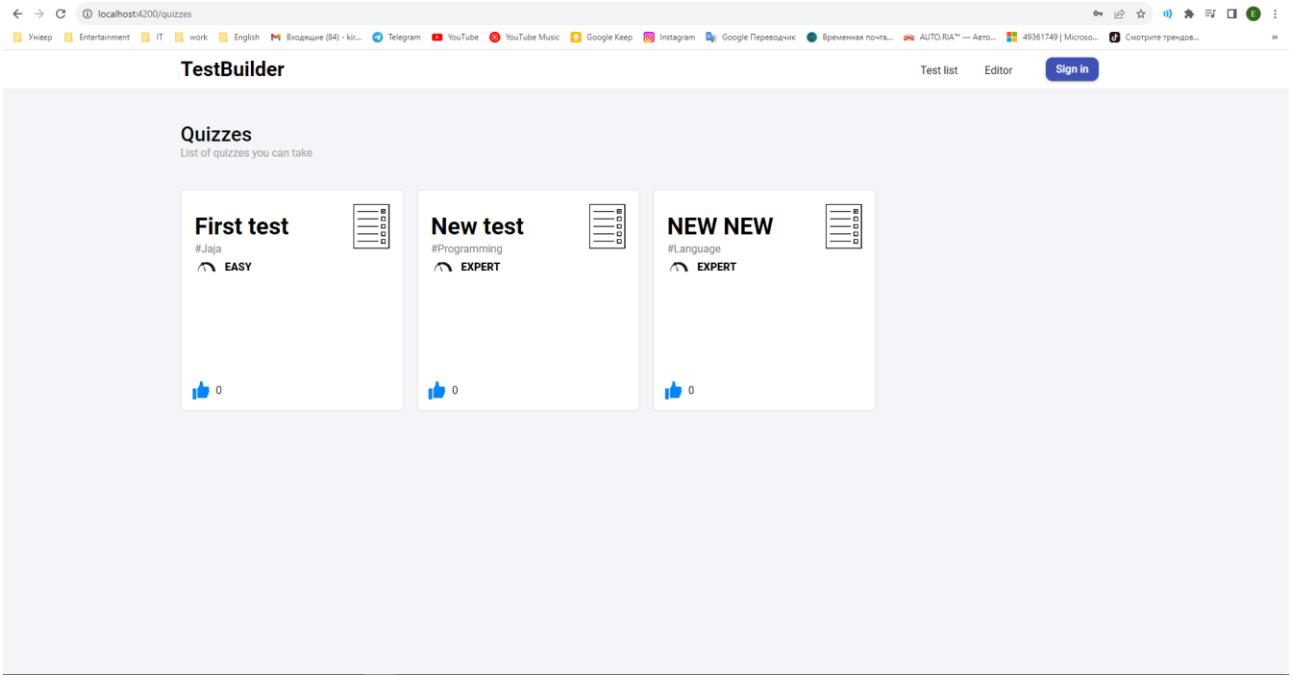


Рисунок 4.11 – Головна сторінка веб додатку

Невід'ємною частиною веб додатку з персоналізацією інформації слугує реєстрація користувача та його вхід в додаток з персональним обліковим записом. Перейшовши на екран авторизації, користувач ввівши свої дані, може потрапити до свого профілю. На рисунках 4.13 та 4.14 зображено екрани авторизації користувача та список ним створених тестів.

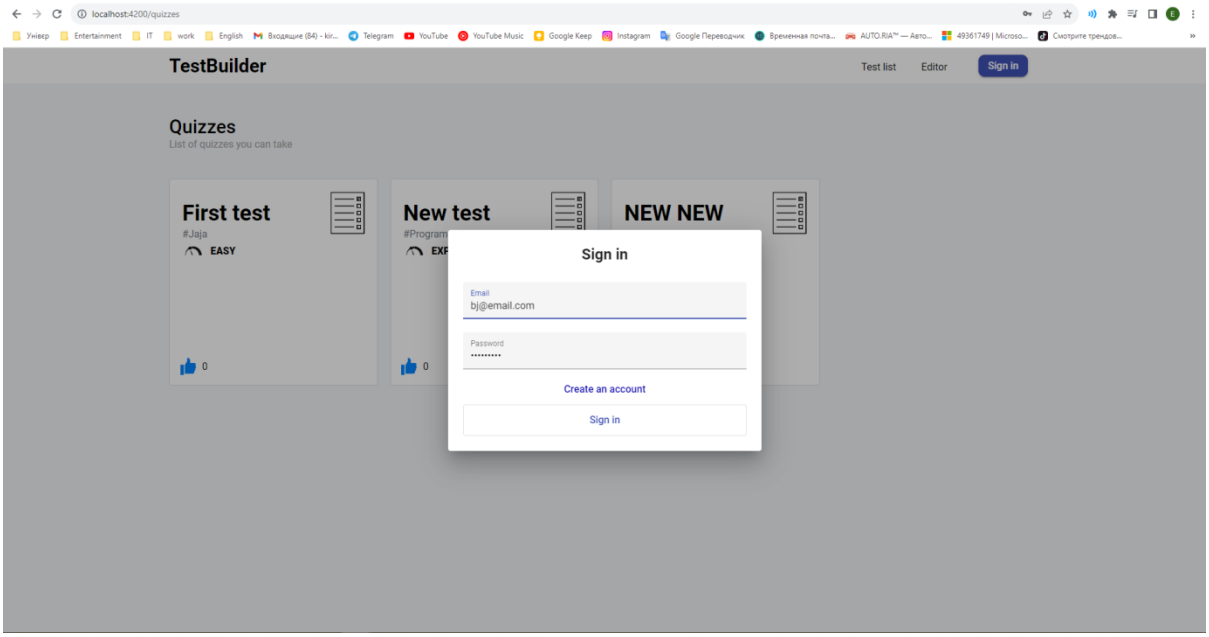


Рисунок 4.12 – Екран входу в персональний обліковий запис

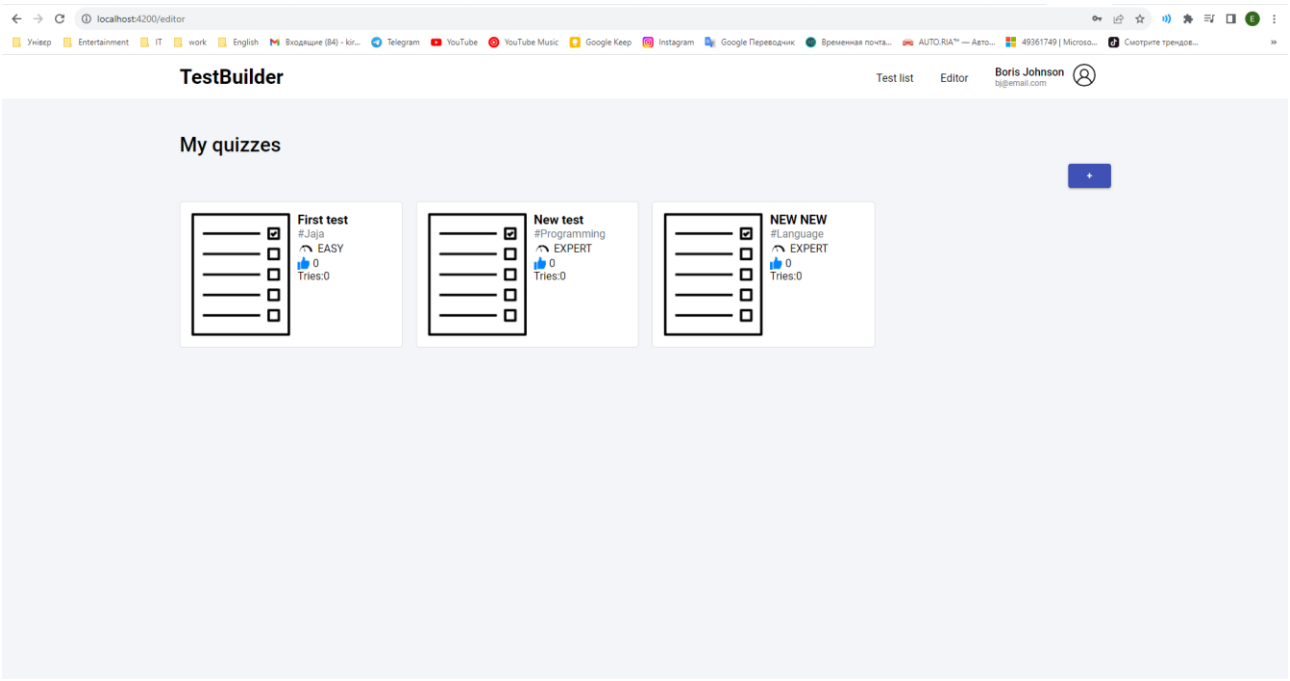


Рисунок 4.13 – Екран реєстрації персонального облікового запису

Клікнувши на тест, користувач потрапить на сторінку редагування тесту, де він зможе видалити питання, додати нові, чи просто відредагувати їх. Приклад екрану редагування тесту зображено на рисунку 4.15.

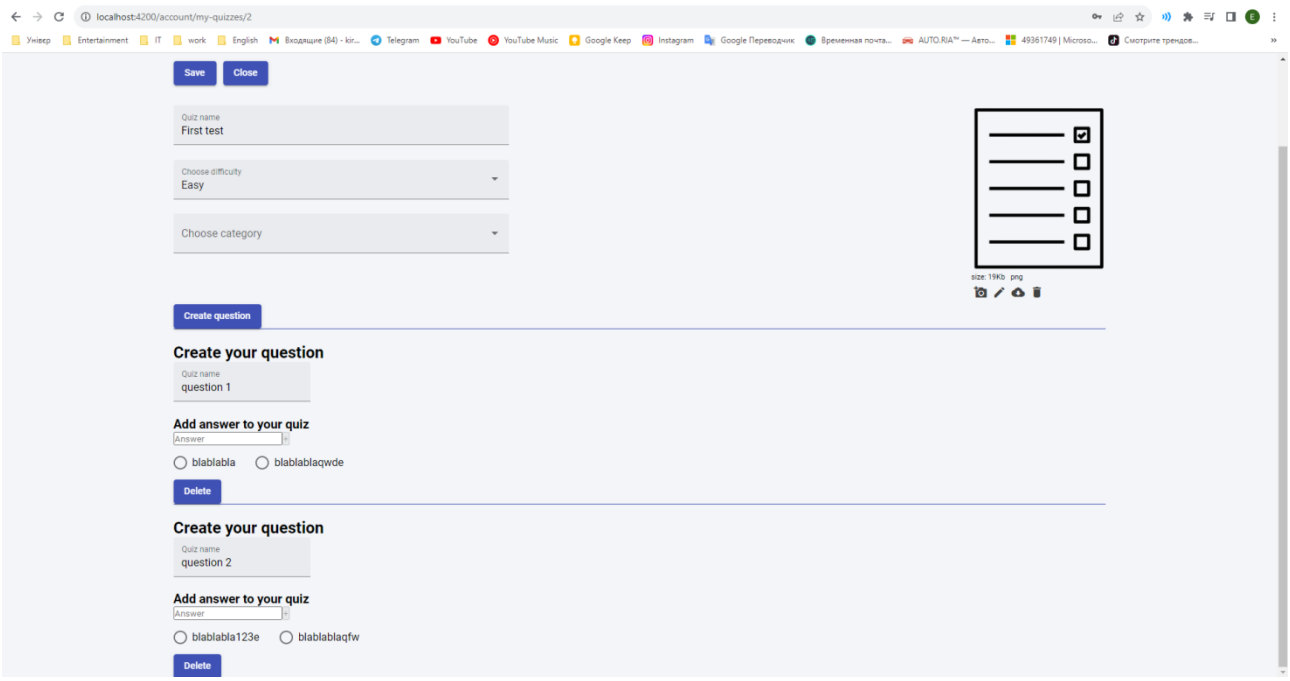


Рисунок 4.14 – Екран редагування тесту

Також на екрані персонального облікового запису, можна створити новий тест натиснувши кнопку «Створити тест», після чого відкриється модальне вікно у якому потрібно вказати назву тесту, його категорію, складність та зображення за бажанням. Приклад такого вікна зображено на рисунку 4.15.

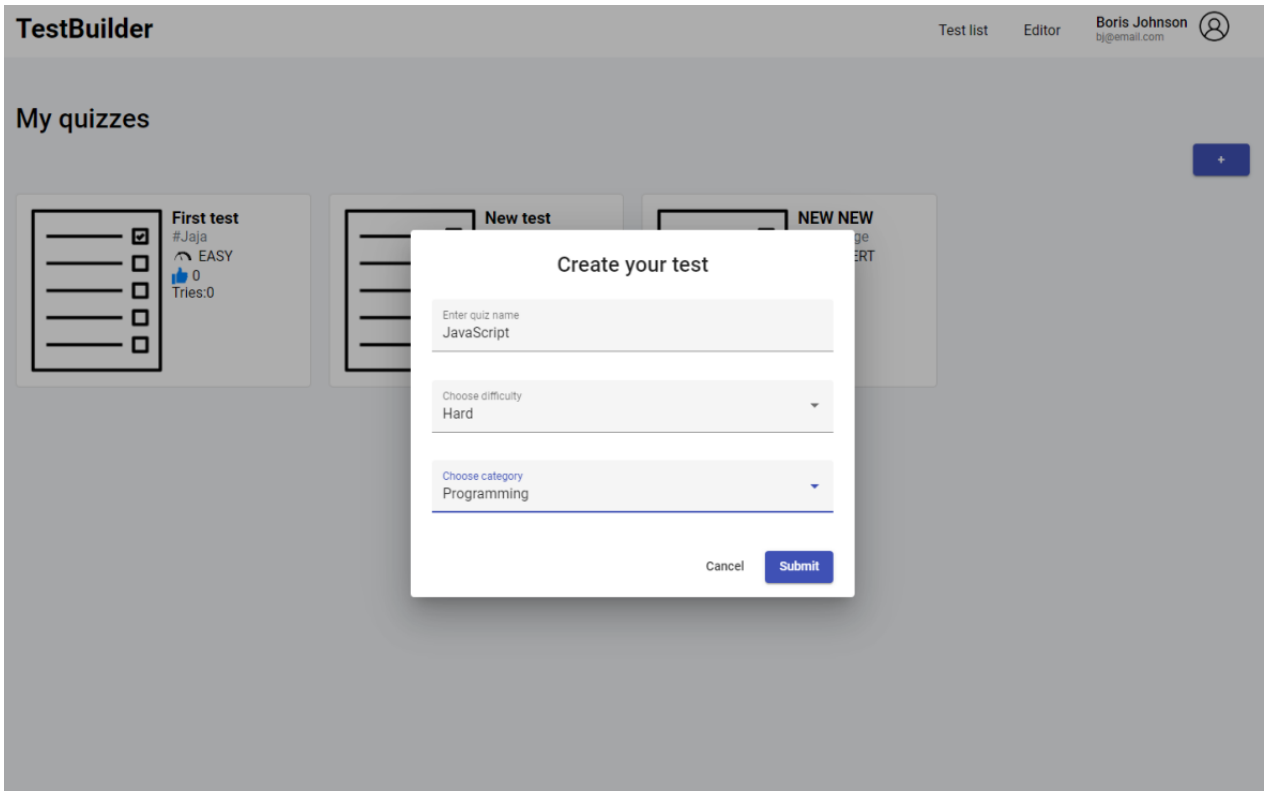


Рисунок 4.15 – Модальне вікно для створення тесту

Отже, розроблено інструкцію користувача, описано основний функціонал, а саме послідовність дій які потрібно для створення та редагування тесту. Також наведено зображення для більш зрозумілого використання веб додатку.

4.3 Висновки

У даному розділі проведено тестування програмного додатку для проходження тестів та створена інструкція для користувача веб додатку.

Тестування показало, що веб додаток розроблено відповідно до технічних вимог та немає критичних помилок. Відображення веб застосунку коректне в різних браузерах та основний функціонал по додаванню, редагуванню тесту працює добре та критичних помилок немає.

Додатково, під час тестування виявлено, що швидкість відгуку веб-додатку на дії користувача є задовільною, а завантаження сторінок відбувається швидко і без затримок. Також, функціонал з автентифікації користувачів та забезпечення їх конфіденційності працює належним чином.

Інтерфейс веб-додатку виглядає привабливо та інтуїтивно зрозуміло, що полегшує користування ним. Користувачеві надано необхідні інструменти та опції для ефективного проведення тестів, а також зручний доступ до результатів та статистики.

Документація та інструкція для користувача веб-додатку є чіткою та зрозумілою. Вона містить достатньо інформації для того, щоб новий користувач міг легко розібратися з основними функціями та можливостями додатку.

Загальний висновок з тестування вказує на те, що програмний додаток готовий до використання та відповідає всім вимогам, визначеним у специфікації. Розробка та тестування додатку проведено на високому рівні, і він готовий до впровадження в робоче середовище.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Науково-технічна розробка має право на існування та впровадження, якщо вона відповідає вимогам часу, як в напрямку науково-технічного прогресу та і в плані економіки. Тому для науково-дослідної роботи необхідно оцінювати економічну ефективність результатів виконаної роботи [30].

Магістерська кваліфікаційна робота за темою «Моделі та програмні засоби дистанційного тестування з використанням технології Google Cloud Vision. Частина 2. Модуль клієнта» відноситься до науково-технічних робіт, які орієнтовані на виведення на ринок (або рішення про виведення науково-технічної розробки на ринок може бути прийнято у процесі проведення самої роботи), тобто коли відбувається так звана комерціалізація науково-технічної розробки. Цей напрямок є пріоритетним, оскільки результатами розробки можуть користуватися інші споживачі, отримуючи при цьому певний економічний ефект. Але для цього потрібно знайти потенційного інвестора, який би взявся за реалізацію цього проекту і переконати його в економічній доцільності такого кроку.

Для наведеного випадку нами мають бути виконані такі етапи робіт:

1) проведено комерційний аудит науково-технічної розробки, тобто встановлення її науково-технічного рівня та комерційного потенціалу;

2) розраховано витрати на здійснення науково-технічної розробки;

3) розрахована економічна ефективність науково-технічної розробки у випадку її впровадження і комерціалізації потенційним інвестором і проведено обґрунтування економічної доцільності комерціалізації потенційним інвестором.

Метою проведення комерційного і технологічного аудиту дослідження за темою «Моделі та програмні засоби дистанційного тестування з використанням технології Google Cloud Vision. Частина 2. Модуль клієнта» є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 5.1 [30].

Таблиця 5.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів

Продовження таблиці 5.1

Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуютьс я у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці.

Таблиця 5.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	4	3	4
2. Ринкові переваги (наявність аналогів)	3	3	4
3. Ринкові переваги (ціна продукту)	4	4	3
4. Ринкові переваги (технічні властивості)	3	3	4
5. Ринкові переваги (експлуатаційні витрати)	3	3	3
6. Ринкові перспективи (розмір ринку)	3	4	3
7. Ринкові перспективи (конкуренція)	3	3	4
8. Практична здійсненність (наявність фахівців)	3	3	3
9. Практична здійсненність (наявність фінансів)	3	3	4
10. Практична здійсненність (необхідність нових матеріалів)	3	3	3
11. Практична здійсненність (термін реалізації)	3	3	3
12. Практична здійсненність (розробка документів)	3	3	3
Сума балів	38	38	41
Середньоарифметична сума балів $СБ_с$	39		

За результатами розрахунків, наведених в таблиці 5.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 5.3 [31].

Таблиця 5.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ , розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Моделі та програмні засоби дистанційного тестування з використанням технології Google Cloud Vision. Частина 2. Модуль клієнта» становить 39 бали, що, відповідно до таблиці 5.3, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

5.2 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Моделі та програмні засоби дистанційного тестування з використанням технології Google Cloud Vision. Частина 2. Модуль клієнта», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

5.2.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками,

тарифними ставками згідно з чинними в організаціях системами оплати праці [31].

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою [31]:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \quad (5.1)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дні;

T_p – середнє число робочих днів в місяці, T_p=21 день.

$$З_о = 25000{,}00 \cdot 65 / 21 = 77380{,}95 \text{ грн.}$$

Проведені розрахунки зведемо до таблиці.

Таблиця 5.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	25000	1190,48	65	77380,95
Інженер-розробник програмного забезпечення	22000	1047,62	65	68095,24
Консультант	15000	714,29	65	46428,57
Всього				191904,76

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР розраховуємо за формулою:

$$З_p = \sum_{i=1}^{n} C_i \cdot t_i \,, \tag{5.2}$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}} \,, \tag{5.3}$$

де M_M – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо $Мм$=6700,00 грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б) [31];

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об’єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

T_p – середнє число робочих днів в місяці, приблизно T_p = 22 дні;

$t_{зм}$ – тривалість зміни, год.

$C_1 = 6700{,}00 \cdot 1{,}10 \cdot 1{,}65 / (21 \cdot 8) = 72{,}38$ грн.

$З_{р1} = 72{,}38 \cdot 10{,}00 = 723{,}84$ грн.

Таблиця 5.5 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Встановлення допоміжного обладнання	10	2	1,1	72,38	723,84
Підготовка робочого місця дослідника	4	2	1,1	72,38	289,54
Інсталяція програмного забезпечення	6	5	1,7	111,87	671,20
Всього					1684,57

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (5.4)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 12%.

$З_{дод} = (191904,76 + 1684,57) \cdot 12 / 100\% = 23230,72$ грн.

5.2.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (5.5)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

Зн = (191904,76+ 1684,57+ 23230,72) · 22 / 100% = 47700,41грн.

5.2.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень.

Витрати на матеріали (*М*), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$М = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{вj}\ , \quad (5.6)$$

де $Н_j$ – норма витрат матеріалу *j*-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу *j*-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 ... 1,15*);

$В_j$ – маса відходів *j*-го найменування, кг;

$Ц_{вj}$ – вартість відходів *j*-го найменування, грн/кг.

$М_1$ = 2 · 190,00 · 1,1 - 0,000 · 0,00 = 418 грн.

Проведені розрахунки зведемо до таблиці.

Таблиця 5.6 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за од	Норма витрат, од	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн

Продовження таблиці 5.6

Папір канцелярський офісний	190	2	0	0	418
Канцелярське приладдя (набір офісного працівника)	150	3	0	0	495
Всього					913

5.2.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Моделі та програмні засоби дистанційного тестування з використанням технології Google Cloud Vision. Частина 2. Модуль клієнта» відсутні.

5.2.5 Спецустаткування для наукових робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення в даному дослідженні не використовувалось.

5.2.6 Програмне забезпечення для наукових робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних)

необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (5.7)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань програмних засобів.

$B_{прг} = 6000 \cdot 1 \cdot 1,1 = 6720$ грн.

Отримані результати зведемо до таблиці:

Таблиця 5.7– Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
IntelliJ IDEA Ultimate	1	2276	2549,12
ОС Windows 10	1	6000	6720
Прикладний пакет Microsoft Office 2019	1	5000	5600
Всього			14869,12

5.2.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (5.8)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн; $t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців; $T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років. *$A_{обл}$ = (50000,00 · 3) / (5 · 12) = 2500грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.8 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Ноутбук	50 000	5	3	2500,00
Монітор	10 000	5	3	500,00
Роутер	2 500	2	3	312,50
Робоче місце дослідника	20000	5	3	1000,00
Оргтехніка	7000	4	3	437,50
Приміщення лабораторії	250000	20	3	3125,00
ОС Windows 10	6720	3	3	560,00
Прикладний пакет Microsoft Office 2019	5600	3	3	466,67
				8901,67

5.2.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \quad (5.9)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо Ц$_e$ = 7,50 грн;

$K_{вni}$ – коефіцієнт, що враховує використання потужності, $K_{вni} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$B_e = 0{,}3 \cdot 520{,}0 \cdot 7{,}50 \cdot 0{,}95 / 0{,}97 = 1145{,}88$ грн.

Проведені розрахунки зведемо до таблиці.

Таблиця 5.9 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Ноутбук	0,3	520	1145,88
Монітор	0,1	520	381,96
Роутер	0,05	520	190,98
Робоче місце дослідника	0,15	520	572,94
Оргтехніка	0,45	10	33,05
Всього			2324,81

5.2.9 Службові відрядження

До статті «Службові відрядження» дослідної роботи належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов'язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з'їзди, конференції, наради, пов'язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{св} = (З_о + З_р) \cdot \frac{Н_{св}}{100\%}, \qquad (5.10)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св}$ = 20%.

$B_{св}$ = (191904,76+ 1684,57) · 20 / 100% = 38717,87грн.

5.2.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуємо як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%}, \qquad (5.11)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 35%.

$B_{сп}$ = (191904,76+ 1684,57) · 35 / 100% = 67756,27 грн.

5.2.11 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (5.12)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 50%.

$I_в$ = (191904,76+ 1684,57) · 50 / 100% = 96794,67грн.

5.2.12 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (5.13)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 120%.

$В_{нзв}$ = (191904,76+ 1684,57) · 120 / 100% = 232 307,20грн.

Витрати на проведення науково-дослідної роботи розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}. \quad (5.14)$$

$B_{заг}$=*191904,76+1684,57+23230,72+47700,41+913+0,00+0+14869,12+ 8901,67+ 2324,81+38717,87+67756,27+96794,67+232 307,20= 727105,06грн.*

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta}, \quad (5.15)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,7.

ЗВ = 727105,06/ 0,7 = 1038721,51грн.

5.3 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження передбачають комерціалізацію протягом 3-х років реалізації на ринку.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

ΔN – збільшення кількості споживачів продукту, у періоди часу, що аналізуються, від покращення його певних характеристик;

1-й рік – 2000 користувачів;

2-й рік – 3000 користувачів;

3-й рік – 1500 користувачів.

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки, приймемо 2500 користувачів;

$Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки, приймемо 105 000 грн;

$\pm\Delta Ц_о$ – зміна вартості програмного продукту від впровадження результатів науково-технічної розробки, приймемо 30000,00 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 3-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [31]:

$$\Delta П_i = (\pm\Delta Ц_о \cdot N + Ц_о \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (5.16)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2023 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0{,}8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту. Приймемо ρ =30%;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2023 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$\Delta П_1 = (30000 \cdot 2500{,}00 + 135000 \cdot 2000) \cdot 0{,}83 \cdot 0{,}3 \cdot (1\text{-}0{,}18/100\%) =$ 70696710 грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2 = (30000 \cdot 2500{,}00 + 135000 \cdot (2000 + 3000)) \cdot 0{,}83 \cdot 0{,}3 \cdot (1\text{-}0{,}18/100\%) =$ 153688500 грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3 = (30000 \cdot 2500{,}00 + 135000 \cdot (2000+3000+1500)) \cdot 0{,}83 \cdot 0{,}3 \cdot (1-0{,}18/100\%) =$ 195184395 грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (5.17)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,25;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$ПП = 70696710/(1+0{,}25)^1 + 153688500/(1+0{,}25)^2 + 195184395/(1+0{,}25)^3 =$ 254852418,24грн.

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \qquad (5.18)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$ =4;

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 1038721,51грн.

$PV = k_{інв} \cdot ЗВ = 4 \cdot$ 1038721,51= 4154886,048грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \tag{5.19}$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 254852418,24грн;

PV – теперішня вартість початкових інвестицій, 4154886,048грн.

$E_{абс} = ПП - PV$ = 254852418,24- 4154886,048= 250697532,19грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{5.20}$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 250697532,19грн;

PV – теперішня вартість початкових інвестицій, 4154886,048грн;

$T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 3 роки.

$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1$ = (1+250697532,19/ 4154886,048)$^{1/3}$= 2,9.

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f\ , \tag{5.21}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d =0,1;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,25.

$\tau_{мін}$ = 0,11+0,25 = 0,36 < 1,85 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Моделі та програмні засоби дистанційного тестування з використанням технології Google Cloud Vision. Частина 2. Модуль клієнта» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}\ , \tag{5.22}$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 2,9 = 0,34 року.

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

5.4 Висновок

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Моделі та програмні засоби дистанційного тестування з використанням технології Google Cloud Vision. Частина 2. Модуль клієнта» становить 39 балів, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу вище середнього).

Також термін окупності становить 0,34 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Моделі та програмні засоби дистанційного тестування з використанням технології Google Cloud Vision. Частина 2. Модуль клієнта».

ВИСНОВКИ

У магістерській кваліфікаційній роботі розроблено програмний веб додаток, який призначений для проходження тестів з фото контролем. Розроблено такі методи: дистанційного тестування, у якому на відміну від існуючих, додано функції отримання та передачі на сервер результатів фотоконтролю, що дозволило підвищити достовірність тестування на 10 % та метод побудови архітектури систем дистанційного тестування, у якому, на відміну від існуючих, використано архітектуру типу Rich client для тестування, що зменшує навантаження на сервер та економить трафік.

1) Проаналізовано дану проблему та виявлено доцільність розробки такого програмного веб додатку. Розглянуто основні аналоги програмних веб застосунків, визначено їх недоліки та переваги, розроблено таблицю для порівняння з розроблюваним програмним веб додатком.

2) Проаналізовано мови програмування та обрано TypeScript та фреймворк Angular. Додаток використовує TypeScript та фреймворк Angular, що забезпечує високу продуктивність та зручність в розробці та підтримці. Вибір цих технологій обґрунтований їхньою популярністю, широким спільнотовим сприйняттям та можливістю легкої інтеграції з іншими рішеннями.

3) Перед створенням програмного продукту розроблено схеми загального алгоритму роботи додатку, обробки вхідних та вихідних даних.

4) Для веб додатку розроблено логіку та екрани авторизації / реєстрації, екран перегляду власного профілю та тесту, екран редагування власного профілю та тесту, екран зі списком активних тестів та власне екран проходження тесту.

5) У четвертому розділі проведено тестування веб додатку в різних браузерах та основний функціонал по додаванню, редагуванню тесту працює добре та критичних помилок немає також створено інструкцію для користувача. Після успішного розроблення та тестування програмного веб-

додатку, виявлено, що він ефективно вирішує завдання з фотоконтролю під час проходження тестів. Розробка відповідає всім поставленим завданням та вимогам.

6) Для полегшення користування додатком для кінцевого користувача була розроблена інструкція, яка дозволяє з легкістю розібратися з основними функціями та можливостями програмного веб-додатку.

У цілому, розроблений програмний веб-додаток успішно вирішує своє призначення, пропонуючи ефективний та зручний інструмент для проходження тестів з фотоконтролем.

Проведено оцінювання комерційного потенціалу розробки, яке показало, що розробка має достатній рівень комерційного потенціалу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Чому працювати у IT-індустрії – вигідна перспектива? [Електронний ресурс] URL: https://eba.com.ua/chomu-pratsyuvaty-u-industriyi-vygidna-perspektyva/.

2. Розробка веб-застосунків. [Електронний ресурс] URL: https://webstudio2u.net/ua/site-develop/641-razrabotka-veb-prilozheniy.html

3. Кирнасюк Є.С. Розробка клієнтської частини адаптивної тестувальної системи з фотоконтролем з використанням технологій Javascript/Typescript та фреймворку Angular [Електронний ресурс] / Є.С. Кирнасюк, Катєльніков Д.І. «Стан, досягнення та перспективи інформаційних систем і технологій», Одеса. – 2022. URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2022/paper/view/15244/12857

4. Кирнасюк Є.С. Розробка клієнтської частини адаптивної тестувальної системи з фотоконтролем з використанням технологій Javascript/Typescript та фреймворку Angular [Електронний ресурс] / Є.С. Кирнасюк, Майданюк В. П. «Електронні інформаційні ресурси: створення, використання, доступ» Вінниця 2023. URL: https://ir.lib.vntu.edu.ua/bitstream/handle/123456789/37983/%d0%b7%d0%b1%d1%96%d1%80%d0%bd%d0%b8%d0%ba%20%d1%82%d0%b5%d0%b7%20%d1%87%d0%b0%d1%81%d1%82%d0%b8%d0%bd%d0%b0%203-pages-1-3%2c6%2c179-181%2c229.pdf?sequence=1&isAllowed=y

5. Робота в системі дистанційного навчання MOODLE. Інструкції для викладача – Тернопіль, ННІІОТ ТНЕУ, 2014. – 73 с

6. JetIQ. [Електронний ресурс] URL: https://jetiq.vntu.edu.ua/

7. Google Форми. [Електронний ресурс] URL: https://uk.wikipedia.org/wiki/google_форми

8. Online Test Pad. [Електронний ресурс] URL: https://onlinetestpad.com/

9. Вікіпедія водоспадна модель. [Електронний ресурс] URL:https://uk.wikipedia.org/wiki/водоспадна_модель

10. Вікіпедія спіральна модель. [Електронний ресурс] URL: https://uk.wikipedia.org/wiki/спіральна_модель

11. Вікіпедія UML. [Електронний ресурс] URL: https://uk.wikipedia.org/wiki/Unified_Modeling_Language

12. Структурна карта Константайна. [Електроний ресупс] URL: https://studref.com/311819/informatika/strukturnye_karty_konstantayna

13. Розробка інтерфейсу користувача. [Електроний ресурс] URL: https://uk.wikipedia.org/wiki/Інтерфейс_користувача

14. Capture Audio and Video in HTML5. [Електроний ресурс] URL: https://www.html5rocks.com/en/tutorials/getusermedia/intro/

15. Книга використання HTML та JavaScript для створення WEB-документів – О.Ю. Ніколаєнко, С.М. Кравченко, С.А. Вернигоренко, Київ, 2003 – 63 с

16. TypeScript. [Електроний ресурс] URL: https://uk.wikipedia.org/wiki/TypeScript

17. C++. [Електроний ресурс] URL:https://uk.wikipedia.org/wiki/C%2B%2B

18. Java 8 in Action Lambdas, streams, and functional-style programming. Рауль Габріель Урма, Маріо Фаско, Manning, 2014. – 424 с

19. Visual Studio Code. [Електроний ресурс] URL: https://uk.wikipedia.org/wiki/Visual_Studio_Code

20. WebStorm. [Електроний ресурс] URL: https://uk.wikipedia.org/wiki/WebStorm

21. Google Cloud Vision. [Електронний ресурс] URL: https://cloud.google.com/vision?hl=ru

22. Прохоренок Н. HTML, JavaScript, PHP и MySQL. Джентльменский набір Web-майстра / Н. Прохоренок.: Харків, 2008. – 640 с

23. Angular. [Електроний ресурс] URL: https://angular.io/

24. Лещев Д. Створення інтерактивного веб сайту / Д. Ле щев. Київ, 2003. – 544 с.

25. Книга «Чиста архітектура», Роберт Сесил Мартин, Фабула, 2018 – 235 с

26. Книга «Чистий код», Роберт Сесил Мартин, Фабула, 2019 – 235 с

27. Вікіпедія Модульне тестування. [Електронний ресурс] URL: https://uk.wikipedia.org/wiki/модульне_тестування

28. Інструкція користувача. [Електронний ресурс] URL: https://wiki.supplier.fozzy.ua/index.php?title=інструкція_користувача

29. Лисенко Г.Л. Методичні вказівки до оформлення курсових проектів (робіт) у Вінницькому національному технічному університеті /Уклад. Г.Л. Лисенко, А.Г. Буда, Р.Р. Обертюх, – Вінниця: ВНТУ, 2006. – 60 с.

30. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. Вінниця : ВНТУ, 2021. 42 с.

31. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа. Вінниця : ВНТУ, 2016. 113 с.

Додаток А

(обов'язковий)

Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ

Завідувач кафедрою ПЗ

д.т.н., проф.

_____ О. Н. Романюк

«19» вересня 2023 р.

Технічне завдання

на магістерську кваліфікаційну роботу «Моделі та програмні засоби дистанційного тестування з використанням технології Google Cloud Vision. Частина 2. Модуль клієнта» за спеціальністю

121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:

_____ к. т. н., доцент Майданюк В.П.

"19"09 2023 р.

Виконав:

_____ студент гр. 3ПІ-22м Кирнасюк Є.С.

"19"09 2023 р.

Вінниця – 2023 року

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Розробка адаптивної тестувальної системи з фотоконтролем. Частина 2. Модуль клієнта з використанням технологій JavaScript/Typescript і фреймворку Angular».

Галузь застосування – навчальна галузь, онлайн тестування, дистанційне навчання.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ № 247 ректора по ВНТУ про закріплення тем МКР.

3. Мета та призначення розробки.

Метою роботи є підвищення якості дистанційного тестування та зменшення випадків порушення академічної доброчесності за рахунок розробки клієнтської частини тестувальної онлайн системи з фотоконтролем.

Призначення роботи – робота призначена для дистанційного тестування знань студентів в закладах вищої освіти.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР:

1. Лещев Д. Створення інтерактивного веб сайту / Д. Лещев. Київ, 2003. – 544 с.

2. Книга «Чиста архітектура», Роберт Сесил Мартин, Фабула, 2018 – 235 с

3. Книга «Чистий код», Роберт Сесил Мартин, Фабула, 2019 – 235 с

4. Лисенко Г.Л. Методичні вказівки до оформлення курсових проектів (робіт) у Вінницькому національному технічному університеті /Уклад. Г.Л. Лисенко, А.Г. Буда, Р.Р. Обертюх, – Вінниця: ВНТУ, 2006. – 60 с.

5. Кирнасюк Є.С. Розробка клієнтської частини адаптивної тестувальної системи з фотоконтролем з використанням технологій Javascript/Typescript та фреймворку Angular [Електронний ресурс] / Є.С. Кирнасюк, Майданюк В. П. «Електронні інформаційні ресурси: створення, використання, доступ» Вінниця 2023. URL: https://ir.lib.vntu.edu.ua/bitstream/handle/123456789/37983/%d0%b7%d0%b1%d1%96%d1%80%d0%bd%d0%b8%d0%ba%20%d1%82%d0%b5%d0%b7%20%d1%87%d0%b0%d1%81%d1%82%d0%b8%d0%bd%d0%b0%203-pages-1-3%2c6%2c179-181%2c229.pdf?sequence=1&isAllowed=y

5. Технічні вимоги

Вихідні дані до роботи: модель розробки – клієнтський модуль для здійснення тестування з фотоконтролем; вхідні дані: середовище розробки WebStorm; каскадна модель розробки; мова програмування JavaScript/TypeScript; фреймворк Angular.

6. Конструктивні вимоги

Графічна та текстова документація повинна відповідати діючим стандартам України.

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

1. Пояснювальна записка до МКР;
2. Технічне завдання;
3. Лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної Роботи	Строк виконання етапів роботи
1	Аналіз та обґрунтування вибору методу розробки та постановки задачі дослідження	20.09.23 – 25.09.23
2	Розробка архітектури та алгоритмів програмного продукту	26.09.23 – 28.09.23
3	Розробка програмного продукту	29.09. 23 – 7.10.23
4	Тестування програми	8.10. 23 – 24.11.23
5	Економічна частина	27.11. 23 – 1.12.23
6	Оформлення пояснювальної записки та матеріалів до захисту	25.11. 23 – 1.12.23

10. Порядок контролю та прийняття

Порядок контролю і приймання роботи регламентується відповідними документами ВНТУ і державними стандартами.

Додаток Б
(обов'язковий)

ПРОТОКОЛ ПЕРЕВІРКИ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Моделі та програмні засоби дистанційного тестування з використанням технології Google Cloud Vision. Частина 2. Модуль клієнта.

Тип роботи: магістерська кваліфікаційна робота

Підрозділ : кафедра програмного забезпечення, ФІТКІ

Науковий керівник: к.т.н., доцент каф. ПЗ Майданюк В. П.

Unicheck	
Оригінальність	86,3
Схожість	13,7

Аналіз звіту подібності

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Черноволик Г. О.

Опис прийнятого рішення: допустити до захисту

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи _____ Кирнасюк Є.С.

Керівник роботи _____ Майданюк В. П.

Додаток В
Лістинг коду клієнтського модуля
(довідниковий)

quiz.service.ts

```
import {Injectable} from '@angular/core';
import {HttpClient} from '@angular/common/http';
import {Question, Quiz, Select} from '../interfaces';
import {BehaviorSubject} from 'rxjs';
import {ToastrService} from 'ngx-toastr';
import {environment} from '../../../environments/environment';
@Injectable({
  providedIn: 'root'
})
export class QuizService {
  readonly IMAGE_TEMPLATE = '/assets/imgs/quiz-image-template.png';
  private readonly quizzesExample: Quiz[] = [
    {
      quizImage:
'https://upload.wikimedia.org/wikipedia/ru/thumb/3/39/Java_logo.svg/1200px-
Java_logo.svg.png',
      category: 'Programing',
      completions: 2,
      difficulty: 'HARD',
      id: 1,
      likes: 32,
      quizName: 'Java',
    }
  ];
  quizzes: Quiz[] = this.quizzesExample;
```

```
  generalQuizList$: BehaviorSubject<Quiz[]> = new
BehaviorSubject<Quiz[]>(this.quizzesExample);
  quizList$: BehaviorSubject<Quiz[]> = new BehaviorSubject<Quiz[]>([]);
  difficultySelect: Select[] = [
    {value: 'EASY', viewValue: 'Easy'},
    {value: 'MEDIUM', viewValue: 'Medium'},
    {value: 'HARD', viewValue: 'Hard'},
    {value: 'EXPERT', viewValue: 'Expert'},
  ];
  categorySelect: Select[] = [
    {value: 'Programming', viewValue: 'Programming'},
    {value: 'Language', viewValue: 'Language'},
    {value: 'Math', viewValue: 'Math'}
  ];
  constructor(private http: HttpClient, private toastr: ToastrService) {
    this.createNewQuiz({
      id: 1653043698821,
      quizName: 'JavaScript',
      category: 'Programming',
      difficulty: 'MEDIUM',
      quizImage: '',
      completions: 1,
      likes: 2,
      questions: [{
        id: Date.now(),
        name: 'Що таке JavaScript?',
        options: [
          {text: 'Автомобіль', isRight: false},
          {text: 'Мова програмування', isRight: true},
          {text: 'Книжка', isRight: false},
```

```
      {text: 'Нічого з перечисляного', isRight: false}
    ]
  },
    {
      id: Date.now() + 1,
      name: 'Java та JavaScript те саме?',
      options: [
        {text: 'Так', isRight: false},
        {text: 'Ні', isRight: true},
        {text: 'Важко сказати', isRight: false}
      ]
    },
    {
      id: Date.now() + 2,
      name: 'JavaScript це ООП мова програмування?',
      options: [
        {text: 'Ні', isRight: true},
        {text: 'Важко сказати', isRight: false},
        {text: 'Так', isRight: false},
      ]
    }]
});
this.generalQuizList$.subscribe(quizzes => {
  quizzes.forEach(el => {
    if (el.quizImage === '' || el.quizImage == null) {
      el.quizImage = this.IMAGE_TEMPLATE;
    }
  });
  console.log('generalQuizList$', quizzes);
});
```

```
  }
  deleteQuestion(quizId: number, questionId: number) {
    const tmpQuizList = this.quizList$.getValue();
    const quizIndex = tmpQuizList.findIndex(obj => obj.id === quizId);
    const questionIndex = tmpQuizList[quizIndex].questions.findIndex(obj =>
obj.id === questionId);
    tmpQuizList[quizIndex].questions.splice(questionIndex, 1);
    this.quizList$.next(tmpQuizList);
    this.toastr.success('Question was successfully deleted');
}
  createNewQuiz(quiz: Quiz) {
    Object.assign(quiz, {id: Date.now()});
    Object.assign(quiz, {likes: 32});
    Object.assign(quiz, {completions: Math.random()});
    this.quizList$.next([...this.quizList$.getValue(), quiz]);
    this.generalQuizList$.next([quiz, ...this.generalQuizList$.getValue()]);
  }
  updateQuizInfo(quiz: Quiz) {
    const tmpQuizList = this.quizList$.getValue();
    const quizIndex = tmpQuizList.findIndex(obj => obj.id === quiz.id);
    tmpQuizList[quizIndex] = quiz;
    this.quizList$.next(tmpQuizList);
    this.toastr.success('Quiz was successfully updated');
  }
  createPOSTNewQuiz(quiz: Quiz) {
    this.http.post<Quiz>(`${environment.baseUrl}/test`, quiz).subscribe(res =>
{
      console.log('Response', res);
      this.generalQuizList$.next([{...res}]);
    });
```

```
  }
}
```

quiz-create-modal.component.html

```
        <h2 class="quiz-create__header">Create Your quiz</h2>
<form [formGroup]="createQuizForm">
  <mat-form-field class="full-width" appearance="fill">
    <mat-label>Enter quiz name</mat-label>
    <input matInput type="text" autocomplete="off"
formControlName="quizName"
        placeholder="Quiz name">
    <mat-error *ngIf="createQuizForm.get('quizName').hasError('required')">
     Quiz name required
    </mat-error>
  </mat-form-field>
  <br>
  <mat-form-field class="full-width" appearance="fill">
    <mat-label>Choose difficulty</mat-label>
    <mat-select formControlName="difficulty">
      <mat-option *ngFor="let item of quizService.difficultySelect"
[value]="item.value">
       {{item.viewValue}}
      </mat-option>
    </mat-select>
    <mat-error
*ngIf="createQuizForm.get('difficulty').hasError('required')">Please choose an
difficulty</mat-error>
  </mat-form-field>
  <br>
```

```
  <mat-form-field class="full-width" appearance="fill">
    <mat-label>Choose category</mat-label>
    <mat-select formControlName="category">
      <mat-option *ngFor="let item of quizService.categorySelect"
[value]="item.value">
        {{item.viewValue}}
      </mat-option>
    </mat-select>
    <mat-error
*ngIf="createQuizForm.get('category').hasError('required')">Please choose an
difficulty</mat-error>
  </mat-form-field>
  <ngp-image-picker
    ($imageChanged)="onImageChanged($event)"
    [_config]="imagePickerConfig"
    [_imageSrc]="initialImage"
  ></ngp-image-picker>
  <br>
  <mat-dialog-actions align="end">
    <button mat-button mat-dialog-close>Cancel</button>
    <button mat-raised-button [mat-dialog-close]="true" color="primary"
type="submit" (click)="onSubmit()"
          [disabled]="!createQuizForm.valid">Submit
    </button>
  </mat-dialog-actions>
</form>
```

quiz-create-modal.component.ts

```
import {Component, OnInit} from '@angular/core';
```

```
import {Quiz, Select} from '../../interfaces';
import {FormControl, FormGroup, Validators} from '@angular/forms';
import {ToastrService} from 'ngx-toastr';
import {QuizService} from '../../services/quiz.service';
import {ImagePickerConf} from 'ngp-image-picker';
@Component({
  selector: 'app-quiz-create',
  templateUrl: './quiz-create-modal.component.html',
  styleUrls: ['./quiz-create-modal.component.scss']
})
export class QuizCreateModalComponent implements OnInit {
  createQuizForm = new FormGroup({
    quizName: new FormControl('', Validators.required),
    difficulty: new FormControl('', Validators.required),
    category: new FormControl('', Validators.required)
  });
  imagePickerConfig: ImagePickerConf = {
    borderRadius: '1px',
    language: 'en',
    width: '100px',
    objectFit: 'contain',
    aspectRatio: 4 / 3,
    compressInitial: null,
  };
  imageSrc: any = '';
  initialImage: string = '';
  constructor(private toastr: ToastrService, public quizService: QuizService) {
  }
  ngOnInit(): void {
  }
```

```
    onSubmit() {
      if (this.createQuizForm.valid) {
        const newQuizData: Quiz = this.createQuizForm.value;
        Object.assign(newQuizData, {quizImage: this.imageSrc});
        Object.assign(newQuizData, {questions: []});
        this.quizService.createNewQuiz(newQuizData);
        this.toastr.success('Quiz created');
      }
    }
    onImageChanged(dataUri) {
      this.imageSrc = dataUri;
    }
}
```

quiz-builder.component.html

```
<hr>
<div>
  <h2 *ngIf="isEditable">Create your question</h2>
  <div>
    <input type="text" placeholder="Put your question here" [disabled]="!isEditable"
[(ngModel)]="quizQuestion.name">
    <h3 *ngIf="isEditable">Add answer to your quiz</h3>
    <div>
      <input type="text" placeholder="Answer"  *ngIf="isEditable"
[(ngModel)]="newAnswer" [disabled]="!isEditable">
      <button (click)="addAnswer()" *ngIf="isEditable"
[disabled]="newAnswer==""">+</button>
    </div>
```

```
    <mat-radio-group
      aria-labelledby="example-radio-group-label"
      class="example-radio-group"
      [(ngModel)]="rightAnswer" (change)="changeRightValue($event.value)"
[disabled]="!isEditable">
      <mat-radio-button class="example-radio-button" [disabled]="!isEditable"
*ngFor="let option of quizQuestion.options" [value]="option.text">
        {{option.text}}
      </mat-radio-button>
    </mat-radio-group>
  </div>
  <button mat-raised-button color="primary" *ngIf="isEditable"
(click)="deleteQuestion()">Delete</button>
</div>
```

quiz-builder.component.ts

```
import {Component, Input, OnInit} from '@angular/core';
import {Question, Quiz} from '../../interfaces';
import {QuizService} from '../../services/quiz.service';

@Component({
  selector: 'app-question-builder',
  templateUrl: './question-builder.component.html',
  styleUrls: ['./question-builder.component.scss']
})
export class QuestionBuilderComponent implements OnInit {

  @Input() quizQuestion: Question;
```

```
    @Input() quizInfo: Quiz;
    @Input() isEditable: boolean = false;
    rightAnswer: string = '';
    newAnswer: string = '';
    constructor(private quizService: QuizService) {
    }
    ngOnInit(): void {
      return this.quizQuestion.options.forEach((el) => {
        if (el.isRight) {
          this.rightAnswer = el.text;
        }
      });
    }
    changeRightValue(rightValue) {
      this.quizQuestion.options.forEach(el => {
        el.isRight = el.text === rightValue;
      });
    }
    addAnswer() {
      this.quizQuestion.options.unshift({text: this.newAnswer, isRight: false});
      this.changeRightValue(this.newAnswer);
      this.rightAnswer = this.newAnswer;
      this.newAnswer = '';
    }
    deleteQuestion() {
      this.quizService.deleteQuestion(this.quizInfo.id, this.quizQuestion.id);
    }
}
```

photo-register.component.html

```
<div style="text-align:center">
  <div>
    <webcam [height]="500" [width]="500" [trigger]="triggerObservable"
(imageCapture)="handleImage($event)"
          *ngIf="showWebcam"
          [allowCameraSwitch]="allowCameraSwitch"
[switchCamera]="nextWebcamObservable"
          [videoOptions]="videoOptions"
          [imageQuality]="1"
          (cameraSwitched)="cameraWasSwitched($event)"
          (initError)="handleInitError($event)"
    ></webcam>
    <br/>
    <button class="actionBtn" (click)="triggerSnapshot();">Take A
Snapshot</button>
    <button class="actionBtn" (click)="toggleWebcam();">Toggle
Webcam</button>
    <br/>
    <button class="actionBtn" (click)="showNextWebcam(true);"
[disabled]="!multipleWebcamsAvailable">Next Webcam
    </button>
    <input id="cameraSwitchCheckbox" type="checkbox"
[(ngModel)]="allowCameraSwitch"><label
for="cameraSwitchCheckbox">Allow
    Camera Switch</label>
    <br/>
    DeviceId: <input id="deviceId" type="text" [(ngModel)]="deviceId"
style="width: 500px">
```

```
    <button (click)="showNextWebcam(deviceId);">Activate</button>
  </div>
</div>

<div class="snapshot" *ngIf="webcamImage">
  <h2>Nice one!</h2>
  <img [src]="webcamImage.imageAsDataUrl"/>
</div>

<h4 *ngIf="errors.length > 0">Messages:</h4>
<ul *ngFor="let error of errors">
  <li>{{error | json}}</li>
</ul>
```

photo-register.component.ts

```
import {Component, OnInit} from '@angular/core';
import {Subject} from 'rxjs';
import {WebcamImage, WebcamInitError, WebcamUtil} from 'ngx-webcam';

@Component({
  selector: 'app-photo-register',
  templateUrl: './photo-register.component.html',
  styleUrls: ['./photo-register.component.scss']
})
export class PhotoRegisterComponent implements OnInit {
  public showWebcam = true;
  public allowCameraSwitch = true;
  public multipleWebcamsAvailable = false;
  public deviceId: string;
```

```
  public videoOptions: MediaTrackConstraints = {
    // width: {ideal: 1024},
    // height: {ideal: 576}
  };
  public errors: WebcamInitError[] = [];
  public webcamImage: WebcamImage = null;
  private trigger: Subject<void> = new Subject<void>();
  private nextWebcam: Subject<boolean|string> = new
Subject<boolean|string>();

  public ngOnInit(): void {
    WebcamUtil.getAvailableVideoInputs()
      .then((mediaDevices: MediaDeviceInfo[]) => {
        this.multipleWebcamsAvailable = mediaDevices &&
mediaDevices.length > 1;
      });
  }
  public triggerSnapshot(): void {
    this.trigger.next();
  }
  public toggleWebcam(): void {
    this.showWebcam = !this.showWebcam;
  }
  public handleInitError(error: WebcamInitError): void {
    this.errors.push(error);
  }
  public showNextWebcam(directionOrDeviceId: boolean|string): void {
    this.nextWebcam.next(directionOrDeviceId);
  }
}
```

user.service.ts

```
import { Injectable } from '@angular/core';
import { AuthService } from './auth.service';
import { QuizData, UserInfo } from '../interfaces';
import { environment } from '../../../environments/environment';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class UserService {
  userInfo: UserInfo = {} as UserInfo;
  userQuizList: QuizData[] = [];

  constructor(private authService: AuthService, private http: HttpClient) {
    this.authService.isLoggedIn$.subscribe({
      next: isLoggedIn => {
        if (isLoggedIn) {
          this.getUserInfoRequest().subscribe({
            next: userInfoResponse => {
              this.userInfo = userInfoResponse;
            }
          });
        }
      }
    });
  }
```

```
  updateUserQuizList() {
    this.getUserQuizzesRequest().subscribe({
      next: userQuizListResponse => {
        this.userQuizList = userQuizListResponse;

        this.userQuizList.forEach(el => {
          if (el.imagePath === '' || el.imagePath == null) {
            el.imagePath = '/assets/imgs/quiz-image-template.png';
          }
        });
      }
    });
  }

  private getUserQuizzesRequest(): Observable<QuizData[]> {
    return this.http.get<QuizData[]>(`${environment.baseUrl}/user/quizzes`,
this.authService.authorizationHeaders);
  }

  private getUserInfoRequest(): Observable<UserInfo> {
    return this.http.get<UserInfo>(`${environment.baseUrl}/user`,
this.authService.authorizationHeaders);
  }
}
```

auth.servive.ts

```
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { environment } from '../../../environments/environment';
```

```
import { BehaviorSubject, Observable } from 'rxjs';
import { AuthorizationData, LoginData, RegisterData, TokenResponse } from
'../interfaces';

@Injectable({
  providedIn: 'root'
})
export class AuthService {
  isLoggedIn$: BehaviorSubject<boolean> = new
BehaviorSubject<boolean>(false);

  set jwt(value: string) {
    localStorage.setItem('token', 'Bearer ' + value);
  }

  get jwt(): string {
    return localStorage.getItem('token');
  }

  get authorizationHeaders(): AuthorizationData {
    return {
      headers: new HttpHeaders({
        Authorization: this.jwt
      })
    };
  }

  constructor(private http: HttpClient) {
    if (localStorage.getItem('token') != null) {
      this.isLoggedIn$.next(true);
```

```
    }
  }

  logout() {
    localStorage.removeItem('token');
    this.isLoggedIn$.next(false);
  }

  loginRequest(loginData: LoginData): Observable<TokenResponse> {
    return this.http.post<TokenResponse>(`${environment.baseUrl}/login`,
loginData);
  }

  registerRequest(registerData: RegisterData): Observable<TokenResponse> {
    return this.http.post<TokenResponse>(`${environment.baseUrl}/register`,
registerData);
  }

}
```

my-quiz-item-page.component.ts

```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute, ParamMap } from '@angular/router';
import { QuizService } from '../../../../shared/services/quiz.service';
import { QUESTION_TYPE, QuizData, QuizQuestion } from
'../../../../shared/interfaces';
import { ImagePickerConf } from 'ngp-image-picker';
import { FormControl, FormGroup, Validators } from '@angular/forms';
import { UserService } from '../../../../shared/services/user.service';
```

```
@Component({
  selector: 'app-my-quiz-item-page',
  templateUrl: './my-quiz-item-page.component.html',
  styleUrls: ['./my-quiz-item-page.component.scss']
})
export class MyQuizItemPageComponent implements OnInit {
  private id: number;
  quizInfo: QuizData;

  imagePickerConfig: ImagePickerConf = {
    borderRadius: '1px',
    language: 'en',
    width: '200px',
    objectFit: 'contain',
    // aspectRatio: 4 / 3,
    compressInitial: null
  };

  myQuizForm = new FormGroup({
    quizName: new FormControl('', Validators.required),
    difficulty: new FormControl('', Validators.required),
    category: new FormControl('', Validators.required)
  });
  isFormEditable: boolean = false;
  editButtonText: string = 'Edit';


  constructor(private route: ActivatedRoute, public quizService: QuizService,
public userService: UserService) {
```

```
}

ngOnInit(): void {
  this.myQuizForm.disable();
  this.route.paramMap.subscribe((params: ParamMap) => {
    this.id = Number.parseInt(params.get('id'));
    this.quizInfo = this.userService.userQuizList.find(obj => {
      return obj.id === this.id;
    });
    this.myQuizForm.get('quizName').setValue(this.quizInfo.quizName);
    this.myQuizForm.get('difficulty').setValue(this.quizInfo.difficulty);
    this.myQuizForm.get('category').setValue(this.quizInfo.category);
  });
}
goBack() {
  window.history.back();
}
imageChange(dataUri) {
  this.quizInfo.imagePath = dataUri;
}
updateQuizData() {
  console.log('updateQuizData');
  this.quizInfo.quizName = this.myQuizForm.get('quizName').value;
  this.quizInfo.difficulty = this.myQuizForm.get('difficulty').value;
  this.quizInfo.category = this.myQuizForm.get('category').value;

  console.log('new this.quizInfo', this.quizInfo);

  this.quizService.updateQuizRequest(this.quizInfo).subscribe({
      next: (updateQuizResponse) => {
```

```
        console.log('updateQuizResponse', updateQuizResponse);
        this.toggleOnEditQuiz();
      }}
  );
}

createQuestion() {
  const emptyQuestion: QuizQuestion = {
    id: Date.now(),
    questionText: '',
    options: [],
    type: QUESTION_TYPE.RADIO_BUTTON
  };
  this.quizInfo.questionList.unshift(emptyQuestion);
}

toggleOnEditQuiz() {
  this.myQuizForm.get('quizName').setValue(this.quizInfo.quizName);
  this.myQuizForm.get('difficulty').setValue(this.quizInfo.difficulty);
  this.myQuizForm.get('category').setValue(this.quizInfo.category);
  if (this.isFormEditable) {
    this.myQuizForm.disable();
    this.editButtonText = 'Edit';
  } else {
    this.myQuizForm.enable();
    this.editButtonText = 'Close';
  }
  this.isFormEditable = !this.isFormEditable;
}
```

Додаток Г

(обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

МОДЕЛІ ТА ПРОГРАМНІ ЗАСОБИ ДИСТАНЦІЙНОГО ТЕСТУВАННЯ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ GOOGLE CLOUD VISION. ЧАСТИНА 2. МОДУЛЬ КЛІЄНТА.

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмної інженерії

Магістерська кваліфікаційна робота

на тему:
Моделі та програмні засоби дистанційного тестування з використанням технології Google Cloud Vision. Частина 2. Модуль клієнта.

Виконав:
студент групи ЗПІ-22м
Кирнасюк Євген Сергійович

Науковий керівник:
к.т.н., доцент кафедри ПЗ
Майданюк Володимир Павлович

Рисунок Г.1 – Слайд презентації №1

Мета, предмет та об'єкт дослідження

Мета та завдання дослідження. Метою роботи є підвищення якості дистанційного тестування та зменшення випадків порушення академічної доброчесності за рахунок розробки клієнтської частини тестувальної онлайн системи з фотоконтролем.

Об'єктом дослідження – процес здійснення дистанційного тестування з фото контролем.

Предмет дослідження – методи та програмні засоби розробки клієнтської частини тестувальної системи з фото контролем.

Рисунок Г.2 – Слайд презентації №2

Наукова новизна та практична цінність отриманих результатів

Наукова новизна отриманих результатів.

1. Подальшого розвитку отримав метод дистанційного тестування, у якому на відміну від існуючих, додано функції отримання та передачі на сервер результатів фотоконтролю, що дозволило підвищити достовірність тестування на 10 %.
2. Подальшого розвитку отримав метод побудови архітектури систем дистанційного тестування, у якому, на відміну від існуючих, використано архітектуру типу Rich client для тестування, що зменшує навантаження на сервер та економить трафік.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає в тому, що на основі отриманих у магістерській кваліфікаційній роботі теоретичних положень запропоновано алгоритми та розроблено клієнтський модуль тестувальної системи з фото контролем.

Рисунок Г.3 – Слайд презентації №3

Аналіз аналогів

	JetIQ	Online Test Pad	Google Forms	TestBuilder
Загальнодоступний	–	+	+	+
Створення розважальних тестів	–	+	–	–
Фото контроль проходження тесту	–	–	–	+
Аналіз фото контролю	–	–	–	+
Створення власних тестів	–	+	+	+
Статистика по проходження тесту	+	+	+	+
Часовий контроль проходження окремого питання	–	+	–	+
Орієнтованість на навчальні заклади	+	–	–	+
Результат:	2	5	3	7

Рисунок Г.4 – Слайд презентації №4

Angular — написаний на TypeScript front-end фреймворк з відкритим кодом, який розробляється у компанії Google. Angular — це AngularJS, який був переосмислений та перероблений тією ж командою розробників.

Технологія Angular використовується у вебдодатках таких компаній:

- Google (Gmail, Google Play, Google Translate, Google Ads, Youtube);
- PayPay;
- Upwork;
- Expedia Group;
- Lego;
- Adidas.

Рисунок Г.5 – Слайд презентації №5

TypeScript — мова програмування, представлена Microsoft восени 2012; позиціонується як засіб розробки вебзастосунків, що розширює можливості JavaScript.

TypeScript є зворотньо сумісним з JavaScript. Фактично, після компіляції програму на TypeScript можна виконувати в будь-якому сучасному браузері або використовувати спільно з серверною платформою Node.js.

Основний принцип мови — будь-який код на JavaScript сумісний з TypeScript, тобто в програмах на TypeScript можна використовувати стандартні JavaScript-бібліотеки і раніше створені напрацювання. Більш того, можна залишити наявні JavaScript-проекти в незмінному вигляді, а дані про типізації розмістити у вигляді анотацій, які можна помістити в окремі файли, які не заважатимуть розробці і прямому використанню проекту (наприклад, подібний підхід зручний при розробці JavaScript-бібліотек).

Рисунок Г.6 – Слайд презентації №6

Схема роботи веб додатку (SPA) напсаного на фрейворку Angular

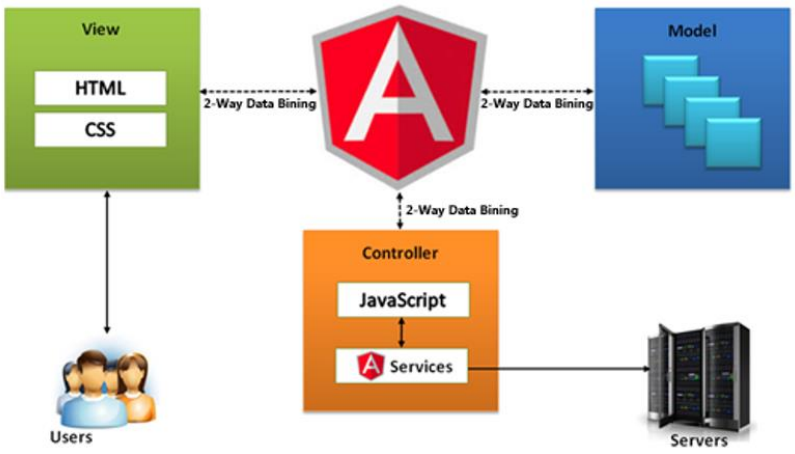
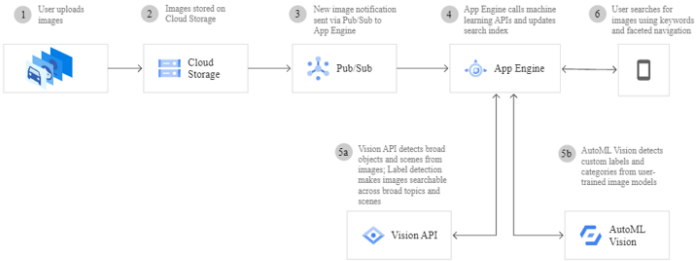


Рисунок Г.7 – Слайд презентації №7

Google Cloud Vision

Google Cloud Vision – це служба компанії Google, яка надає можливість використання розпізнавання образів та комп'ютерного зору у ваших програмах та проектах. Цей сервіс пропонує API для розв'язання завдань, пов'язаних з обробкою та розпізнаванням зображень.

Важливо відзначити, що Google Cloud Vision використовує глибокі нейронні мережі для розпізнавання образів. Це означає, що алгоритми базуються на великому обсязі навчальних даних та потужних моделях глибокого навчання, що дозволяє їм високоефективно розпізнавати різні об'єкти та характеристики на зображеннях.



Алгоритм роботи Google Cloud Vision

Рисунок Г.8 – Слайд презентації №8

Алгоритми роботи програми

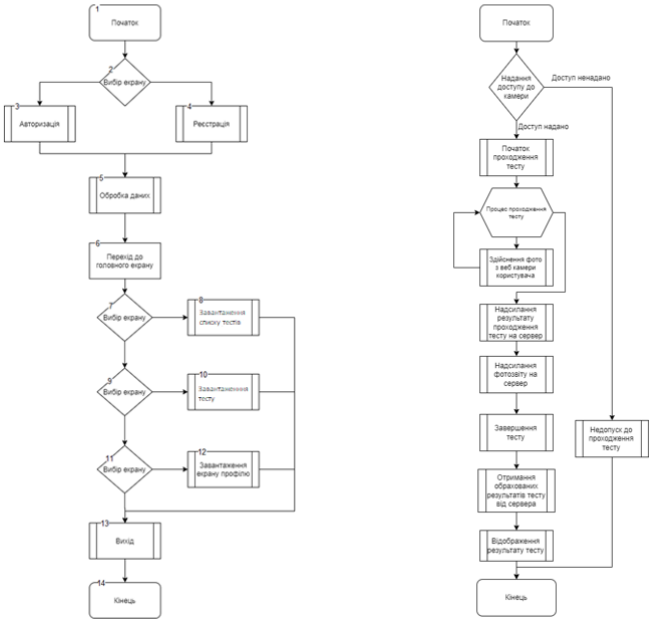
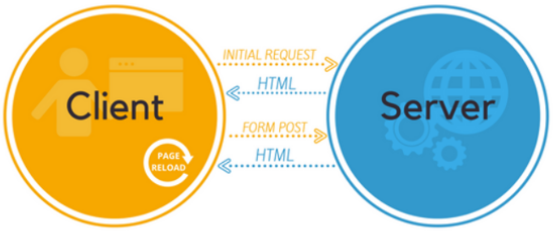


Рисунок Г.9 – Слайд презентації №9

Багатосторінкові (веб сайти) та односторінкові додатки (веб додатки), їх переваги та недоліки



Принцип роботи багато сторінкового додатку (MPA)

Недоліки багатосторінкових додатків

•Розробка як десктопної, так і мобільної версії веб-сайту займає значно більше часу;

•Збільшує вартість змін при додаванні нових функціональних можливостей в додатку.

Переваги багатосторінкових додатків

•Багатосторінкова (MPA) архітектура дозволяє легко оптимізувати кожну сторінку для пошукових систем. Розробник може додати мета-теги для будь-якої сторінки;

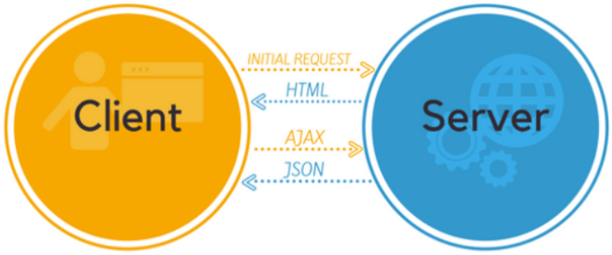
•Як правило, для розробки багатосторінкової програми потрібен менший стек технологій, таким чином вартість таких додатків виходить дешевшою.

Рисунок Г.10 – Слайд презентації №10

Односторінкова програма веб додатки (SPA)

Недоліки:
- Залежність від мережі;
- Висока конкуренція;
- Продуктивність та оптимізація для слабких пристроїв (необхідно пам'ятати про мобільні пристрої та пристрої з низькою продуктивністю)

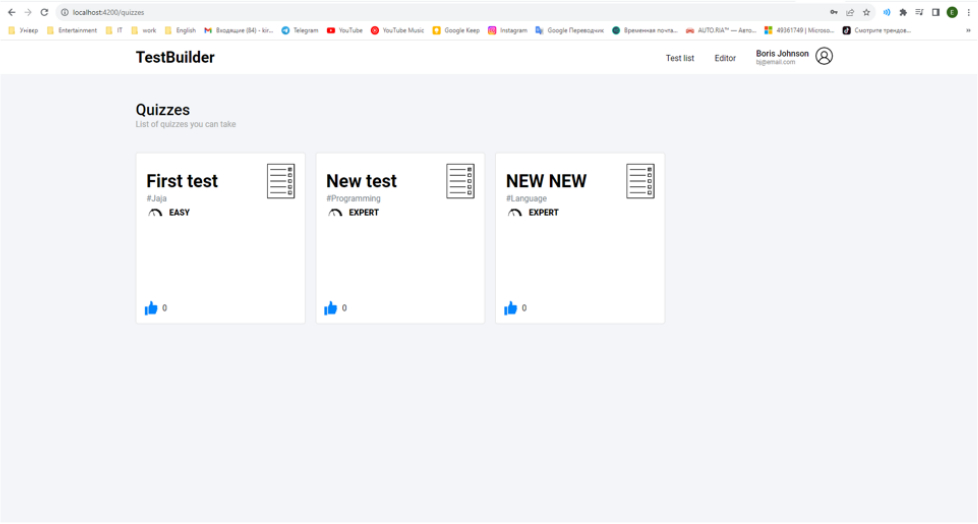
Переваги:
- Мультиплатформеність;
- Модульність;
- Популярність;
- Швидкість розробки;
- Високий рівень розвитку



Принцип роботи односторінкового додатку

Рисунок Г.11 – Слайд презентації №11

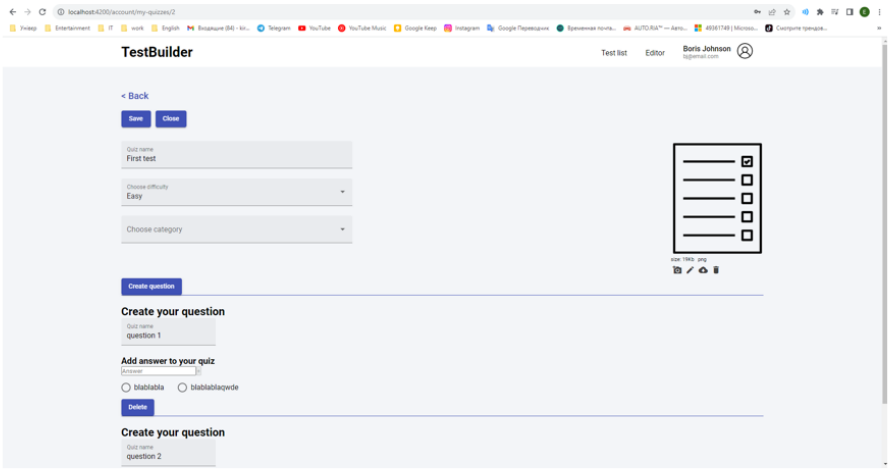
Приклад роботи програми



Головна сторінка на якій зображено загальний список тестів

Рисунок Г.12 – Слайд презентації №12

Сторінка редагування тесту



Конструктор тесту(режим редагування)

Рисунок Г.13 – Слайд презентації №13

Список усіх тестів в кабінеті викладача

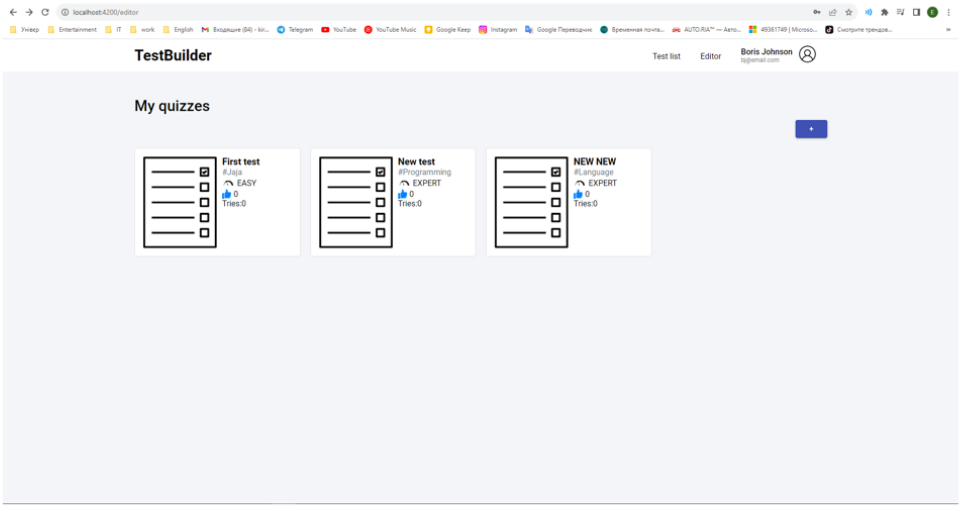
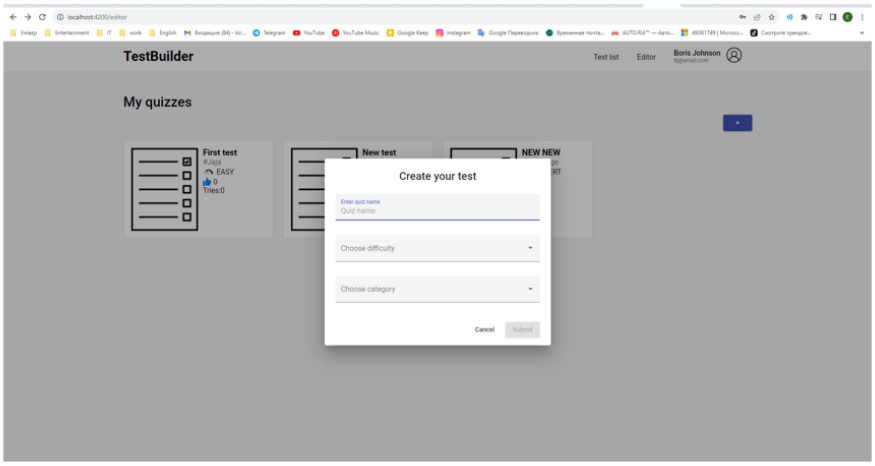


Рисунок Г.14 – Слайд презентації №14

Модальне вікно для створення тесту



Створення тесту

Рисунок Г.15 – Слайд презентації №15

Результата проходження тесту з аналізом Google Cloud Vision

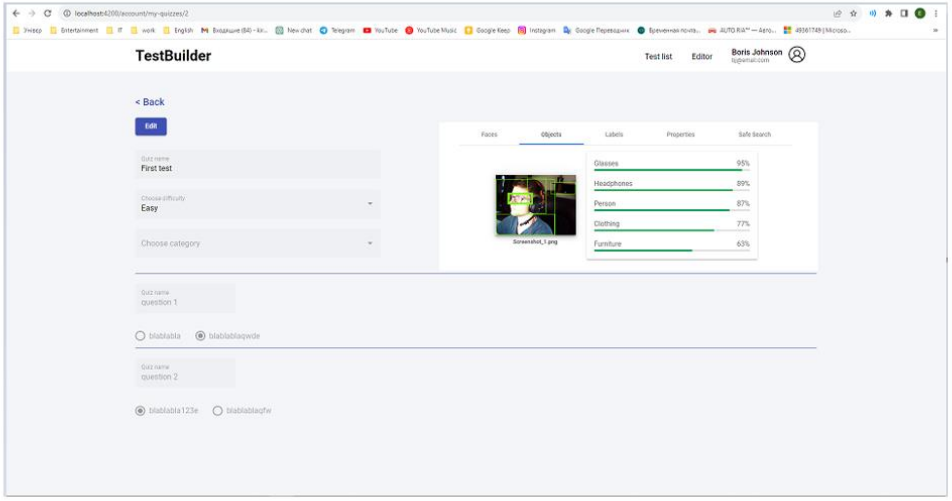


Рисунок Г.16 – Слайд презентації №16

Дякую за увагу!

Рисунок Г.17 – Слайд презентації №17