Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Пояснювальна записка

до бакалаврської дипломної роботи
бакалавр
(освітньо–кваліфікаційний рівень)

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

Виконав: студент IV курсу
групи 2ПІ-18б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Кирнасюк Є.С.
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ Катєльніков Д. І.
(прізвище та ініціали)
Рецензент: к.т.н.,доц.каф. КН Колесницький О.К.
(прізвище та ініціали)

ВНТУ– 2022

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

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
д.т.н., професор
Романюк О. Н._____
" 25 " березня 2022 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Кирнасюку Євгену Сергійовичу

1. Тема роботи – розробка адаптивної тестувальної системи з фотоконтролем. Частина 2. Модуль клієнта з використанням технологій JavaScript/Typescript і фреймворку Angular.

Керівник роботи: Катєльніков Денис Іванович, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від «24» березня 2022 року № 66.

2. Строк подання студентом роботи _____.

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 розрахунково-пояснювальної записки: вступ; аналіз та обґрунтування вибору методу розробки та постановка задачі дослідження; розробка структури та алгоритмів роботи програмного продукту; розробка програмних компонент для додатку; тестування програми; висновки; список використаних джерел; додатки.

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	завдання прийняв
1-4	Катєльніков Д.І., к.т.н., доцент кафедри ПЗ		

7. Дата видачі завдання 25 березня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз завдання і вибір методу вирішення поставленої задачі дослідження	26.03.2022 – 10.04.2022	Вик.
2	Розробка алгоритму створення тесту	11.04.2022 – 17.04.2022	Вик.
3	Розробка алгоритму проходження тестів з фотоконтролем	18.04.2022 – 21.04.2022	Вик.
4	Аналіз і вибір мови програмування та середовища розробки	22.04.2022 – 15.05.2022	Вик.
5	Програмна реалізація веб додатку	16.05.2022 – 23.05.2022	Вик.
6	Тестування розробленого веб додатку	24.05.2022 – 10.06.2022	Вик.
7	Оформлення матеріалів до захисту БДР	26.03.2022 – 10.04.2022	Вик.

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

Керівник бакалаврської дипломної роботи _____ **Катєльніков Д.І.**
(підпис) (прізвище та ініціали)

Рецензент бакалаврської дипломної роботи _____ **Колесницький О.К.**

(підпис) (прізвище та ініціали

АНОТАЦІЯ

В бакалаврській дипломній роботі розроблено модуль клієнтської частини вебдодатку адаптивної тестувальної системи з фотоконтролем. Розроблений веб додаток призначений для покращення процесу створення та проходження онлайн тестів.

Даний веб додаток написаний на мові JavaScript/TypeScript з використанням фреймворку Angular, та характеризується швидкодією та адаптивністю.

Бакалаврська дипломна робота складається з 77 сторінок формату А4, на яких є 52 рисунків, 3 таблиці, список використаних джерел містить 15 найменувань.

У бакалаврській дипломній роботі проведено детальний аналіз стану програмного забезпечення для проведення тестування. Сформульовано мету досліджень – підвищення ефективності процесу тестування людей, забезпечення чесності результату тесту за допомогою 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.

Розроблено алгоритми для реалізації процесу онлайн тестування з фотоконтролем.

ABSTRACT

In the bachelor's thesis the module of the client part of the web application of the adaptive testing system with photocontrol is developed. The developed web application is designed to improve the process of creating and passing online tests.

This web application is written in JavaScript / TypeScript using the Angular framework, and is characterized by speed and adaptability.

The bachelor's thesis consists of 77 A4 pages, which contain 52 figures, 3 tables, a list of sources used contains 15 titles.

In the bachelor's thesis a detailed analysis of the software for testing was conducted. The purpose of the research is formulated - to increase the efficiency of the process of testing people, to ensure the honesty of the test result through photocontrol 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.

Algorithms for the implementation of the process of online testing with photocontrol have been developed.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. У нас 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.

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 між-платформовими сервісами.

Програмування та технології змінило спосіб життя людини, ми отримали змогу працювати дистанційно, отримувати останні новини з будь-якої точки нашого розташування, можливість навчатися новому завдяки Інтернету.

Саме тому різні за діяльністю групи населення відчули необхідність мати в своєму розпорядженні веб додаток, в якому вони зможуть дізнаватися рівень своїх знань або навіть створювати тести для інших людей.

Так само для студентів і викладачів, є необхідність в такому програмному додатку де викладачі зможуть перевіряти рівень знань студентів,
за допомогою адаптивної тестувальної системи з фотоконтролем, бачучи порушення студентів та статистику по проходження тесту.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення [1].

Мета та завдання дослідження. Метою бакалаврської дипломної роботи є розробка адаптивної тестувальної системи з фотоконтролем. Частина 2. Модуль клієнта з використанням технологій JavaScript/Typescript і фреймворку Angular.

Об’єктом дослідження – є процес розробки адаптивної тестувальної системи з фотоконтролем, а саме модуля клієнта з використанням технологій JavaScript/Typescript і фреймворку Angular..

Предмет дослідження – методи та засоби розробки адаптивної тестувальної системи з фотоконтролем.

Завданням дипломної роботи було визначено:

- Створення клієнтської частини веб додатку (інтерфейсу користувача);
- Додання клієнтського модуля для створення тесту;
- Додання клієнтського модуля для перегляду усіх доступних тестів.
- Додання клієнтського модуля для проходження тесту з фотоконтролем.
- Додання клієнтського модуля для роботи з сервером.
- Додання клієнтського модуля для надсилання результатів фотоконтролю.

Методи дослідження. У процесі дослідження використовувались: комплексний аналіз з метою визначення недоліків існуючих аналогів та подальший синтез отриманих даних для формування нових функціональних характеристик і вимог; комп’ютерне моделювання та засоби програмного проектування для розробки веб додатку адаптивної тестувальної системи з фотоконтролем; методи тестування для перевірки роботи створеного веб додатку.

Новизна отриманих результатів.

1. Удосконалено алгоритм створення тестів та їх відображання за допомогою використання фреймворку Angular та мови програмування TypeScript.

2. Запропоновано алгоритм фіксації проходження тесту, а саме вести фотофіксацію користувача під час проходження ним тесту.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає в тому, що на основі отриманих в бакалаврській дипломній роботі теоретичних та практичних положень запропоновано алгоритм фотофіксації тесту та розроблено веб додаток для підвищення ефективності процесу навчання та тестувань знань студентів.

Особистий внесок здобувача. Усі наукові результати, що викладені у бакалаврській дипломній роботі, отримано автором самостійно. У наукових працях, опублікованих у співавторстві, автору належать такі результати: використання фреймворку Angular для розробки гнучких веб додатків, розробка алгоритму проходження тесту з фотофіксацією та використання засобів керування JavaScript/Typescript пакетами [2]**.**

Апробація результатів роботи. Основні положення бакалаврської дипломної роботи доповідалися на LI Науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії (2022 р., м. Вінниця);

Публікації. За тематикою дослідження опубліковано 1 наукову працю праці у збірниках матеріалів конференцій.

1 АНАЛІЗ ТА ОБҐРУНТУВАННЯ ВИБОРУ МЕТОДУ РОЗРОБКИ ТА ПОСТАНОВКИ ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Аналіз стану проблеми

Дистанційне навчання в світі поступово перейшло з розряду дискусійних питань у широко розповсюджену практику. Ще десять років тому цей термін у свідомості педагогів асоціювався з повністю віддаленим навчанням, при якому очні зустрічі зі студентами, якщо й відбуваються, то досить рідко, і тому складно перевірити автентичність студента та справжній рівень його знань. При цьому потрібно спочатку витратити багато часу, щоб налаштувати дистанційні курси та тести, а потім правильно їх використати [3].

У випадку з освітніми установами університети, коледжі та школи по всьому світу практикують створення власних веб додатків, мобільних додатків у яких публікується інформація про їх навчальний заклад. Всі вони націлені на надання інформації від закладу до абітурієнтів, але одиниці надають функціонал який здатний надавати не тільки знання а і перевіряти їх, зменшуючи при цьому рівень шахрайства при проходженні.

Наявність веб додатку який був би націлений на освітні установи значно спростив би тестування студентів для зрізу рівня знань якими вони володіють. Можливість перегляду статистики тесту, результатів студента та фото звіт який буде надавати сервіс після проходження тесту, за для забезпечення адекватної та чесної оцінки знань конкретного студента.

1.2 Порівняльний аналіз аналогів

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 [4]. 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).

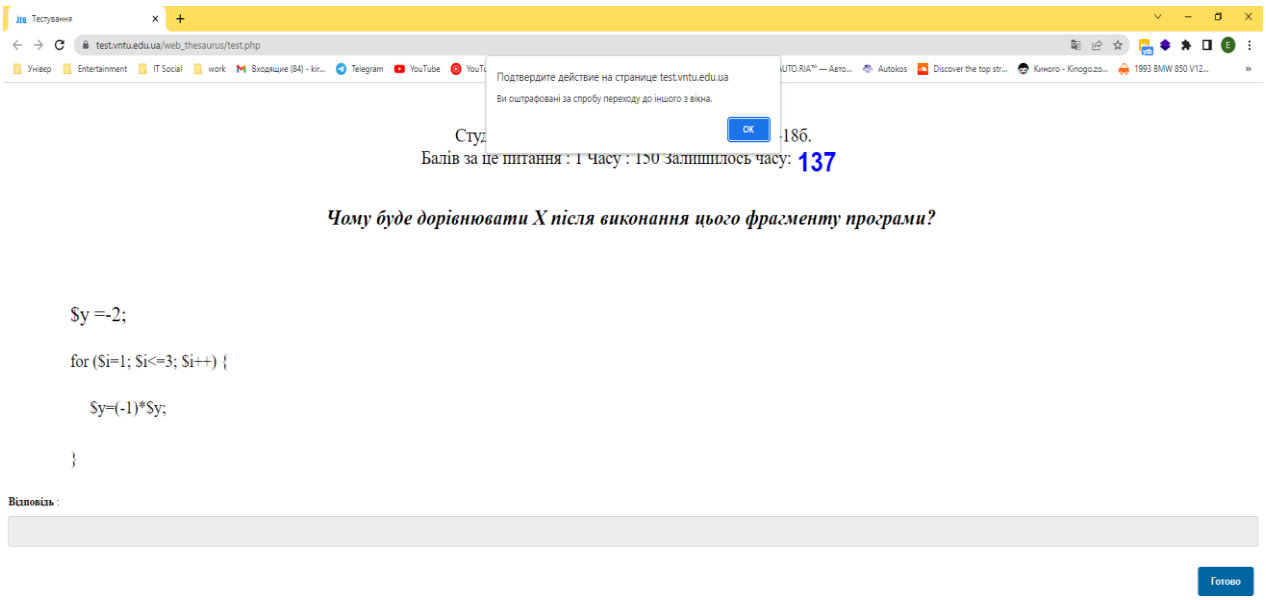


Рисунок 1.1 – Процес проходження тесту та виявлення шахрайства

Google Forms – веб додаток для створення та проходження опитування та тестів, який було створено Google та входить до складу безкоштовного вебпакету Google Docs Editors [5]. 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.

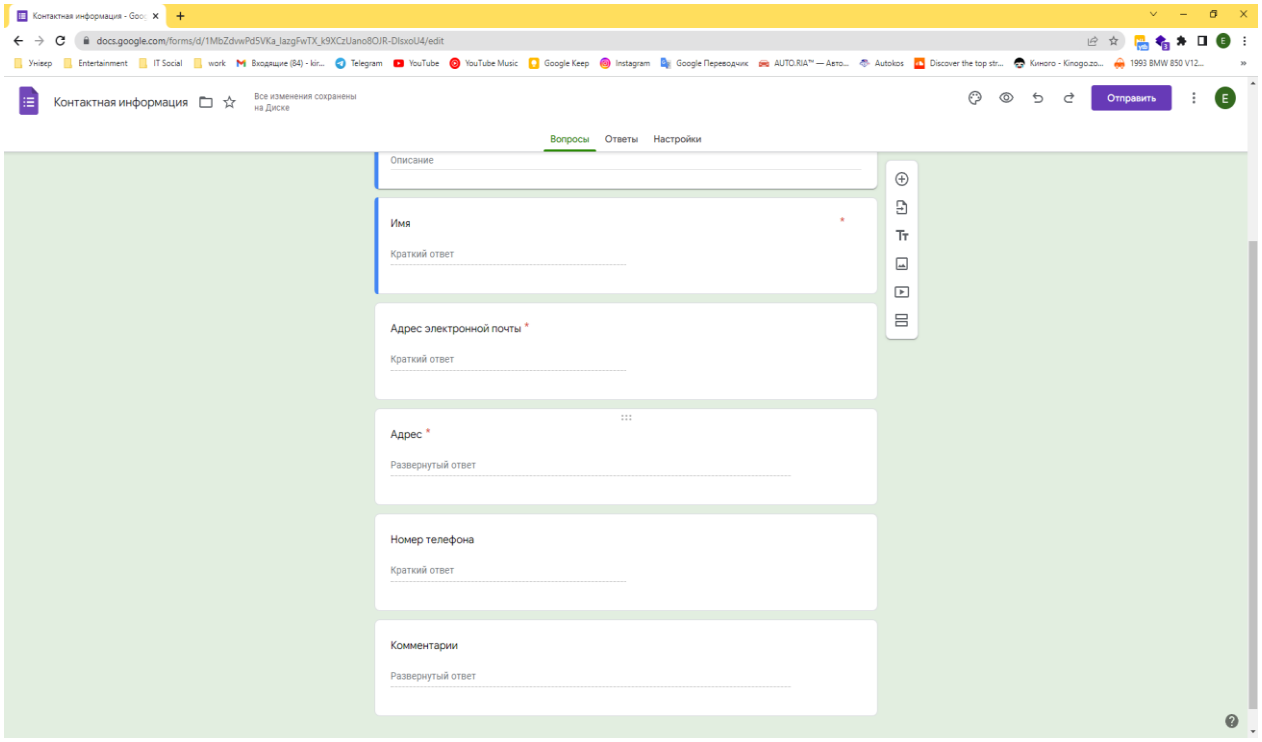


Рисунок 1.2 – Процес створення тесту Google Forms

Online Test Pad – простий та зручний сервіс для створення тестів та проведення тестування [6]. 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).

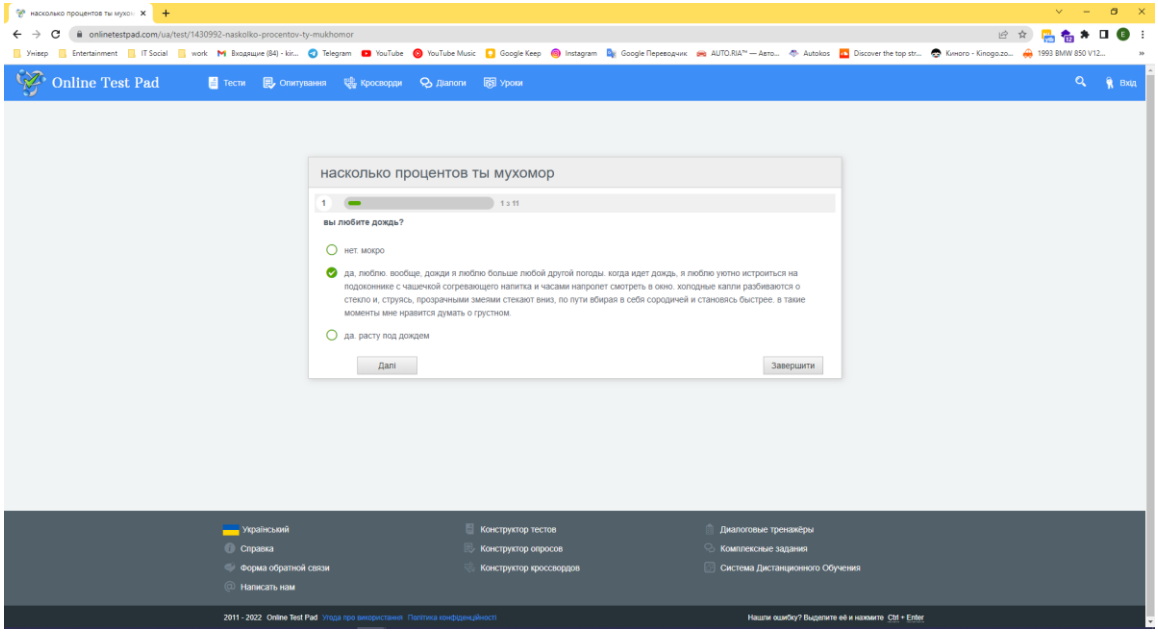


Рисунок 1.3 – Процес проходження розважального тесту на Online Test Pad

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

Таблиця 1.1 – Порівняльна таблиця аналогів та розроблюваного додатку.

	JetIQ	Online Test Pad	Google Forms	Quizzy
Загальнодоступний	–	+	+	+
Створення розважальних тестів	–	+	–	–
Фото контроль проходження тесту	–	–	–	+
Створення власних тестів	–	+	+	+
Статистика по проходження тесту	+	+	+	+

Продовження таблиці 1.1

	JetIQ	Online Test Pad	Google Forms	Quizzy
Часовий контроль проходження окремого питання	–	+	–	+
Орієнтованість на навчальні заклади	+	–	–	+
Результат:	2	5	3	6

Отже, проаналізувавши таблицю 1.1 було 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. 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 [7]. На рисунку 1.4 представлено приклад каскадної моделі життєвого циклу

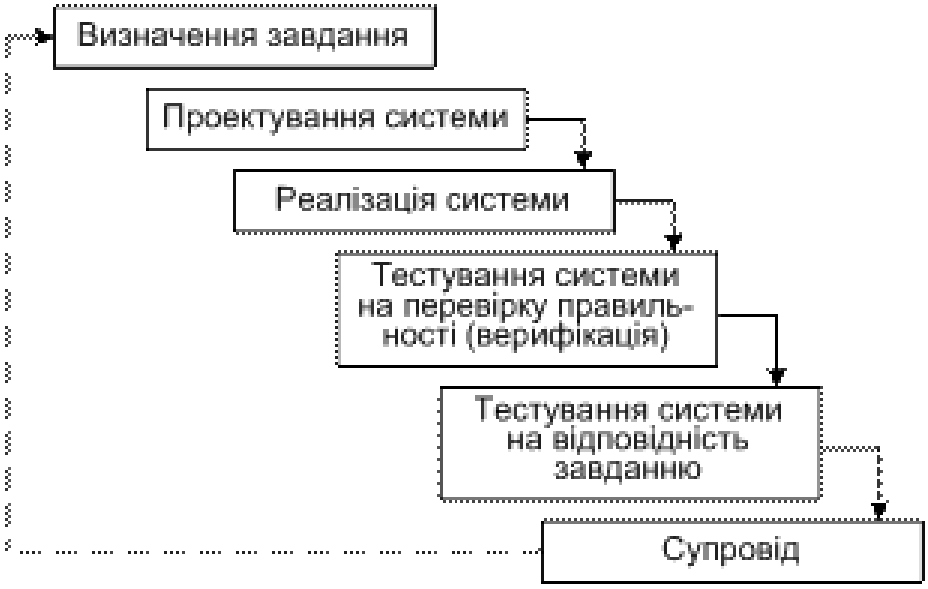


Рисунок 1.4 – Приклад каскадної моделі життєвого циклу програмного додатку

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 [8].

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

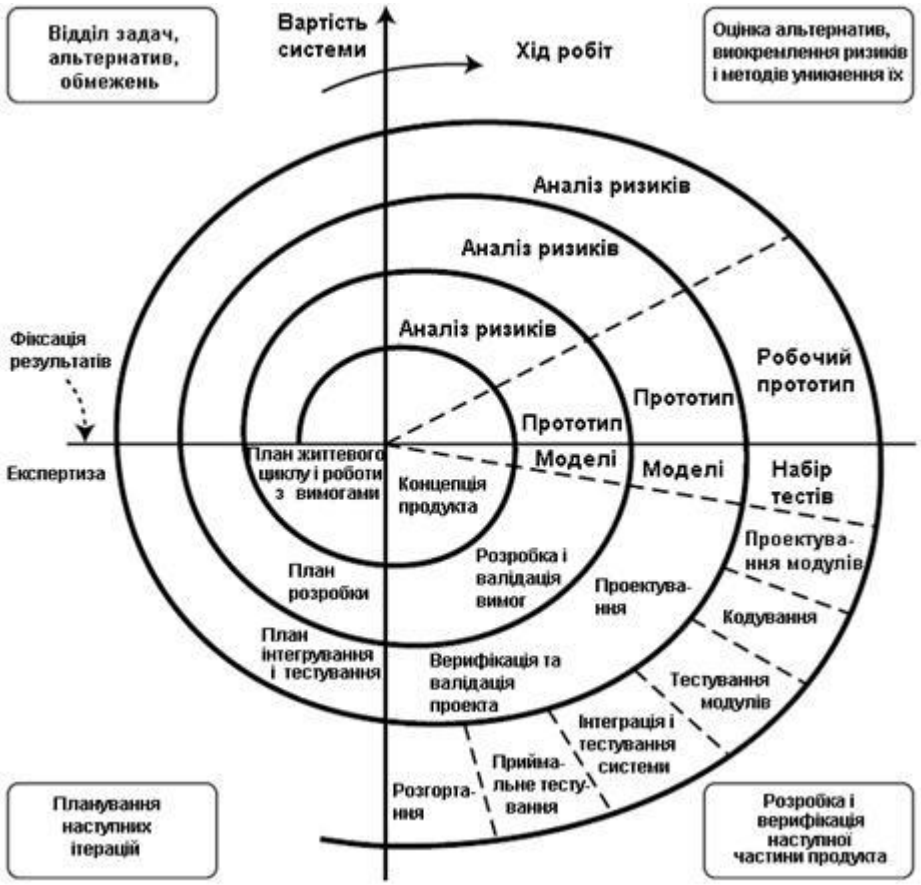


Рисунок 1.5 – Приклад спіральної моделі життєвого циклу програмного додатку

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 для адаптивної тестувальної системи з фото контролем Quizzy передбачає можливість створення та проходження тестів з фотофіксацією. Можливість зберігати дані про проходження тесту, перегляд історії проходження та статистики тесту. Також перегляд фото звітів проходження тестів.

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

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 було визначено основні задачі, які повинен виконувати веб додаток Quizzy, також визначено рівні доступу користувачів.

1.5 Висновки

У цьому розділі було проведено аналіз стану проблеми, обґрунтовано потребу у розробці веб застосунку для проходження адаптивних тестів з фото контролем.

Було розглянуто та проаналізовано аналоги веб застосунку, серед розглянутих були наступні додатки: JetIQ, Google Forms, Online Test Pad розглянуті веб додатки також мають задачі, які вони можуть виконувати, також співпадають з задачами, які повинен виконувати розроблюваний додаток.

Проаналізувавши аналоги, було визначено, що розроблюваний веб додаток набрав більшу кількість балів, зокрема через свою направленість на навчальні заклади та функціонал з фото контролем, що робить його кращим серед розглянутих.

Розроблюваний додаток матиме схожі з розглянутими аналогами функціонал, матиме фото контроль проходження тесту, буде реалізований на українській мові та буде безкоштовним. Було визначено задачі, які повинен виконувати веб додаток, розподілено рівні доступу користувачів.

2 РОЗРОБКА АРХІТЕКТУРИ ТА АЛГОРИТМІВ ПРОГРАМНОГО ПРОДУКТУ

2.1 Розробка загальної архітектури додатку

Для ефективної розробки додатку «Quizzy», потрібно визначитись та побудувати його загальну архітектуру. 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 [9]. 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.

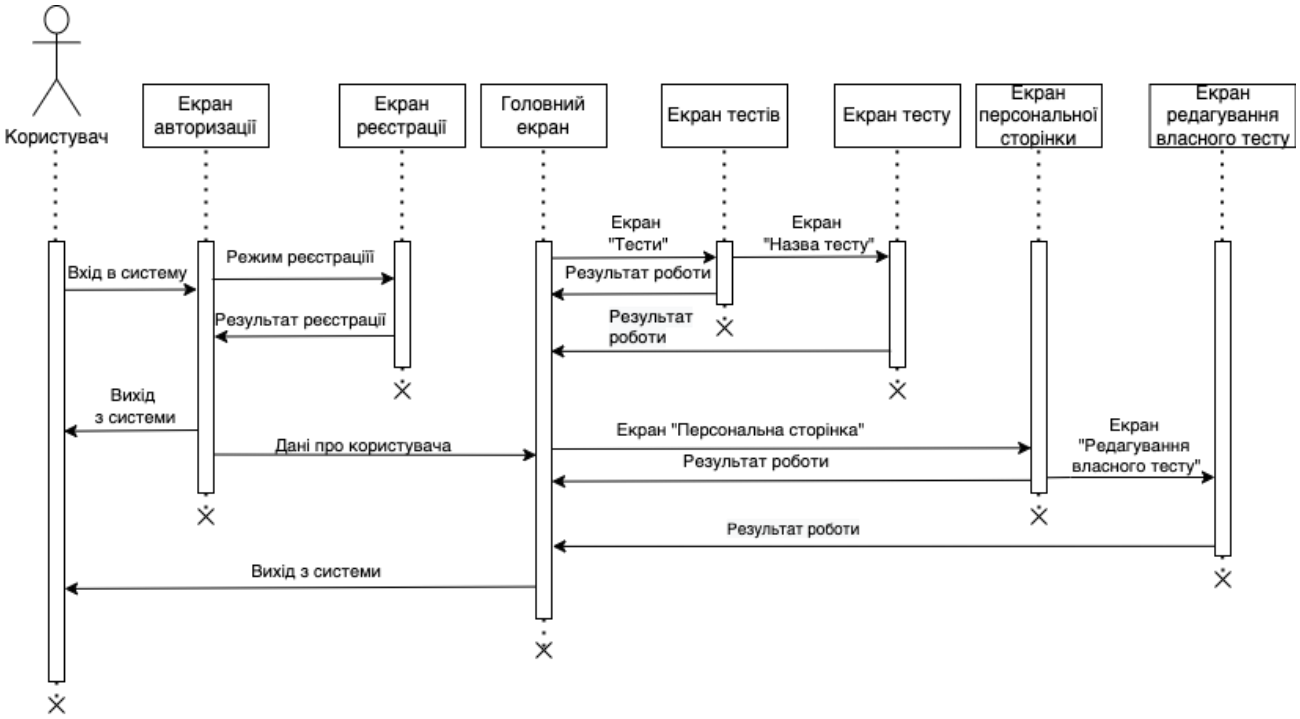


Рисунок 2.2 – Діаграма послідовності програмного додатку

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 [10]. Розроблену структурну карту Константайна представлено на рисунку 2.3

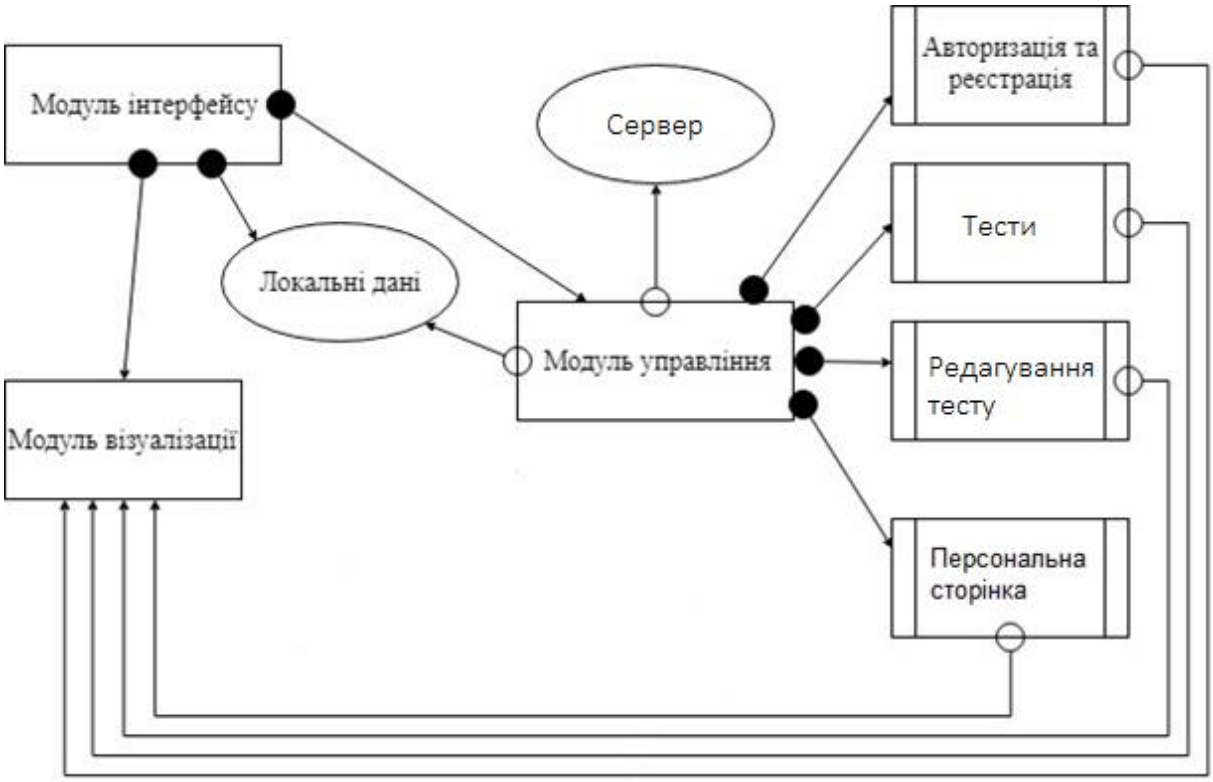


Рисунок 2.3 – Структурна карта Константайна для веб додатку

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.

Розроблені діаграми описують загальну архітектуру розроблюваного додатку і будуть використовуватись під час його програмної реалізації.

2.2 Розробка користувацького інтерфейсу

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.

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 [11].

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.

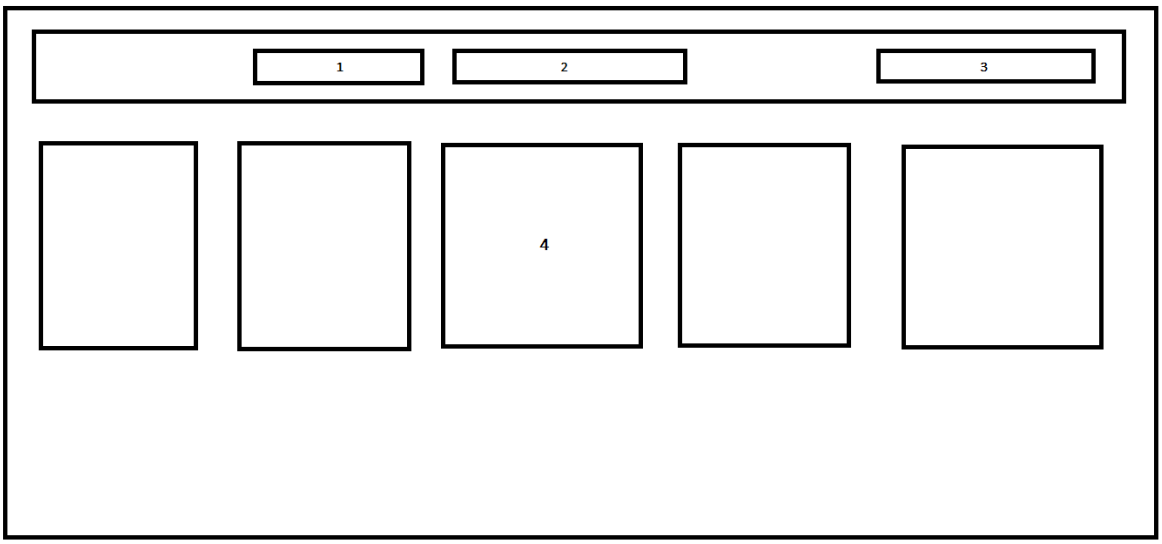


Рисунок 2.4 – Схема інтерфейсу головного екрану

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

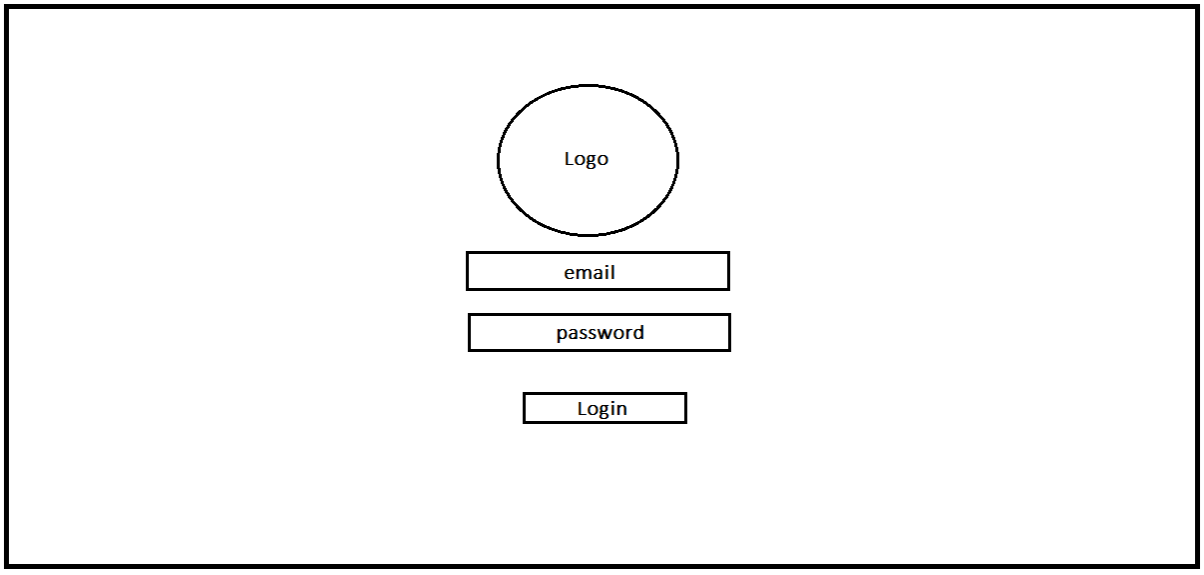


Рисунок 2.5 – Схема інтерфейсу авторизації та реєстрації в веб додатку

Важливим атрибутом веб додатку є екран з проходженням тесту, представлено на рисунках 2.6.

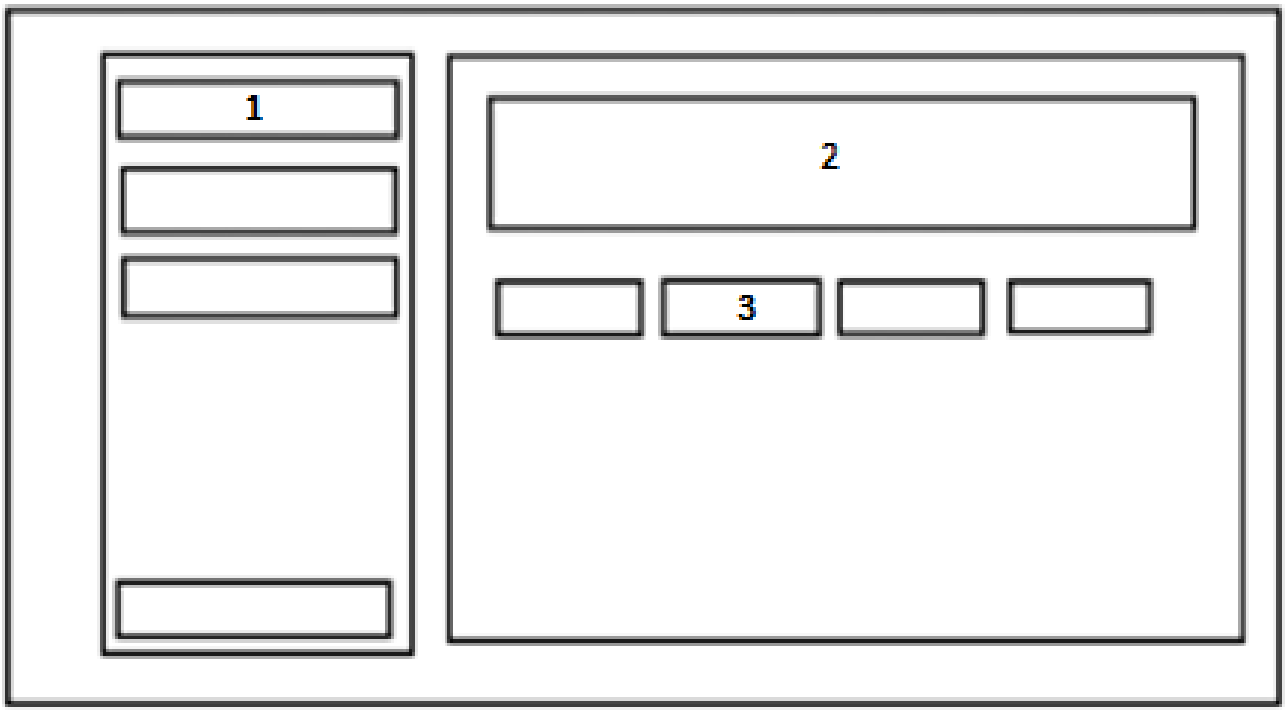


Рисунок 2.6 – Схема інтерфейсу проходження тесту

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, який представлений на рисунку 2.7.

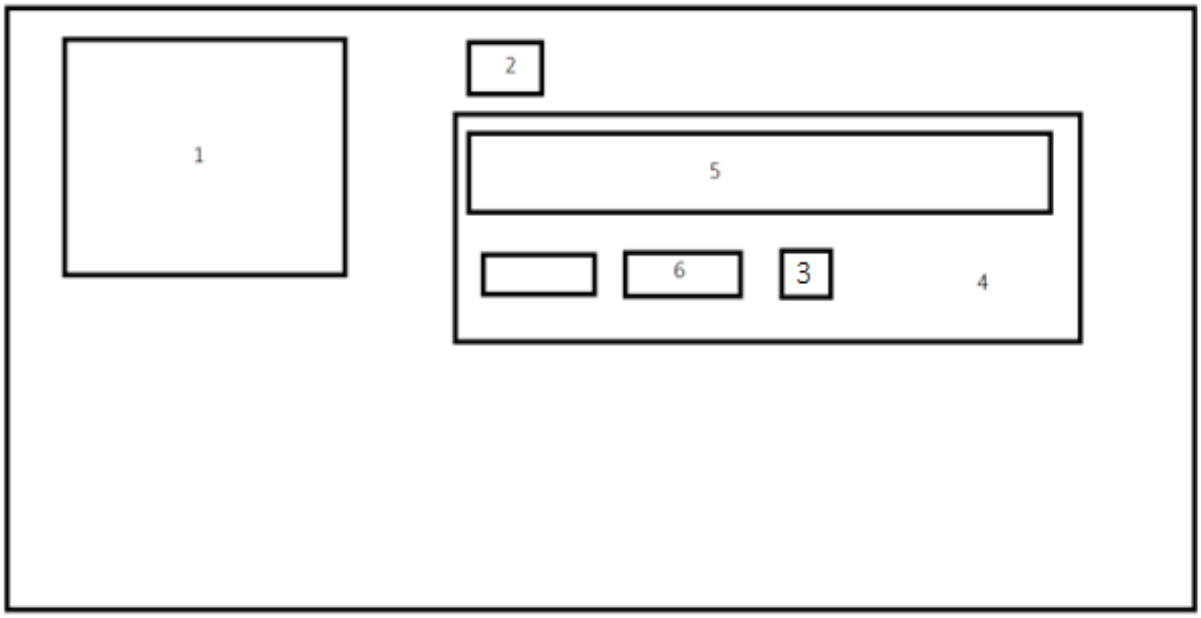


Рисунок 2.7 – Схема інтерфейсу редагування та створення тесту

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.

2.3 Розробка блок-схеми алгоритму роботи додатка

Наступним кроком буде розробка алгоритму роботи програмного додатку, на основі якого буде писатися програмний код. Блок–схема алгоритму роботи програмного додатку представлено на рисунку 2.8.

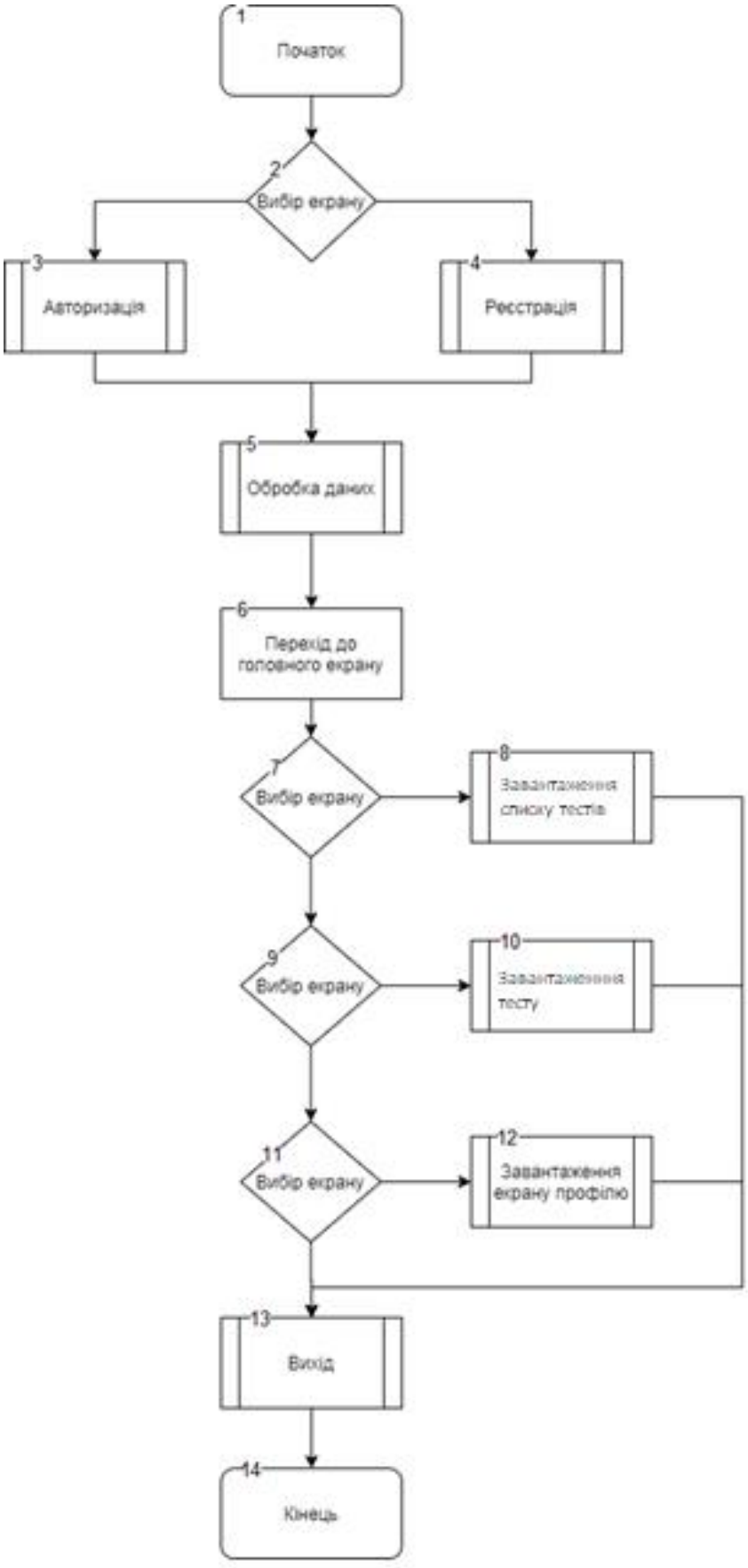


Рисунок 2.8 – Загальний алгоритм роботи програмного додатку

2.4 Розробка блок-схеми алгоритму роботи фотоконтролю

Фотоконтроль - спосіб контролю за результатами використання засобів ураження, що здійснюється за допомогою фотоапаратури.

Захоплення потокового відео та аудіо є пріоритетним напрямком у розробці веб-застосунків протягом багатьох років. На допомогу прийшов HTML5. Можливо, цей момент не є очевидним, але поширення HTML5 порушило питання доступу до апаратних ресурсів. Прекрасними прикладами цього є функції Geolocation (GPS), Orientation API (акселерометр), WebGL (графічні процесори) та API веб-аудіо (звукове обладнання). Ці розробки дуже багатофункціональні. Вони створені на основі API JavaScript високого рівня та спираються на апаратні можливості вихідного пристрою [12].

Наступним кроком буде розробити алгоритм проходження тесту та здійснення фотофіксації. Блок-схема алгоритму проходження тесту та здійснення фотофіксації представлено на рисунку 2.9. 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.

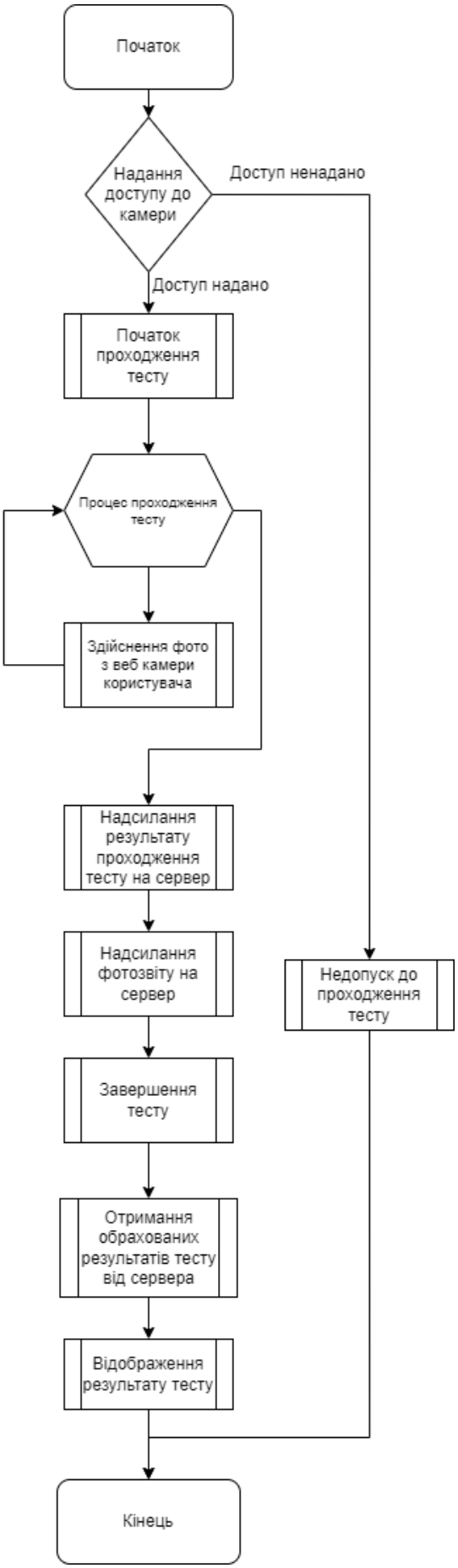


Рисунок 2.9 – Загальний алгоритм проходження тесту з фотоконтролем

2.5 Висновки

У цьому розділі було розроблено загальну архітектуру веб додатку, було розроблено діаграми послідовності, діаграми варіантів та структурну карту Константайна. Було розроблено схеми користувацького інтерфейсу для кожного екрану присутнього в веб додатку. Також було побудовано загальний алгоритм роботи програмного додатку та проходження тесту з фотофіксацією.

3. РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

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.

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 [13].

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 [14].

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.

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 [16].

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 було розроблено більшість сучасних веб додатків.

3.2 Обґрунтування вибору середовища розробки

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 [17]. 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.

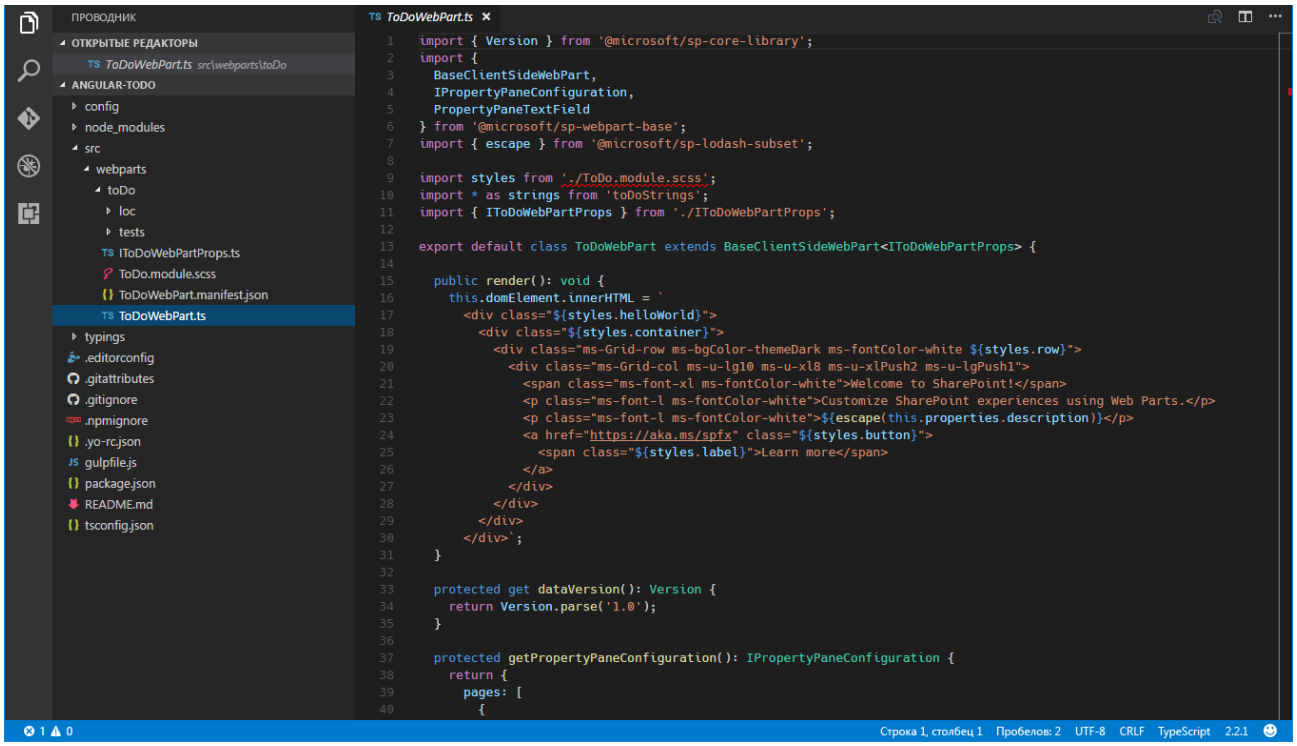


Рисунок 3.1 – Приклад інтерфейсу Visual Studio Code

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 [18].

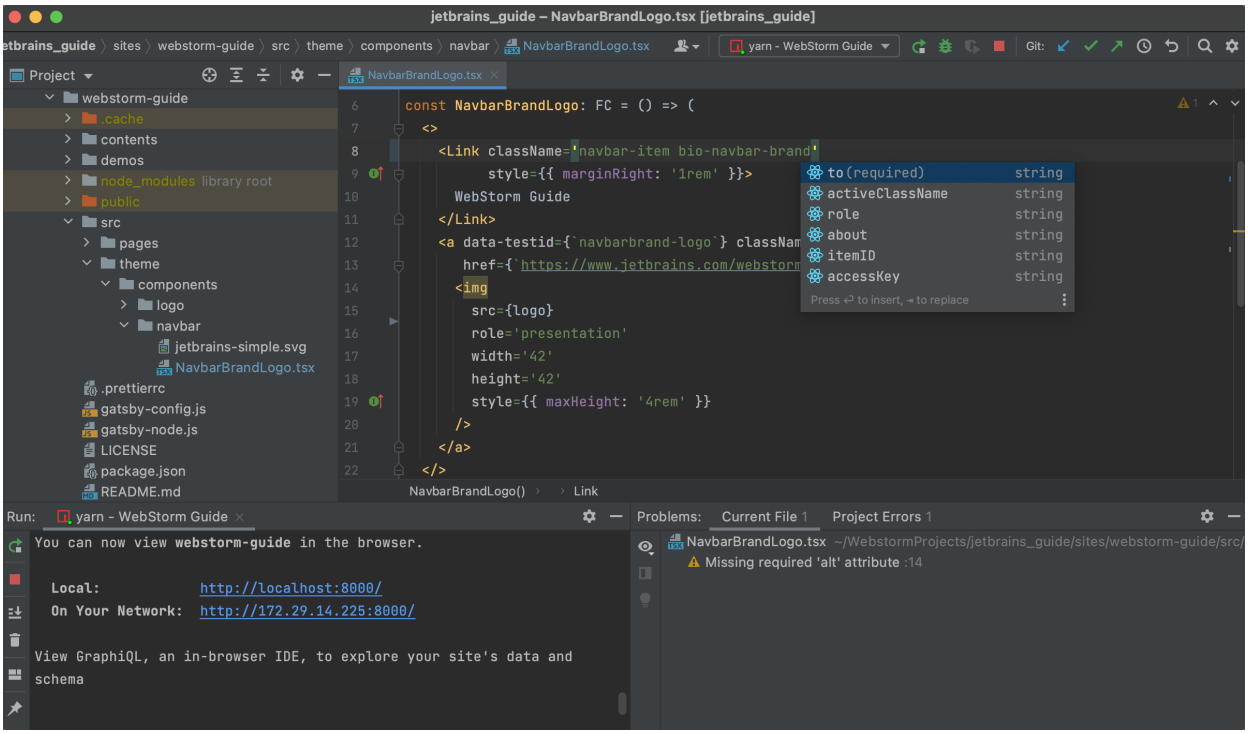


Рисунок 3.2 – Приклад інтерфейсу WebStorm

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

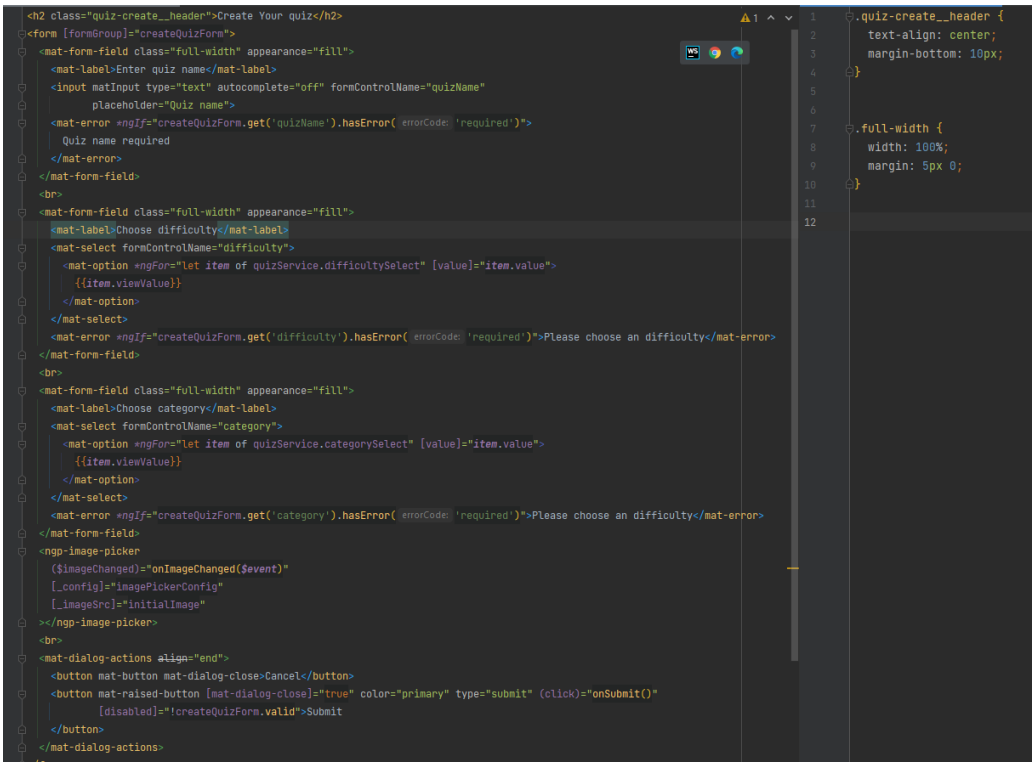
3.3 Програмна реалізація

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.3.3), функціональної частини компоненти та для сервісу, в якому була реалізована уся логіка, що пов’язана з створенням та редагуванням тесту, було виконано на мові програмування TypeScript рисунок 3.4 та 3.5 [19].

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 [20]. Angular – це AngularJS, який був переосмислений та перероблений тією ж командою розробників. Angular використовує принципи сервісо-орієнтованого та компонентного програмування [21].

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 [22].

Сервісно-орієнтована архітектура – архітектурний шаблон програмного забезпечення, модульний підхід до розробки програмного забезпечення, заснований на використанні розподілених, слабко пов'язаних замінних компонентів, оснащених стандартизованими інтерфейсами для взаємодії за стандартизованими протоколами [23].

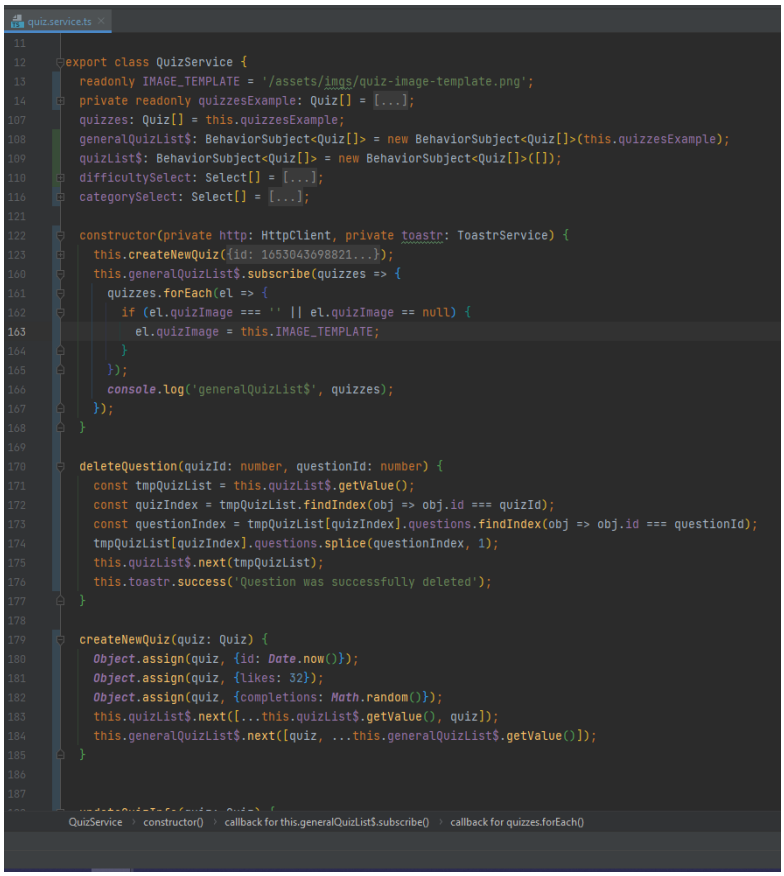


```
<h2 class="quiz-create__header">Create Your quiz</h2>
<form [formGroup]="createQuizForm">
  <mat-form-field class="full-width" appearance="fill">
    <mat-label>Enter quiz name</mat-label>
    <input matInput type="text" autocomplete="off" formControlName="quizName"
           placeholder="Quiz name">
    <mat-error *ngIf="createQuizForm.get('quizName').hasError( errorCode: '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( errorCode: '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( errorCode: '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>
```

```
.quiz-create__header {
  text-align: center;
  margin-bottom: 10px;
}

.full-width {
  width: 100%;
  margin: 5px 0;
}
```

Рисунок 3.3 – Приклад модулю візуалізації (компонента)



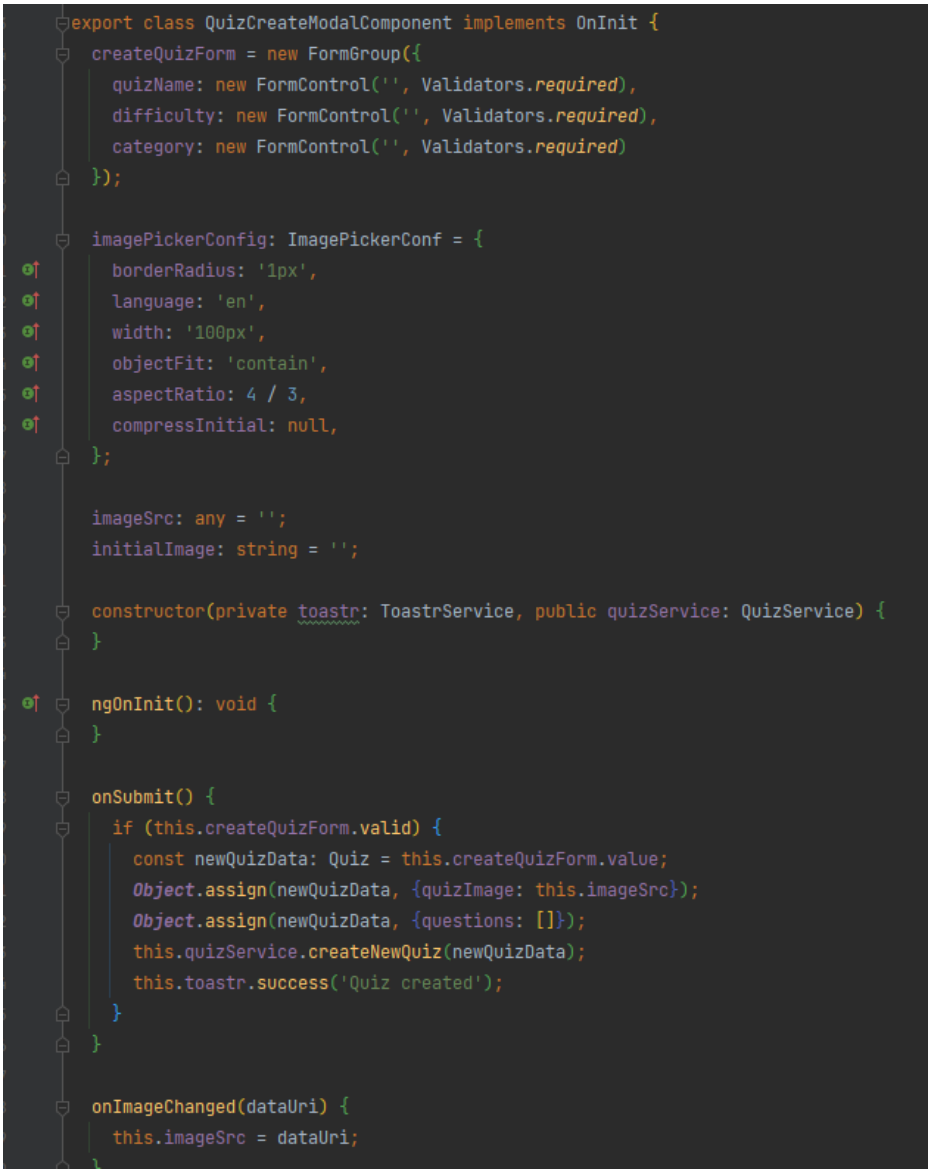
```
export class QuizService {
  readonly IMAGE_TEMPLATE = '/assets/imgs/quiz-image-template.png';
  private readonly quizzesExample: Quiz[] = [...];
  quizzes: Quiz[] = this.quizzesExample;
  generalQuizList$: BehaviorSubject<Quiz[]> = new BehaviorSubject<Quiz[]>(this.quizzesExample);
  quizList$: BehaviorSubject<Quiz[]> = new BehaviorSubject<Quiz[]>([]);
  difficultySelect: Select[] = [...];
  categorySelect: Select[] = [...];

  constructor(private http: HttpClient, private toastr: ToastrService) {
    this.createNewQuiz({id: 16530436988821...});
    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()]);
  }
```

Рисунок 3.4 – Приклад функціонального модулю (сервіс)



```
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;
  }
```

Рисунок 3.5 – Приклад функціонального модулю (компонента)

Також було розроблено програмну компоненту для фотоконтролю під час проходження тесту. Дана компонента має функціональний та модуль візуалізації, в якій прописаний основний функціонал фотоконтролю. Лістинг коду зображено на рисунку 3.6 та 3.7.

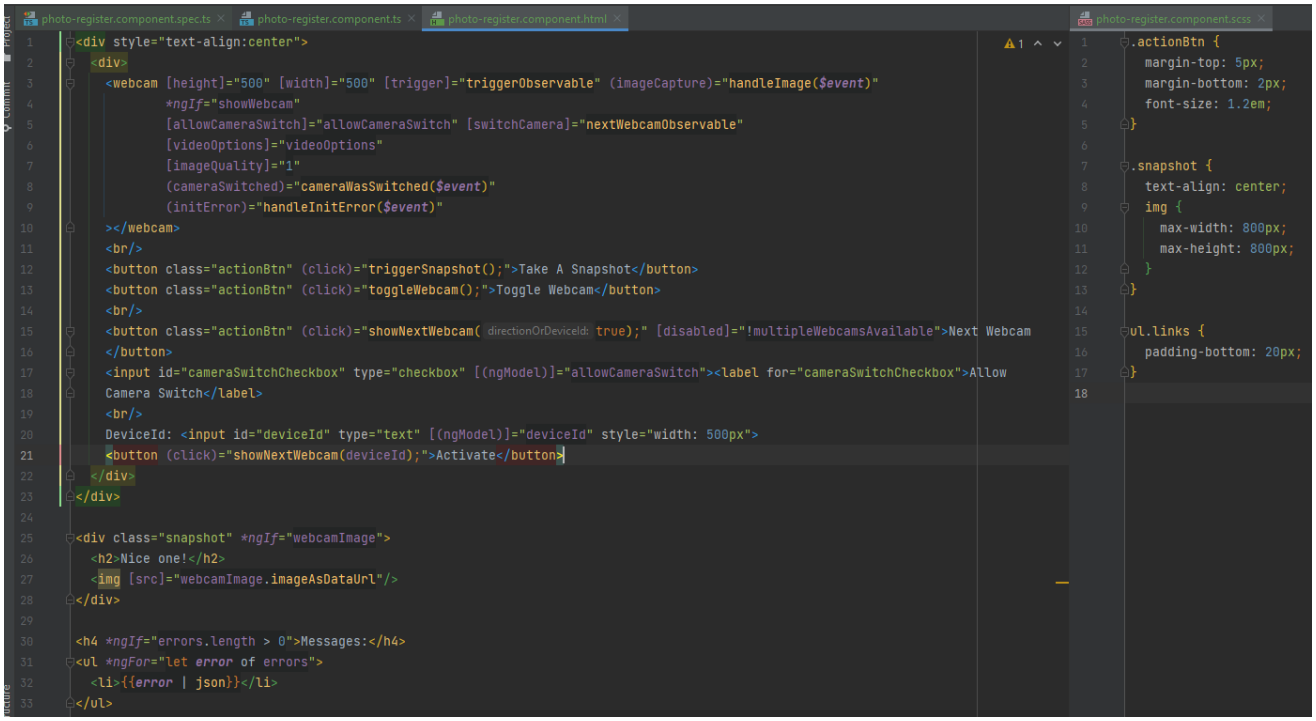


Рисунок 3.6 – Лістинг модулю візуалізації (компонента)

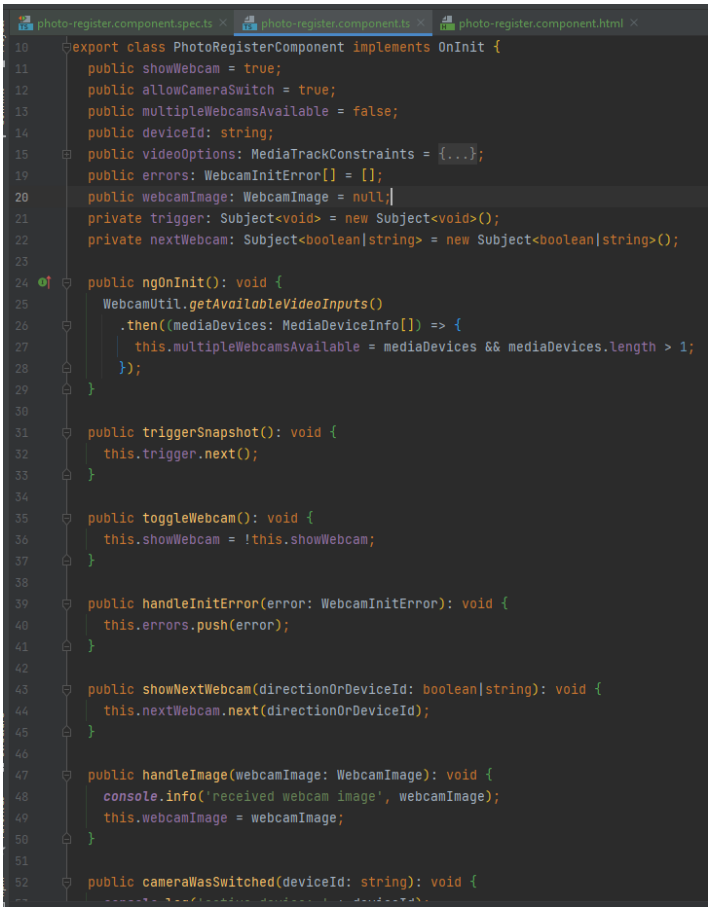


Рисунок 3.7 – Лістинг функціонального модулю (компонента)

3.4 Висновки

У даному розділі було проведено варіантний аналіз і обґрунтування вибору мови програмування. Було проаналізовано переваги й недоліки C++, Java та JavaScript/TypeScript. Для розробки програмного продукту було обрано мову JavaScript/TypeScript. Було досліджено ринок популярних IDE для цієї мови програмування, проаналізовано середовища розробки WebStorm та Visual Studio Code. На основі розглянутих критеріїв було прийнято рішення використовувати WebStorm для розробки додатку.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Тестування програмного забезпечення

Тестування програмного забезпечення – одна з найважливіших стадій його розробки. Вона допомагає виявити помилки, що не були знайдені під час розробки програмного забезпечення, а також перевіряє на відповідність додатку вимогам специфікації.

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.

Веб додаток – це по суті 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.

Отже протестуємо роботу веб додатку загалом. Відкриємо головну сторінку веб додатку у різних браузерах, щоб порівняти коректне відображення стилів рисунок. 4.1 та 4.2.

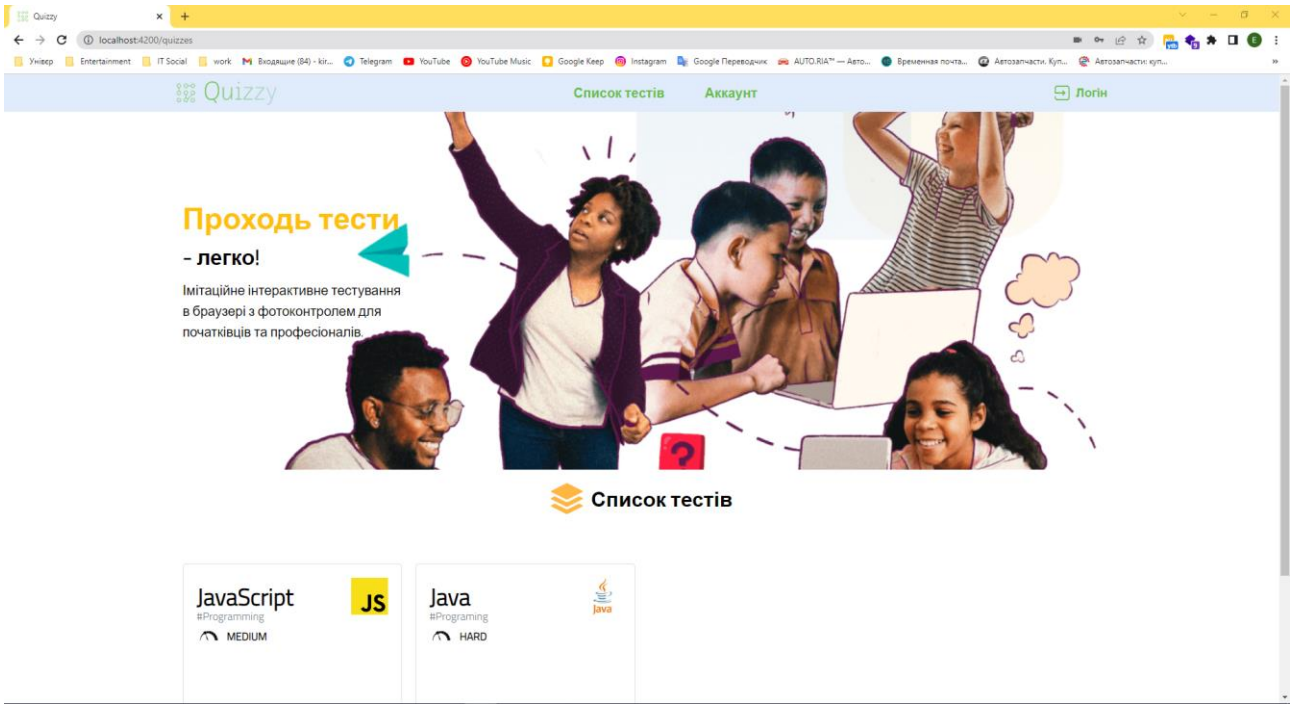


Рисунок 4.1 – Головне меню додатку в браузері Google Chrome

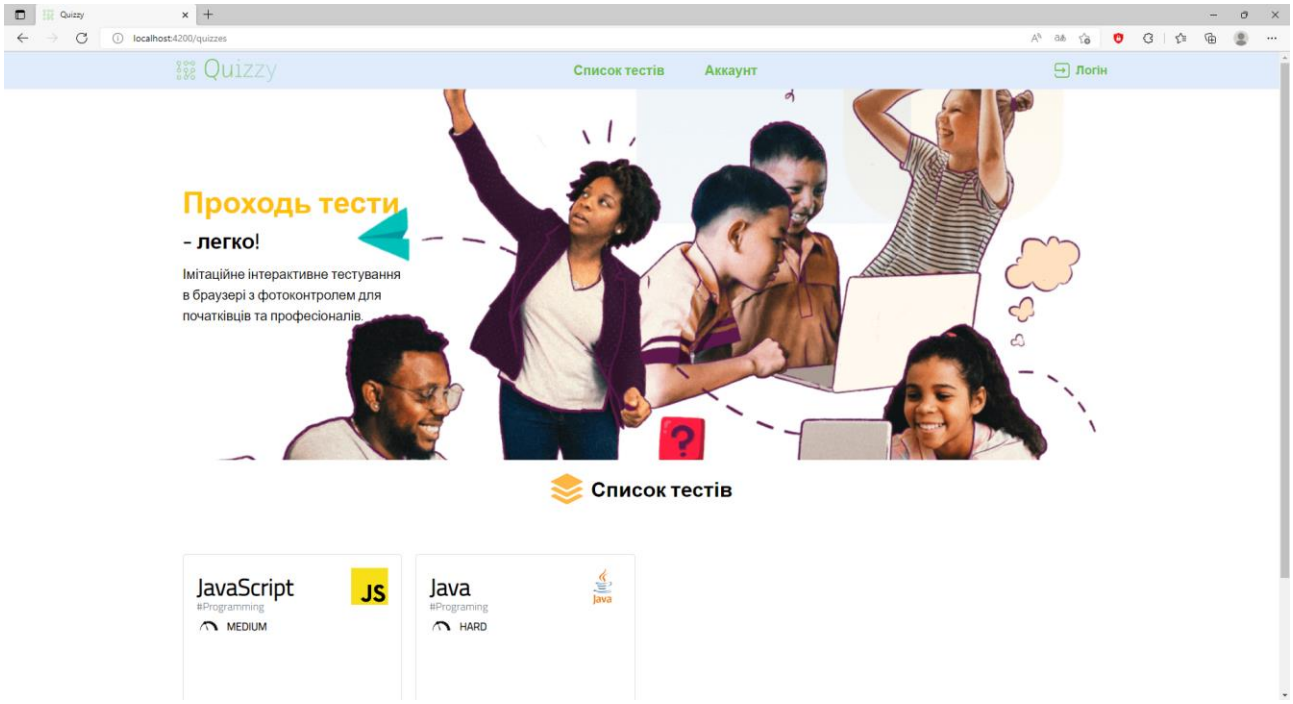


Рисунок 4.2 – Головне меню додатку в браузері Microsoft Edge

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

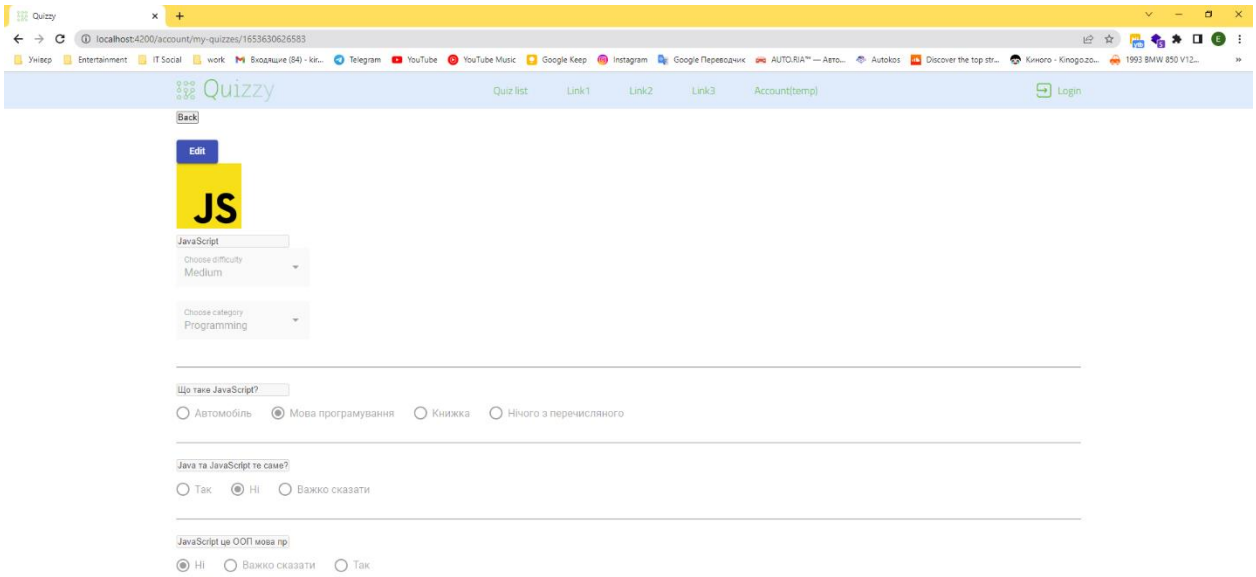


Рисунок 4.3 – Сторінка редагування тесту в неактивному стані

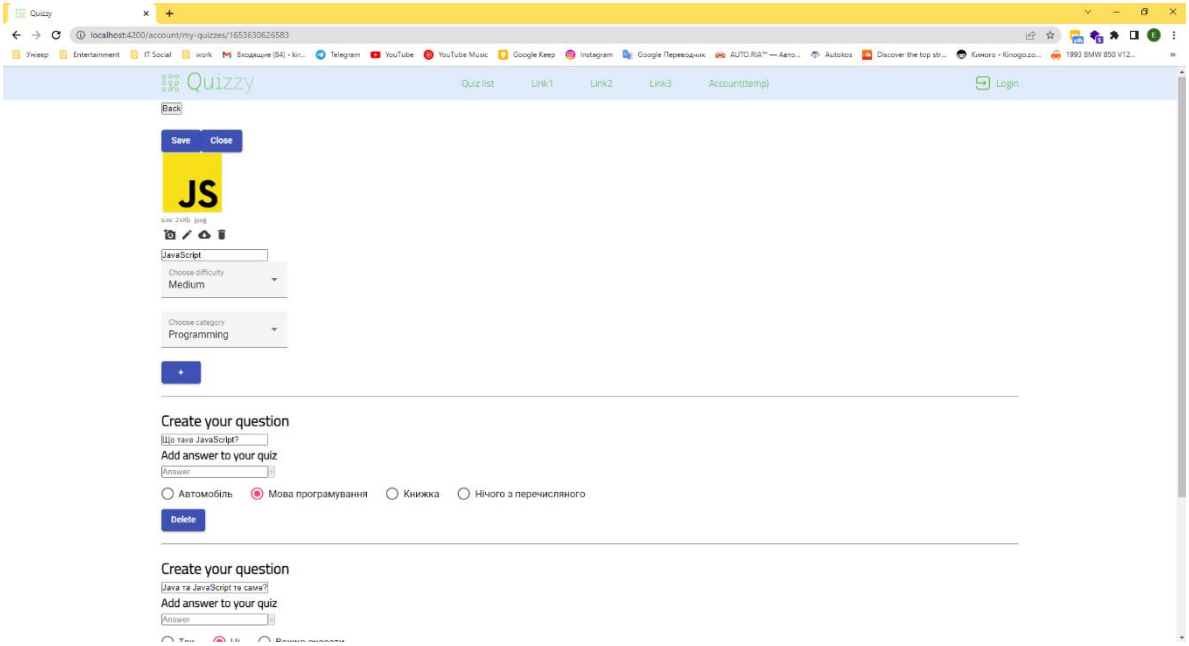


Рисунок 4.4 – Сторінка редагування тесту в активному стані

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.

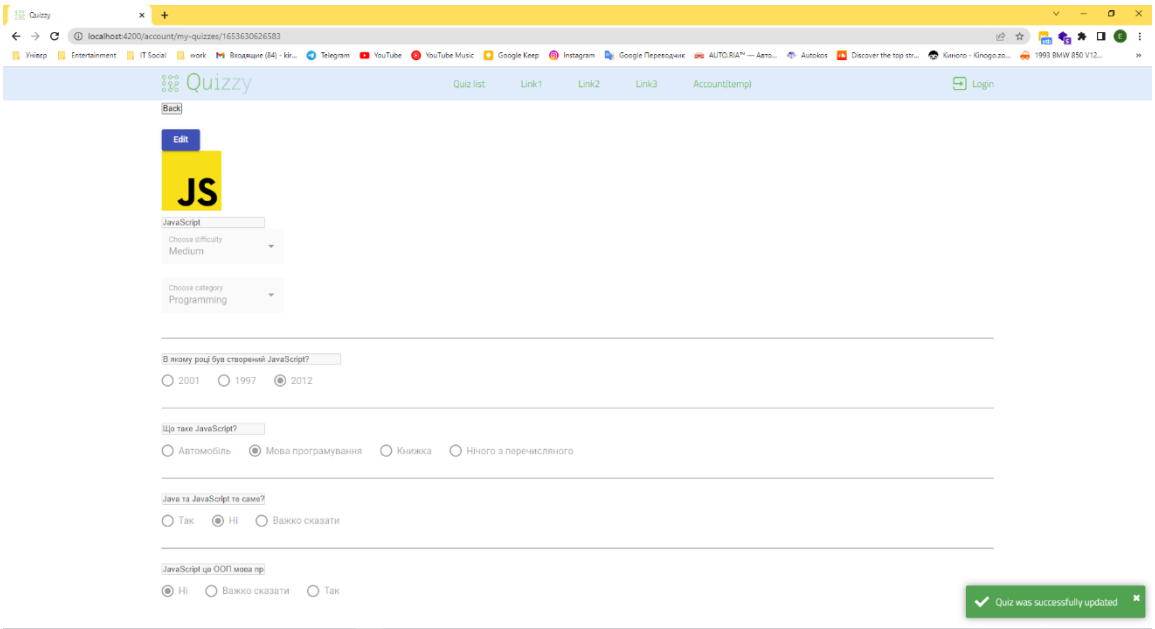


Рисунок 4.5 – Результат додання нового питання

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

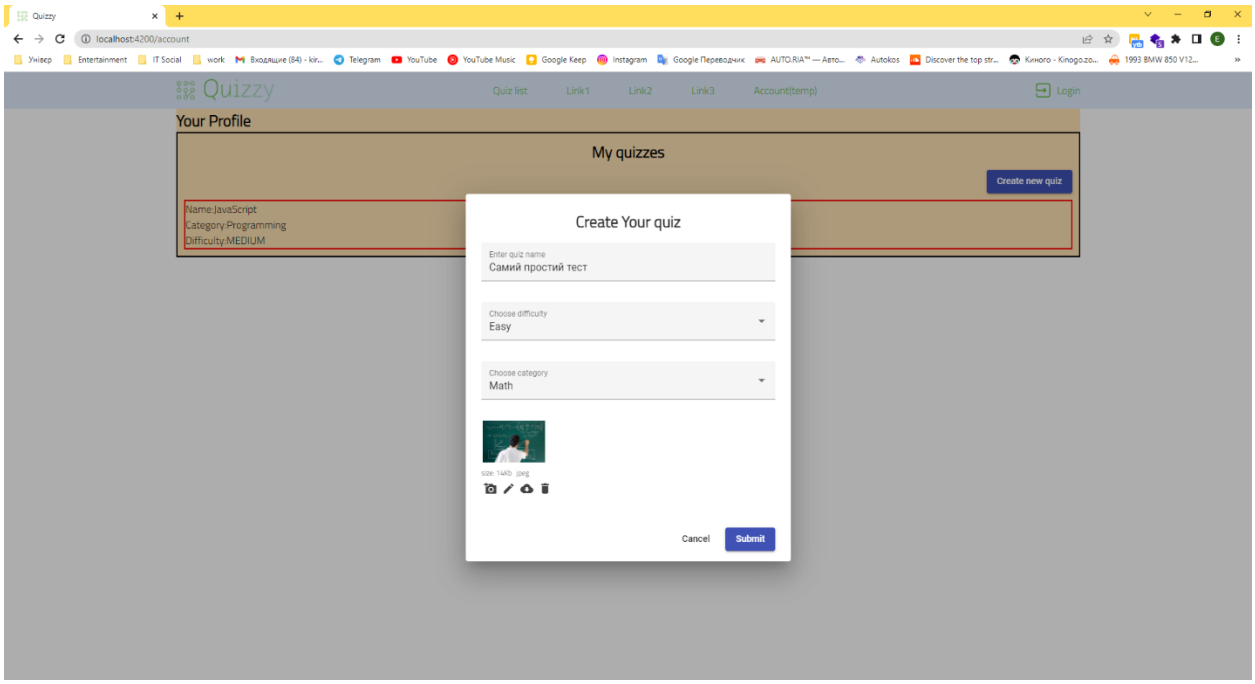


Рисунок 4.6 – Результат заповнення форми для створення тесту

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

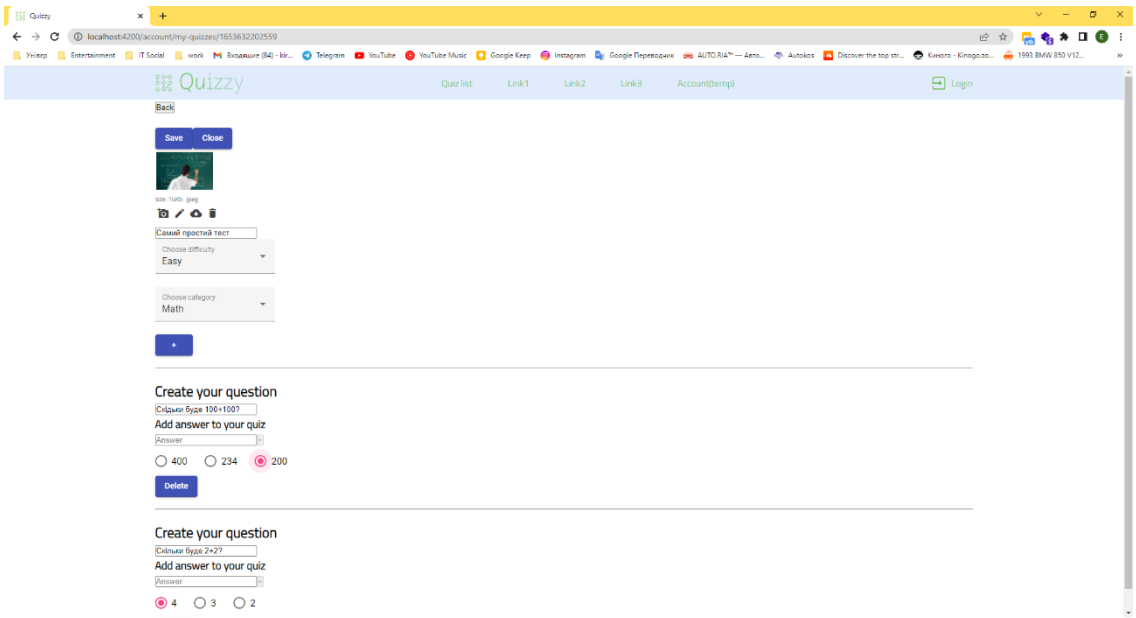


Рисунок 4.7 – Результат додання двох питань

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.

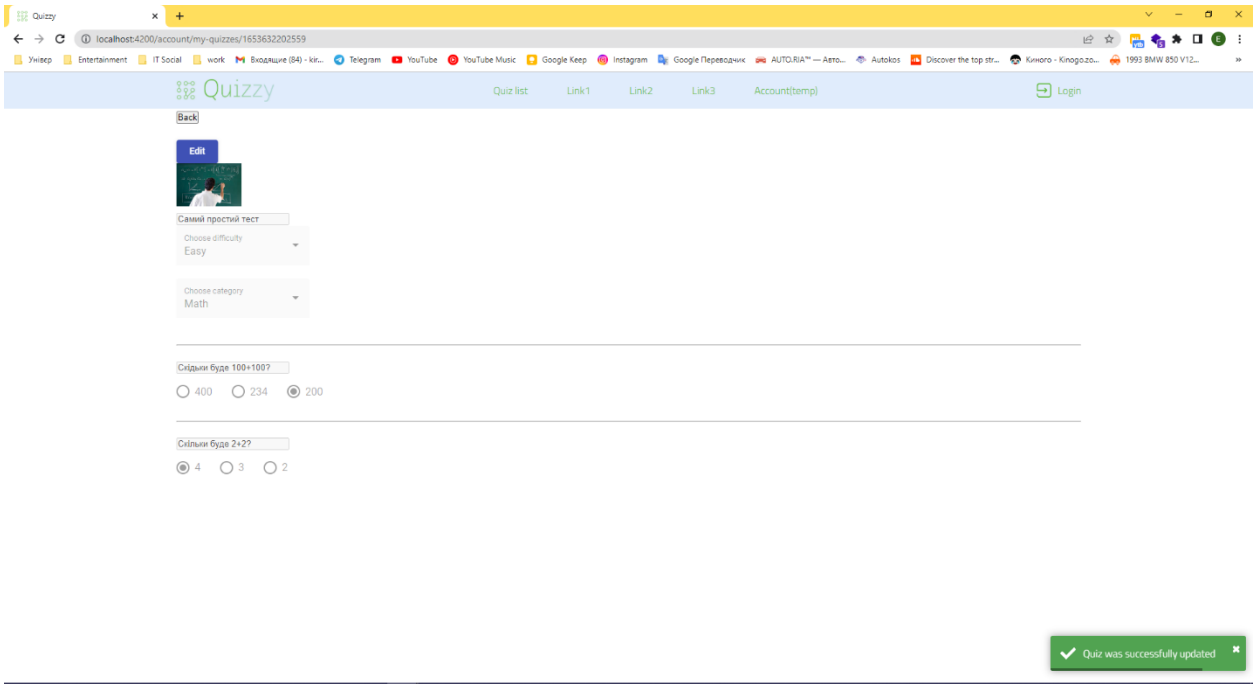


Рисунок 4.8 – Результат успішного редагування тесту

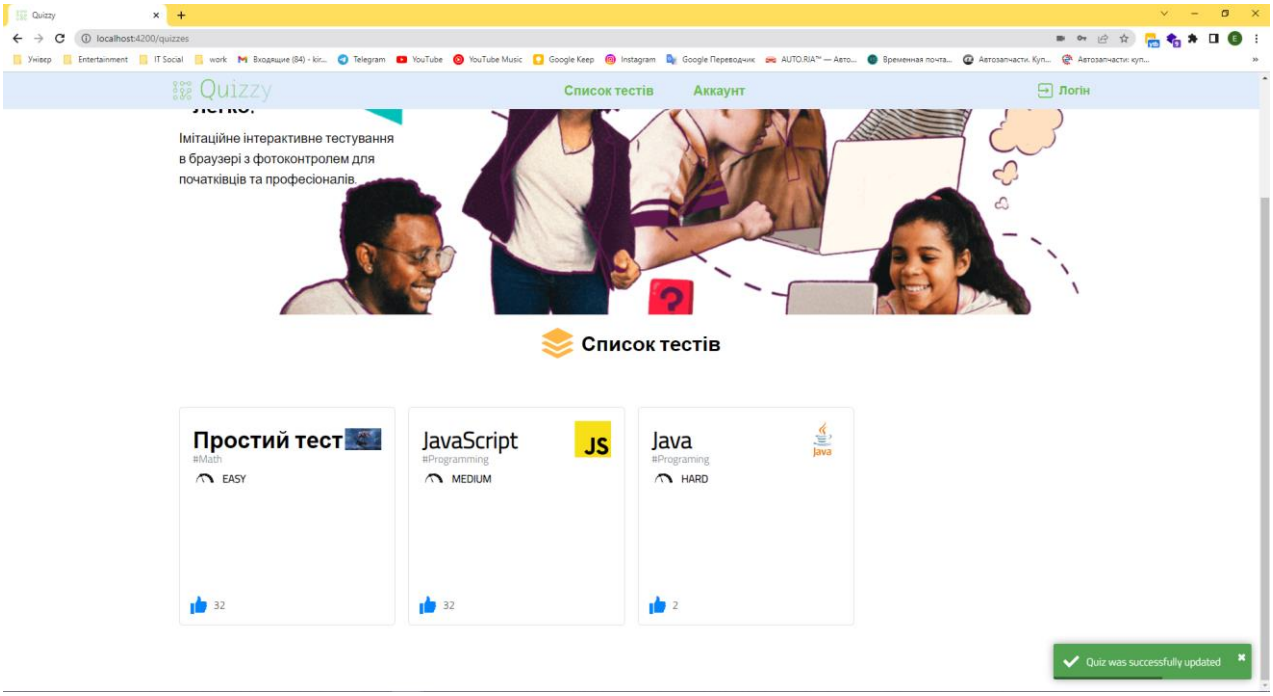


Рисунок 4.9 – Новостворений тест в основному списку тестів

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 [24].

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

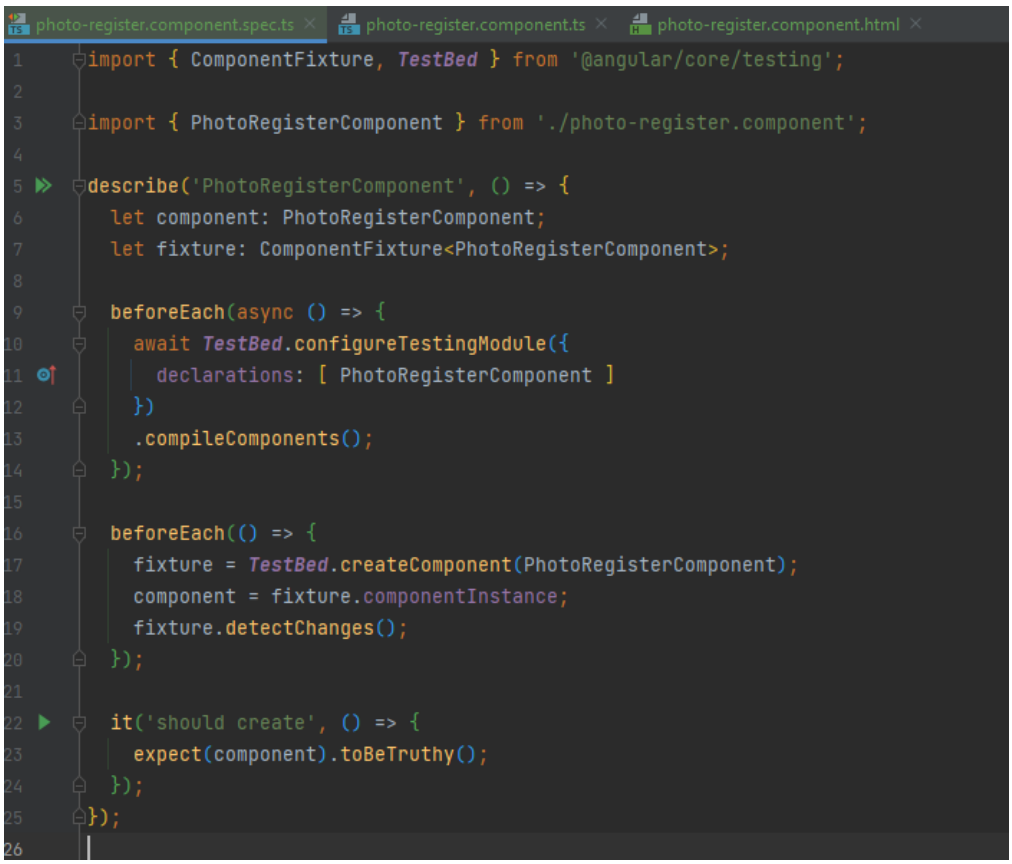


Рисунок 4.10 – Лістинг Unit тесту компоненти фотофіксації

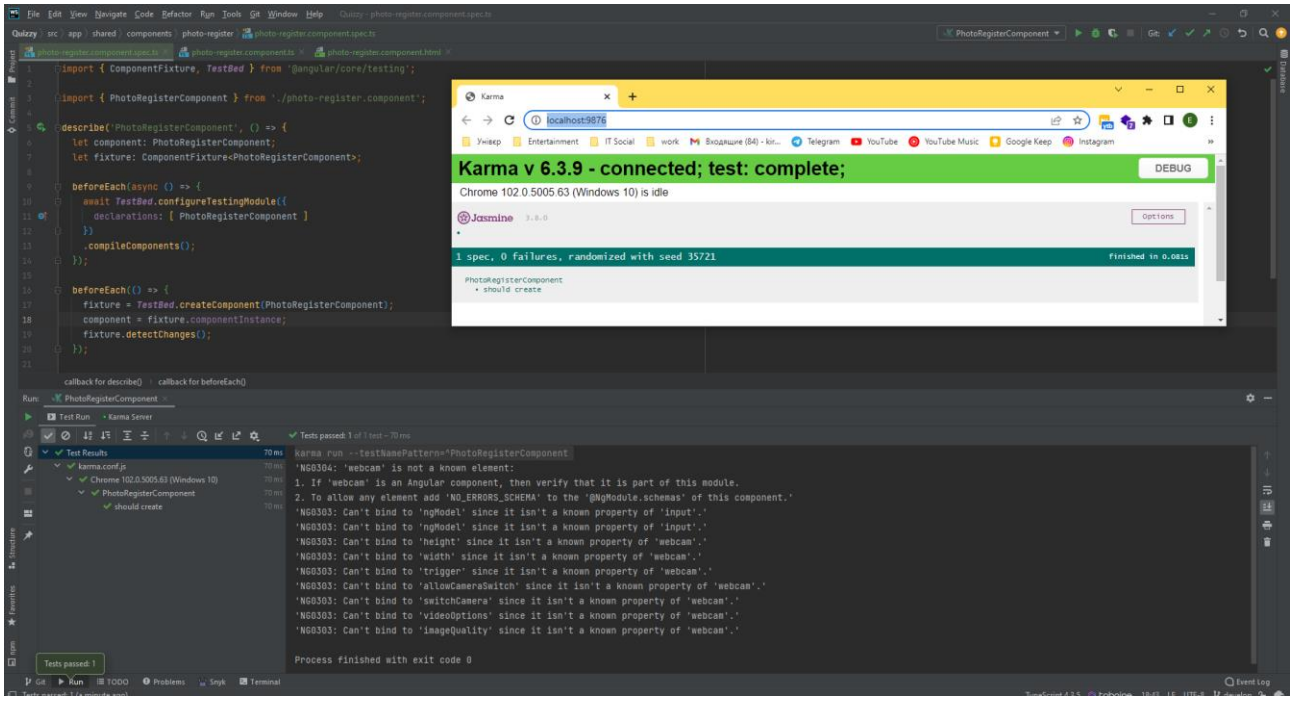


Рисунок 4.11 – Результат виконання Unit тесту компоненти фотофіксації

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 [25].

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.

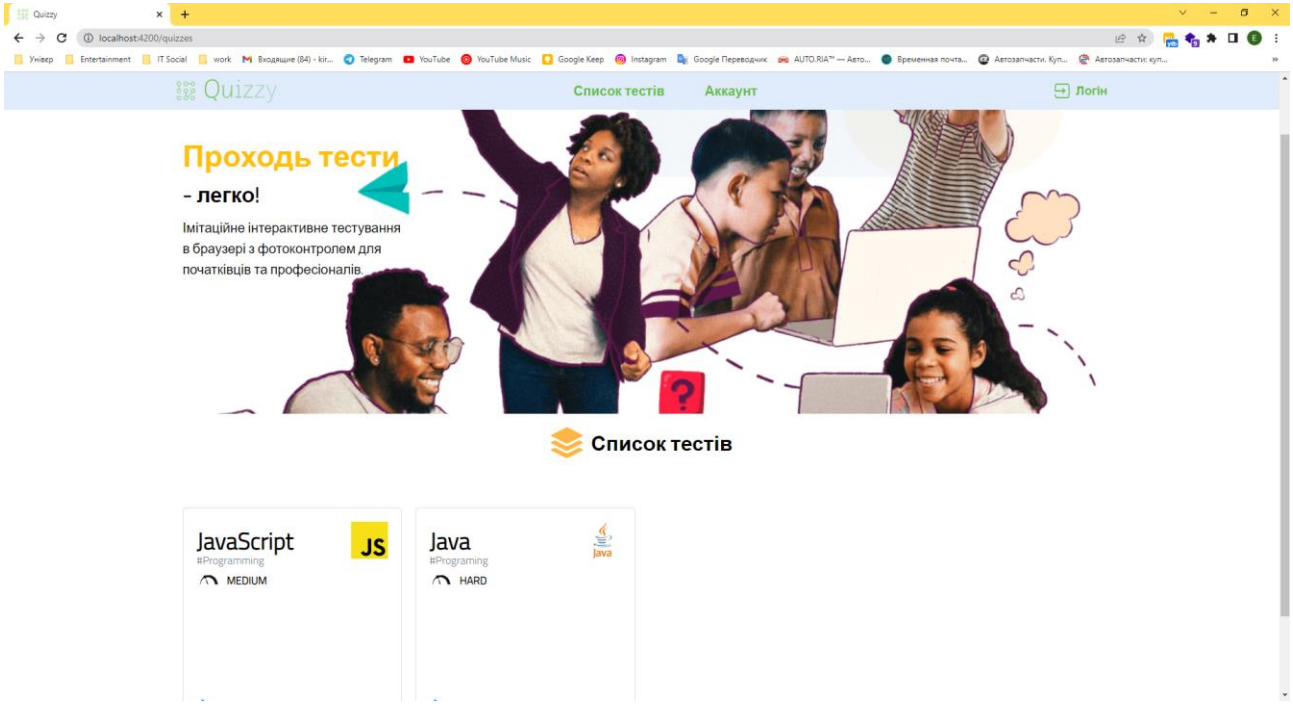


Рисунок 4.12 – Головна сторінка веб додатку

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.

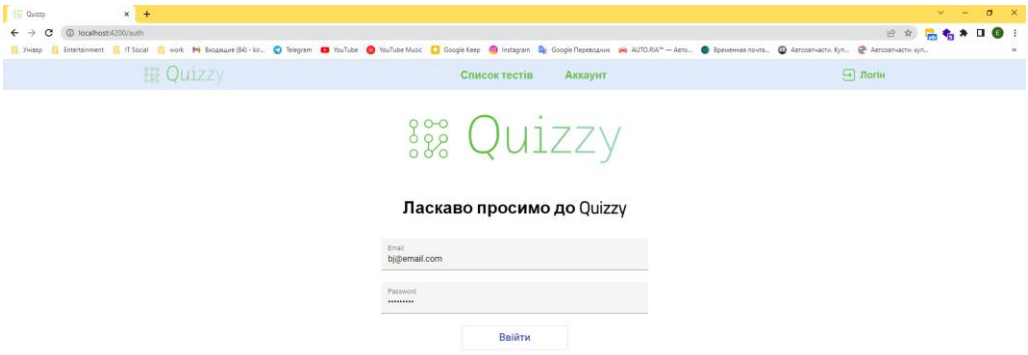


Рисунок 4.13 – Екран входу в персональний обліковий запис

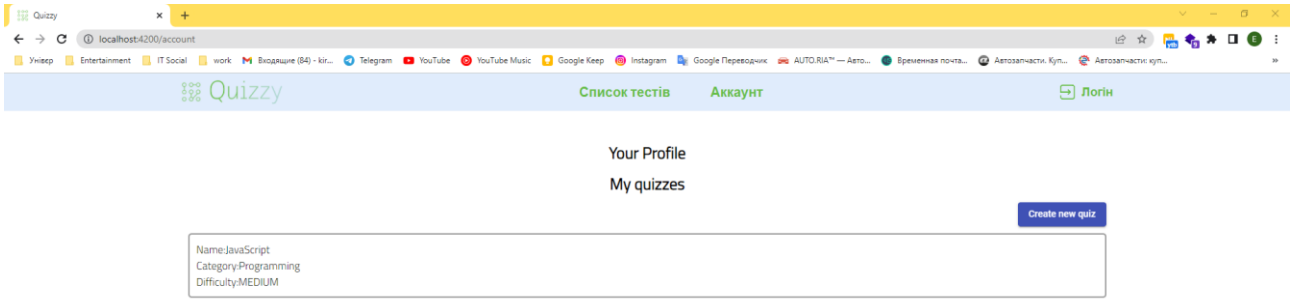


Рисунок 4.14 – Екран реєстрації персонального облікового запису

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.

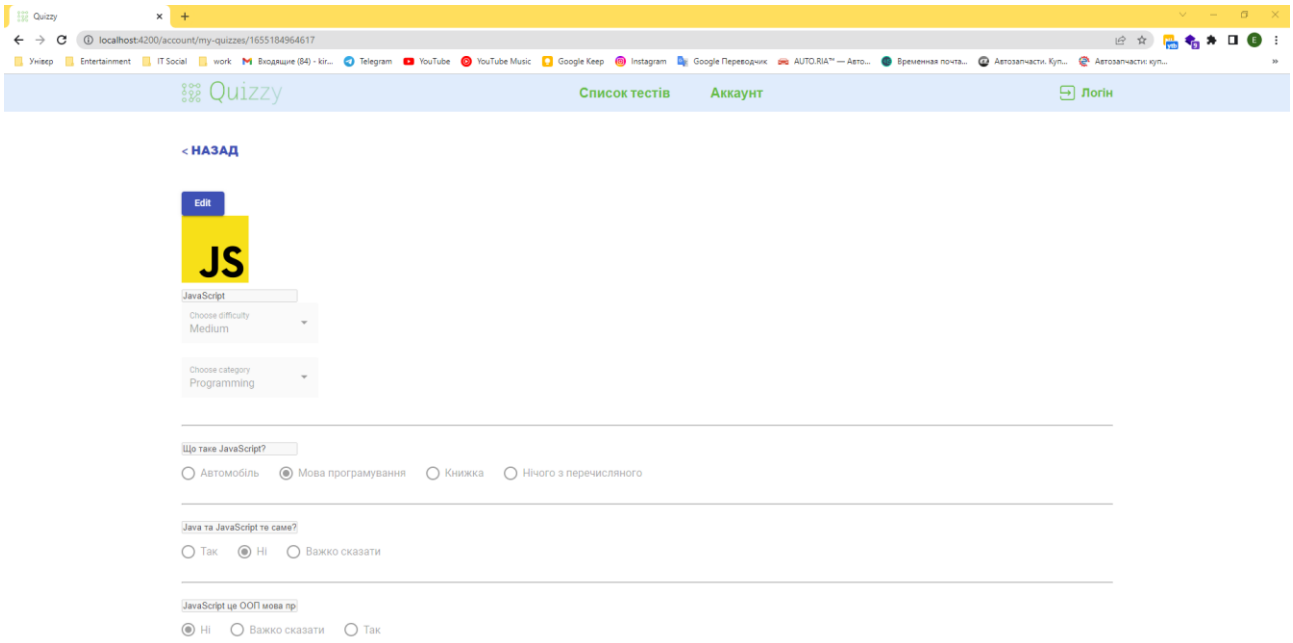


Рисунок 4.14 – Екран редагування тесту

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.

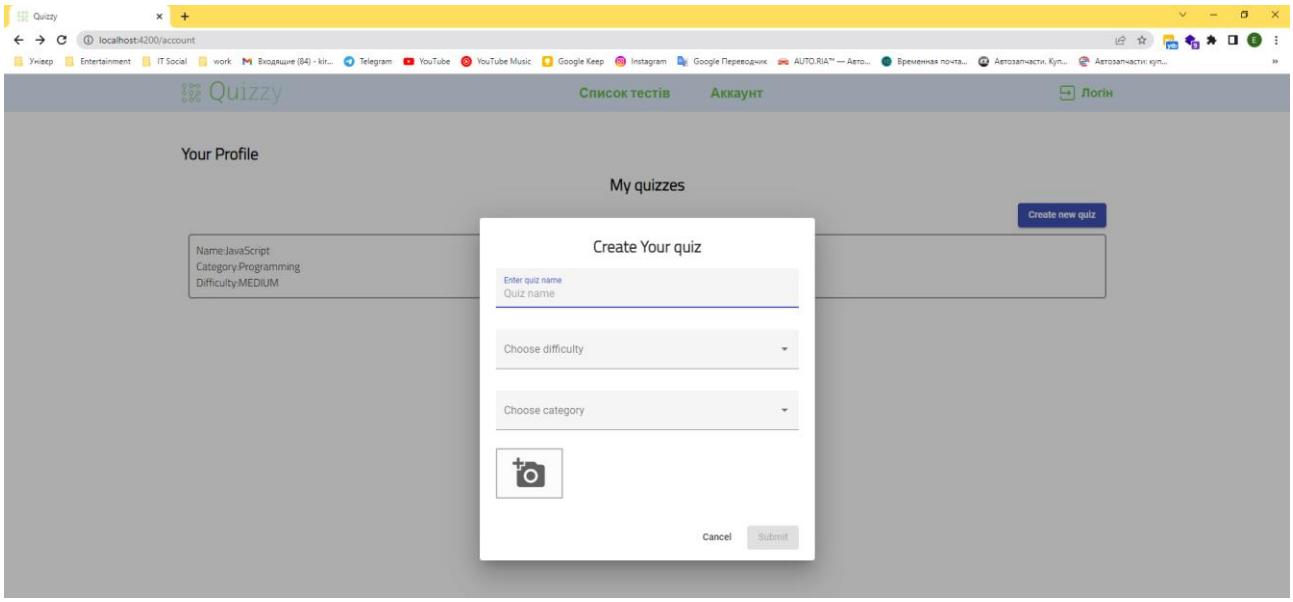


Рисунок 4.14 – Модальне вікно для створення тесту

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

4.3 Висновки

У даному розділі було 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.

ВИСНОВКИ

У бакалаврській дипломній роботі було розроблено програмний веб додаток, який призначений для проходження тестів з фото контролем.

Було 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.

Було проаналізовано мови програмування та було обрано TypeScript та фреймворк Angular.

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

Для веб додатку було 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.

У четвертому розділі було проведено тестування веб додатку в різних браузерах та основний функціонал по додаванню, редагуванню тесту працює добре та критичних помилок немає також було створено інструкцію для користувача.

Результати проекту оформлені відповідно до вимог [13].

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. 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.
2. 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
3. Робота в системі дистанційного навчання MOODLE. Інструкції для викладача – Тернопіль, ННІІОТ ТНЕУ, 2014. – 73 с
4. JetIQ. [Електронний ресурс] URL: https://jetiq.vntu.edu.ua/
5. Google Форми. [Електронний ресурс] URL: https://uk.wikipedia.org/wiki/google_форми
6. Online Test Pad. [Електронний ресурс] URL: https://onlinetestpad.com/
7. Вікіпедія водоспадна модель. [Електронний ресурс] URL: https://uk.wikipedia.org/wiki/водоспадна_модель
8. Вікіпедія спіральна модель. [Електронний ресурс] URL: https://uk.wikipedia.org/wiki/спіральна_модель
9. Вікіпедія UML. [Електронний ресурс] URL: https://uk.wikipedia.org/wiki/Unified_Modeling_Language
10. Структурна карта Константайна. [Електроний ресупс] URL: https://studref.com/311819/informatika/strukturnye_karty_konstantayna
11. Розробка інтерфейсу користувача. [Електроний ресурс] URL: https://uk.wikipedia.org/wiki/Інтерфейс_користувача
12. Capture Audio and Video in HTML5. [Електроний ресурс] URL: https://www.html5rocks.com/en/tutorials/getusermedia/intro/

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

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

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

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

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

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

19. Прохоренок Н. HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера / Н. Прохоренок. – СПб.: БХВ-Петербург, 2008. – 640 с

20. Книга «Angular для профессионалов», Адам Фримен, Питер, 2017 – 800 с

21. Лещев Д. Создание интерактивного web-сайта: учебный курс / Д. Ле щев. – СПб.: Питер, 2003. – 544 с.

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

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

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

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

ДОДАТКИ

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

"25" березня 2022 р.

Технічне завдання

на бакалаврську дипломну роботу «Розробка адаптивної тестувальної системи з фотоконтролем. Частина 2. Модуль клієнта з використанням технологій JavaScript/Typescript і фреймворку Angular» за спеціальністю 121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:

_____ к.т.н., доцент Д.І. Катєльніков

"__" __ 2022 р.

Виконав:

_____ студент гр. 2ПІ-18б А.В. Кирнасюк Є.С.

"__" __ 2022 р.

Вінниця – 2022 року

1. Найменування та галузь застосування

Бакалаврська дипломна 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.

2. Підстава для розробки.

Підставою для виконання бакалаврської дипломної роботи (БДР) є індивідуальне завдання на БДР та наказ №66 від «24» березня 2022 р. ректора по ВНТУ про закріплення тем БДР.

3. Мета та призначення розробки.

Метою бакалаврської дипломної роботи є розробка адаптивної тестувальної системи з фотоконтролем. Частина 2. Модуль клієнта з використанням технологій JavaScript/Typescript і фреймворку Angular.

Призначення роботи – розробка та програмна реалізація веб додатку для проходження тестів з фотоконтролем.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись БДР.

1. Лещев Д. Создание интерактивного web-сайта: учебный курс / Д. Ле щев. – СПб.: Питер, 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. 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 БДР;
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 бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз завдання і вибір методу вирішення поставленої задачі дослідження	26.03.2022 – 10.04.2022	Вик.
2	Розробка алгоритму створення тесту	11.04.2022 – 17.04.2022	Вик.
3	Розробка алгоритму проходження тестів з фотоконтролем	18.04.2022 – 21.04.2022	Вик.
4	Аналіз і вибір мови програмування та середовища розробки	22.04.2022 – 15.05.2022	Вик.
5	Програмна реалізація веб додатку	16.05.2022 – 23.05.2022	Вик.
6	Тестування розробленого веб додатку	24.05.2022 – 10.06.2022	Вик.
7	Оформлення матеріалів до захисту БДР	26.03.2022 – 10.04.2022	Вик.

10. Порядок контролю та прийняття

Виконання етапів бакалаврської дипломної роботи контролюється керівником згідно з графіком виконання роботи.

Прийняття бакалаврської дипломної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

ДОДАТОК Б

(обов’язковий)

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка адаптивної тестувальної системи з фотоконтролем. Частина 2. Модуль клієнта з використанням технологій JavaScript/Typescript і фреймворку Angular

Тип роботи: БДР

Підрозділ : кафедра програмного забезпечення, ФІТКІ

Науковий керівник: Катєльніков Д.І.

Оригінальність	
Схожість	

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.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи _____ Кирнасюк Є.С.

Керівник роботи _____ Катєльніков Д.І.

ДОДАТОК В

(довідниковий)

Лістинг програми

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: 'Hi', 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);
    }
    public handleImage(webcamImage: WebcamImage): void {
      console.info('received webcam image', webcamImage);
      this.webcamImage = webcamImage;
    }
    public cameraWasSwitched(deviceId: string): void {
      console.log('active device: ' + deviceId);
      this.deviceId = deviceId;
    }
    public get triggerObservable() {
      return this.trigger.asObservable();
    }
    public get nextWebcamObservable() {
      return this.nextWebcam.asObservable();
    }
}
```

ДОДАТОК Г

(обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА АДАПТИВНОЇ ТЕСТУВАЛЬНОЇ СИСТЕМИ З ФОТОКОНТРОЛЕМ. ЧАСТИНА 2. МОДУЛЬ КЛІЄНТА З ВИКОРИСТАННЯМ ТЕХНОЛОГІЙ JAVASCRIPT/TYPESCRIPT І ФРЕЙМВОРКУ ANGULAR

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

Бакалаврська дипломна робота

на тему: Розробка адаптивної тестувальної системи з фотоконтролем. Частина 2. Модуль клієнта з використанням технологій JavaScript/Typescript і фреймворку Angular

Виконав:
студент групи 2ПІ-18б
Кирнасюк Євген Сергійович

Науковий керівник:
к.т.н., доцент кафедри ПЗ
Катєльніков Денис Іванович

Рисунок Г.1 – Титульний слайд

Мета, об'єкт та предмет дослідження

Метою бакалаврської дипломної роботи є розробка адаптивної тестувальної системи з фотоконтролем. Частина 2. Модуль клієнта з використанням технологій JavaScript/Typescript і фреймворку Angular.

Об'єктом дослідження бакалаврської дипломної роботи є процес розробки адаптивної тестувальної системи з фотоконтролем, а саме модуля клієнта з використанням технологій JavaScript/Typescript і фреймворку Angular.

Предмет дослідження – методи та засоби розробки адаптивної тестувальної системи з фотоконтролем.

Рисунок Г.2 – Мета, об'єкт та предмет дослідження

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.3 – Мова програмування

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.4 – Фреймворк

Схема роботи веб додатку (SPA) напсаного на фрейворку Angular

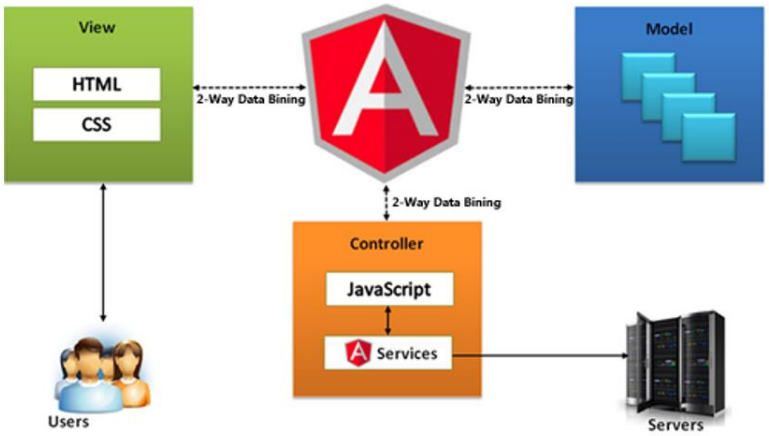
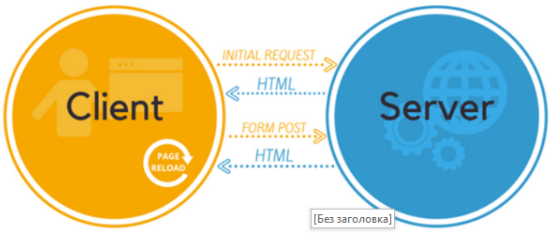


Рисунок Г.5 – Архітектура додатку

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;

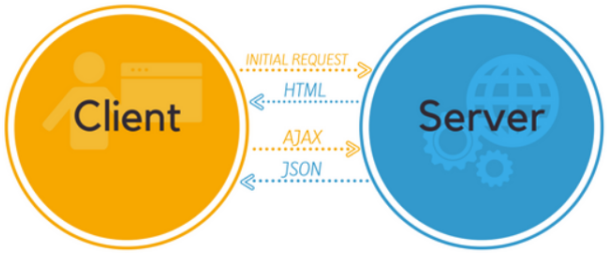
•Як правило, для розробки багатосторінкової програми потрібен менший стек технологій, таким чином вартість таких додатків виходить дешевшою.

Рисунок Г.6 – Визначення «веб сайт»

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.7 – Визначення «веб додаток»

Блок-схема алгоритму роботи тесту

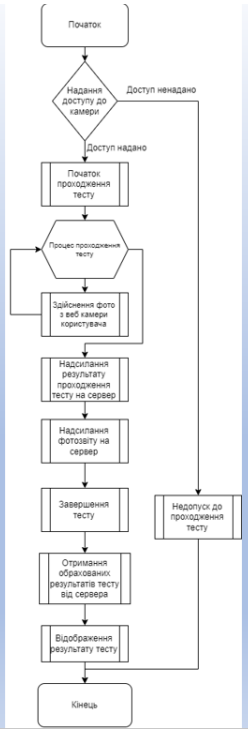


Рисунок Г.8– Алгоритм проходження тесту

Головна сторінка на якій зображено загальний список тестів

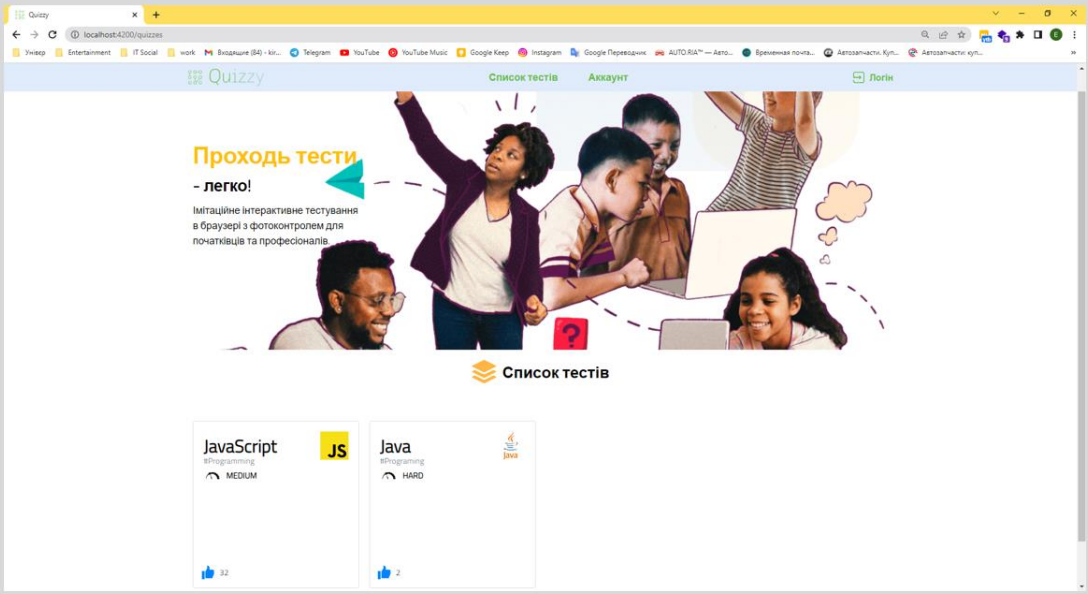
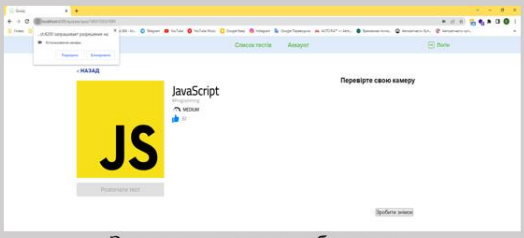
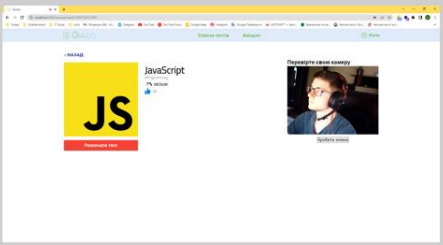


Рисунок Г.9 – Головна сторінка веб додатку

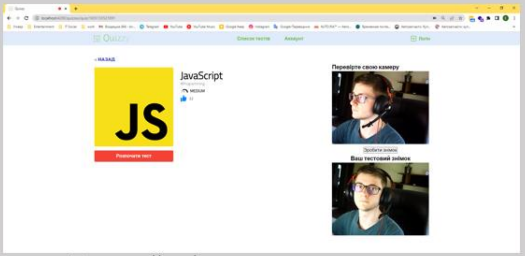
Сторінка вибраного тесту



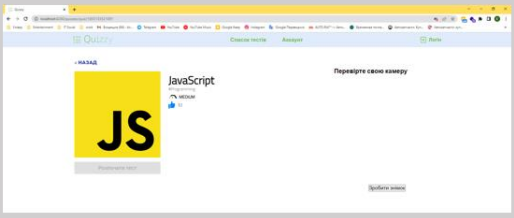
Запит доступу до веб-камери



Результат запиту на доступ - дозволено



Тестовий знімок перед початком тесту



Результат запиту на доступ – заблоковано
Не доступ до тесту

Рисунок Г.10 – Сторінка тесту та тестування камери

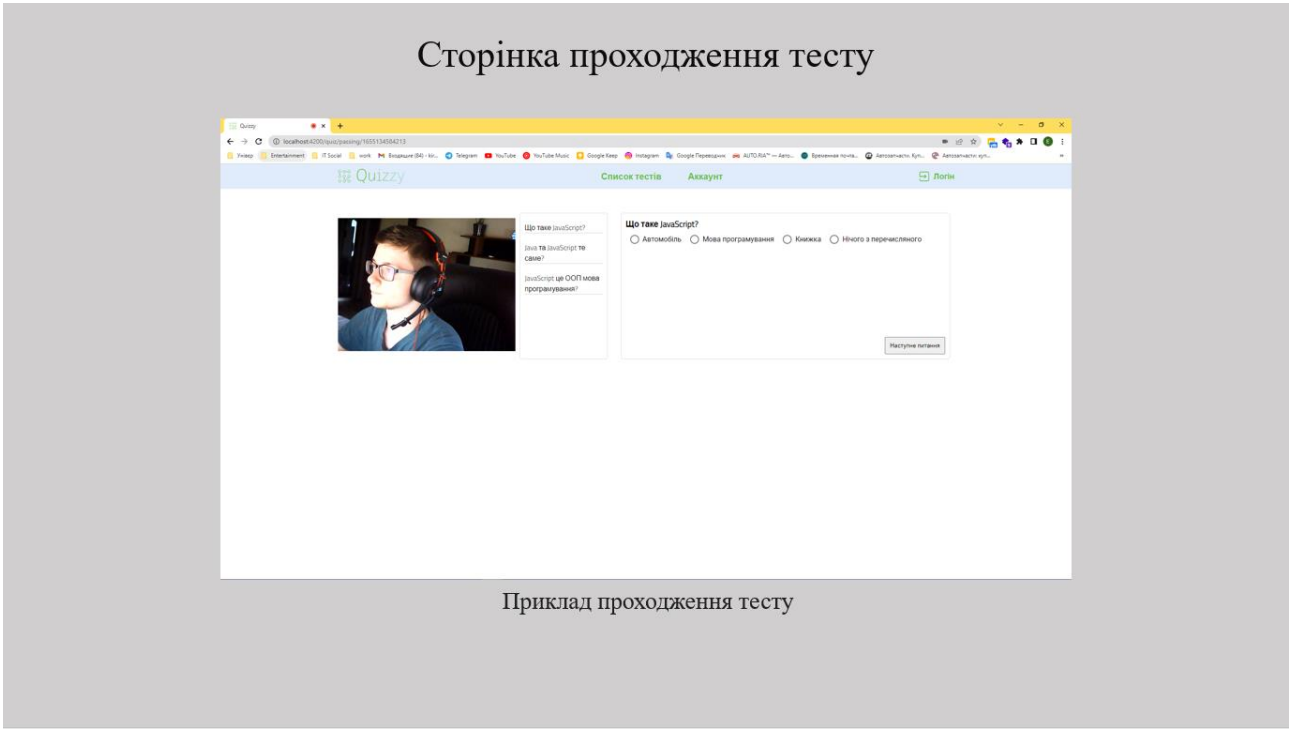


Рисунок Г.11 – Сторінка проходження тесту

Рисунок Г.12 – Фінальний слайд