МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ ДЕРЖАВНИЙ ПОЛІТЕХНІЧНИЙ КОЛЕДЖ

Циклова комісія інформаційних технологій

КУРСОВИЙ ПРОЕКТ
(РОБОТА)

Програмування
(назва дисципліни)

на тему: Розробка програмного забезпечення для тестування

Студентки IV курсу 47-кі групи

спеціальності 123 Комп'ютерна інженерія

спеціалізації 5.123.1 Обслуговування
комп'ютерних систем і мереж

Бурди М.О.

Керівник: викладач ***Васильєв М.В.***

Національна шкала _____
Кількість балів: __ Оцінка: ECTS _____

Члени комісії _____ (Васильєв М.В.)
(підпис)

_____ (_____)
(підпис)

_____ (_____)
(підпис)

м. Харків – 2023 рік

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ ДЕРЖАВНИЙ ПОЛІТЕХНІЧНИЙ КОЛЕДЖ

РОЗГЛЯНУТО
на засіданні циклової комісії
Голова циклової комісії
Марина ВЕЛИЧКО
« 4 » січня 2023 р.

ЗАТВЕРДЖЕНО
Зав. відділення
Ірина КРЮКОВА
« 5 » січня 2023 р.

Завдання

на курсову роботу студенту гр. 47-кі
спеціальність/освітньо-професійна програма 123 Комп'ютерна інженерія /5.123.1 Обслуговування комп'ютерних систем і мереж

Бурді Марині Олегівні

Тема: ***Розробка програмного забезпечення для тестування.***
Термін виконання роботи з 18.01.2023 р. по 7.04.2023 р.
Вхідні дані для проектування: (Варіант №4)

Розробити програму тестування знань по темі «Мова програмування Pascal». Програма повинна містити щонайменше 10 запитань, на кожне – кілька варіантів відповідей, з яких лише одна правильна. В кінці повинна виводитися кількість правильних відповідей та оцінка тестування.

Перед початком тестування передбачити можливість реєстрації, всі питання тестів зберігаються в окремому файлі, виведення результатів тестування проводиться за допомогою простого вікна повідомлень або нової форми.

Література та посібники для проектування:

1. Head First JavaScript Programming: A Brain-Friendly Guide: Eric Freeman, Elisabeth Robson. – Sebastopol, California: O′Reilly, 2014. – 700 с.
2. Hands-On RESTful API Design Patterns and Best Practices: Design, develop, and deploy highly adaptable, scalable, and secure RESTful web APIs: Harihara Subramanian, Pethuru Raj. – Birmingham, United Kingdom: Packt, 2019. – 378 с.
3. Getting to Know Vue.js: Learn to Build Single Page Applications in Vue from Scratch: Brett Nelson. - New York City, USA: Apress, 2018. – 278 с..
4. Clean Code: A Handbook of Agile Software Craftsmanship: Robert C. Martin. - London, England: Pearson, 2008. – 464 с.
5. MDN Web Docs: веб-сайт. URL: https://developer.mozilla.org (дата звернення: 25.8.2022)
6. W3Schools Online Web Tutorials: веб-сайт. URL: htpps://www.w3schools.com (дата звернення 25.8.2022)
7. Refactoring and Design Patterns: веб-сайт. URL: https://refactoring.guru (дата звернення: 25.8.2022)

Зміст розрахунково-пояснювальної записки

№ п/п	Зміст	Планований термін виконання	Фактичний термін виконання	%
1	Вступ	27.01.2023	30.01.2023	10
2	Аналіз задачі, засобів та методів її вирішення	1.02.2023	3.02.2023	25
3	Проектування загального алгоритму роботи програми	15.02.2023	13.02.2023	40
4	Розробка функціональних алгоритмів роботи програми	1.03.2023	17.02.2023	65
5	Розробка програмного забезпечення	15.03.2023	13.03.2023	80
6	Керівництво користувача	29.03.2023	24.03.2023	85
7	Висновки	31.03.2023	31.03.2023	90
8	Список використаних джерел	5.03.2023	5.03.2023	100

Керівник курсової роботи _____ Микола ВАСИЛЬЄВ

Завдання до курсової роботи

Одержала студентка гр. 47-КІ _____ (_____)

Дата «18» січня 2023 р.

Реферат

Пояснювальна записка: 35 сторінка, 26 рисунків, 5 використаної літератури.

Ключові слова: ПРОГРАМУВАННЯ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, JAVA SCRIPT, LOCAL STORAGE, ТЕСТУВАННЯ ЗНАНЬ.

Об’єктом розробки є програмне забезпечення для тестування.

Мета розробки – розробка програмного забезпечення.

У процесі роботи проведено розробку блок-схеми алгоритму роботи програми та окремих функцій, розроблено програму, яка виконує алгоритм роботи.

Основні конструктивні та техніко-економічні показники: висока надійність, зручність, кросбраузерність, мультиплатформеність.

Перелік використаних скорочень

- МП – мова програмування
- ПЗ – програмне забезпечення
- CSS – Cascading Style Sheets
- JS – JavaScript
- VS Code – Visual Studio Code

Зміст

ВСТУП

Найважливішим аспектом будь-якої освітньої діяльності є система контролю якості знань. Активне використання навчальними закладами засобів інформатизації забезпечило передумови до створення й використання автоматизованих тестів для контролю знань учнів (студентів) на всіх етапах навчання. Такі системи використовуються не тільки для визначення рівня підготовленості, але й для проведення моніторингу навчального процесу, для організації адаптивного навчання, дистанційного утворення. Актуальність тестового методу обумовлена його перевагами перед іншими педагогічними методами: наукова обґрунтованість тесту, що дає об’єктивну оцінку; технологічність тестових методів; точність визначень; наявність однакових вимог для всіх випробуваних; сумісність тестових технологій з іншими сучасними освітніми технологіями

Мета курсової роботи полягає в тому, щоб розробити програмне забезпечення для тестування знань по темі «Мова програмування Pascal».

Для виконання даного проекту необхідно розробити алгоритм рішення поставленого завдання, правильно вказавши послідовне виконання відповідних команд для отримання необхідних результатів.

Програмне забезпечення було розроблене в середовищі Visual Studio Code. Visual Studio Code – засіб для створення, редагування та налагодження сучасних веб-застосунків і програм для хмарних систем. Visual Studio Code розповсюджується безкоштовно і доступний у версіях для платформ Windows, Linux і OS X.

Компанія Microsoft представила Visual Studio Code у квітні 2015 на конференції Build 2015. Це середовище розробки стало першим кросплатформним продуктом у лінійці Visual Studio.

JavaScript (JS) — динамічна, об'єктно-орієнтована прототипна МП. Реалізація стандарту ECMAScript. Найчастіше використовується для створення сценаріїв вебсторінок, що надає можливість на боці клієнта

(пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки.

JavaScript класифікують як прототипну (підмножина об'єктно-орієнтованої), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу.

ОСНОВНА ЧАСТИНА

Аналіз задачі, засобів та методів її вирішення

Згідно з завданням необхідно розробити програму тестування, яка буде перевіряти знання студентів з мови програмування Pascal. Роботу виконано на МП JavaScript.

Також потрібно, щоб результати тестування десь зберігалися. За допомогою Local Storage можна зберігати дані локально.

JavaScript - це крос-платформна, об'єктно-орієнтована скриптова мова, яка є невеликою і легковажною. Всередині середовища виконання JavaScript може бути пов'язаний з об'єктами даного середовища та надавати програмний контроль над ними.

JavaScript включає стандартну бібліотеку об'єктів, наприклад, Array, Datета Math, а також базовий набір мовних елементів, наприклад, оператори та керуючі конструкції.

JavaScript на стороні клієнта розширює ядро мови, надаючи об'єкти контролю браузера та його Document Object Model (DOM). Наприклад, розширення клієнтів дозволяють застосуванню розміщувати елементи у формі HTML і обробляти події користувача, такі як клацання миші, введення даних у форму і навігація по сторінках.

JavaScript на стороні сервера розширює ядро мови, надаючи об'єкти для запуску JavaScript на сервері. Наприклад, розширення на стороні сервера дозволяє з'єднуватися з базою даних, забезпечувати безперервність інформації між викликами програми або виконувати маніпуляції над файлами на сервері.

Local Storage - це властивість, що відкриває доступ до спеціального об'єкта Storage (сховище). Його використовують із отримання інформації з локального сховища. Дані зберігаються там необмежений період і можуть бути видалені лише за допомогою JavaScript. Ці дані характерні для

протоколу веб-сторінки: це ключі та значення у форматі рядків (навіть цілі значення ключів будуть перетворюватися на рядки). Простіше кажучи, Local Storage це об'єкт у браузері, яким ми можемо скористатися. Мовою JS він є властивості глобального об'єкта браузера (тобто — window, вікна).

CSS (Cascading Style Sheets) – формальна мова опису зовнішнього вигляду документа (веб-сторінки), написаного з використанням мови розмітки (найчастіше HTML або XHTML).

CSS використовується творцями веб-сторінок для завдання кольорів, шрифтів, стилів, розташування окремих блоків та інших аспектів представлення цих веб-сторінок. Основною метою розробки CSS є огородження та відокремлення опису логічної структури веб-сторінки (яке виробляється за допомогою HTML або інших мов розмітки) від опису зовнішнього вигляду цієї веб-сторінки (яке тепер виробляється за допомогою формальної мови CSS). Такий поділ може збільшити доступність документа, надати більшу гнучкість та можливість управління його поданням, а також зменшити складність та повторюваність у структурному вмісті.

Проектування загального алгоритму роботи програми

Для зручності доцільно розділити роботу на два етапи. Перший – це створення головної сторінки програми, а другий – саме тестування.

На першому етапі доцільно виконати розробку інтерфейсу таким чином, щоб було зрозуміло, що це ПЗ для тестування знань, яке буде перевіряти знання.

При виконанні другого етапу слід наповнити вміст програми запитаннями та варіантами відповідей. Зробити зрозумілим, коли користувач відповів правильно, а коли – ні. Повинна аналізуватися правильність та коректність відображення інформації у користувача. На цьому етапі потрібно перевірити відповідність розробленої програми із завданням, а також перевірити її на відсутність помилок.

Програма буде складатися з класу Quiz, що буде головним класом програми, до якого ми будемо передавати масив з питаннями, відповідями і з результатами оцінювання. Ці масиви будуть складатися з створення нових об’єктів класів результату, питання і відповідей.

Клас Question буде приймати запитання.

Клас Answer буде приймати варіанти відповідей, а також індекс, де 0 – неправильна відповідь, а 1 – правильна.

Клас Result буде приймати оцінювання і відповідний результат.

Підсумувавши все вищесказане, можна сформулювати вимоги до розробки ПЗ і виконати постановку завдання на проектування.

Наступні розділи будуть присвячені вирішенню поставлених завдань і розробці програми з вищепереліченими функціями.

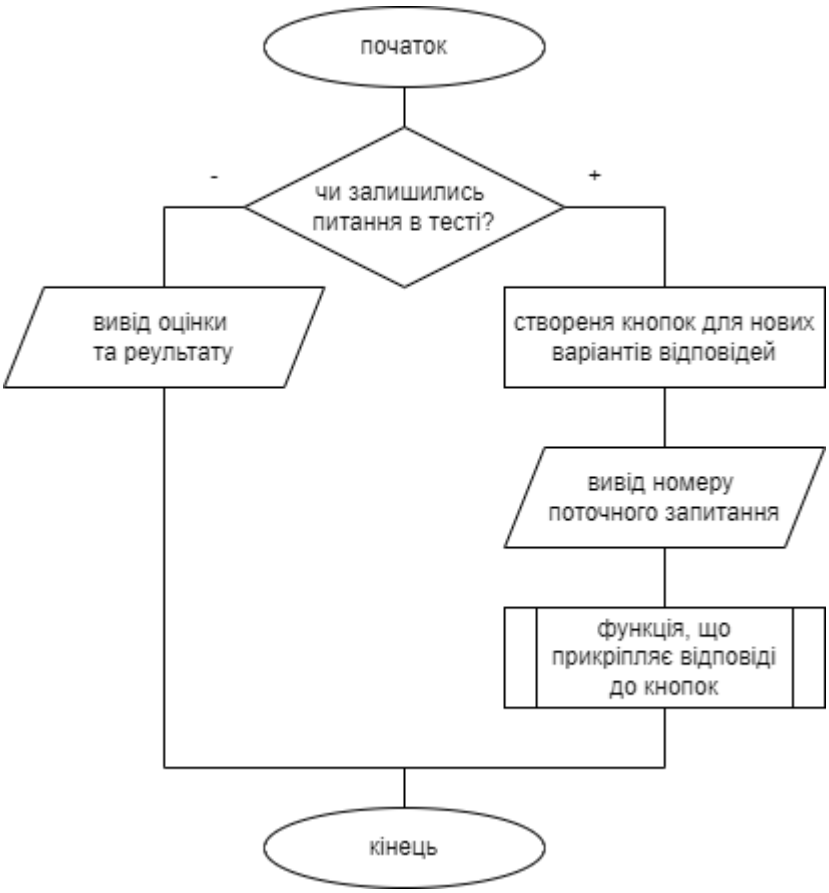

Рис. 2.1 – Схема основної функції Update()

Розробка функціональних алгоритмів роботи програми

На головній сторінці буде розташований заголовок тестування, запит ім’я та прізвища та кнопка для підтвердження. На сторінці тестування буде виводитися питання та варіанти відповідей. Вкінці виводитиметься результат та бал тестування.

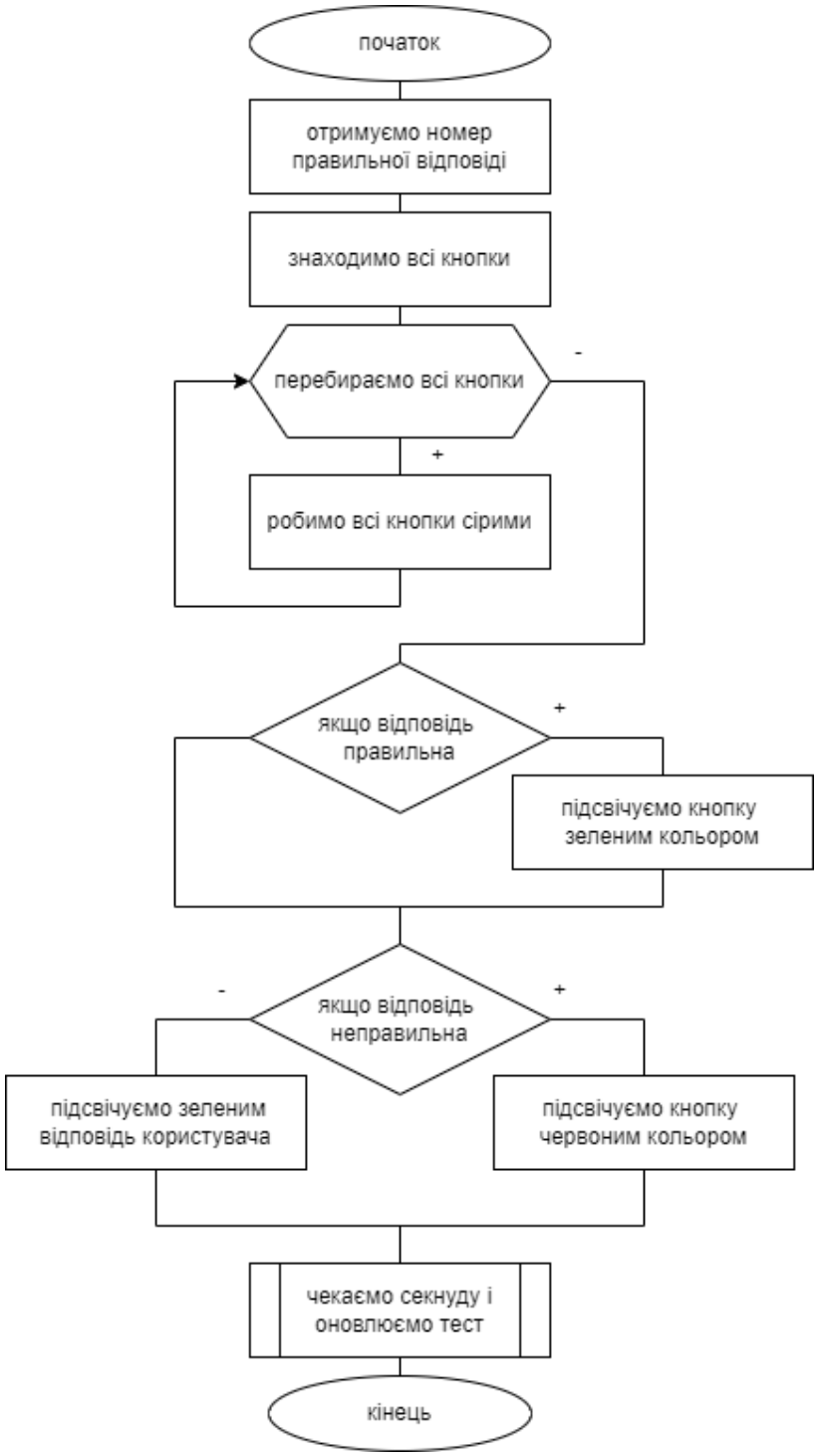

Рис. 3.1 – Схема функції Click()

Рис. 3.2 – Схема функції Init()

Розробка програмного забезпечення

Для розробки ПЗ для тестування знань потрібно виконати наступні кроки:

1. Визначити вимоги до програми – які функції має виконувати програма, які дані обробляти, які формати даних маж підтримувати, які мови програмування має підтримувати.
2. Розробити архітектуру програми – визначити, які компоненти програми потрібні, як повинні співпрацювати, які мови програмування використовувати.
3. Реалізувати компоненти програми – написати код компонентів програми відповідно до архітектури.
4. Провести тестування програми – перевірити, що програма працює правильно.
5. За необхідності, внести виправлення в програму та перевірити правильність їх роботи.

Перший і другий крок описаний в попередніх розділах. Приступимо до написання коду.

Спочатку було створено файл index.html. Це веб-сторінка з формою, яка запитує ім’я та прізвище користувача. Коли користувач натискає на кнопку, запускається функція func() в файлі save-name.js, що зберігає ім’я та прізвище користувача. Тож ще був створений скрипт і функція в ньому.

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Головна</title>
    <link rel="stylesheet" href="./style.css" />
    <link
      rel="stylesheet"
      href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.10.3/font/bootstrap-icons.css"
    />
  </head>
  <body>
    <form action="app.html">
      <div class="container">
        <div class="flex-column flex-center" id="home">
          <h1>
            <i class="Puzzle bi bi-puzzle-fill"></i>Тестування знань по темі
            "Мова програмування Pascal"
          </h1>
          <h3>Перед початком тестування введіть ваші дані:</h3>
          <input id="name" type="text" name="name" placeholder="Ім'я" />
          <input id="surname" type="text" name="surname" placeholder="Прізвище" />
          <button class="btn" onclick="func()">Розпочати</button>
        </div>
      </div>
    </form>
    <script src="save-name.js"></script>
  </body>
</html>
```

Рис. 4.1 – Код сторінки index.html

```javascript
const func = () => {
    const userName = document.getElementById("name").value;
    const userSurname = document.getElementById("surname").value;
    const user = userName + " " + userSurname;
    localStorage.setItem("user", user);
}
```

Рис. 4.2 – Код скрипту save-name.js

Також було прописано, що при натисканні кнопки, відбувається перенаправлення на іншу сторінку. Документ також містить деякі стилі з файлу style.css та іконку з бібліотеки Bootstrap.

```css
.flex-center {
    justify-content: center;
    align-items: center;
}

input[type=text]{
    width: 30%;
    outline: none;
    height: 40px;
    margin-bottom: 10px;
    padding-left: 15px;
    background: rgba(255, 255, 255, 0.699);
    border: none;
    color: #464545;
    font-size: 15pt;
}

.btn {
    font-size: 20pt;
    padding: 10px 25px;
    margin-bottom: 3px;
    margin-top: 20px;
    text-align: center;
    color: rgba(0, 0, 0, 0.61);
    background: rgba(89, 60, 221, 0.575);
    text-decoration: none;
    border-radius: 10px;
    border: 0
}

.btn:hover {
    cursor: pointer;
    box-shadow: 0 4px 2px 0  rgba(89, 60, 221, 0.911);
    transition: transform 150ms;
    transform: scale(1.01);
}

.Puzzle {
    padding-right: 10px;
    opacity: 0.5;
}
```

Рис. 4.3 – Деякі стилі з файлу style.css

Потім було створено сторінку з самим тестуванням. Вона містить необхідні HTML-теги для створення сторінки, підключення CSS-файлу, а також посилання на два JS-файли (app.js і q-a.js). Сторінка містить заголовок,

чотири кнопки з різними класами та нижній колонтитул. Класи використовуються для різного стилю кнопок, а файли JS – для додавання функціонування до тесту. Коли скрипт не буде підключено чи виникне помилка, сторінка буде виглядати так:

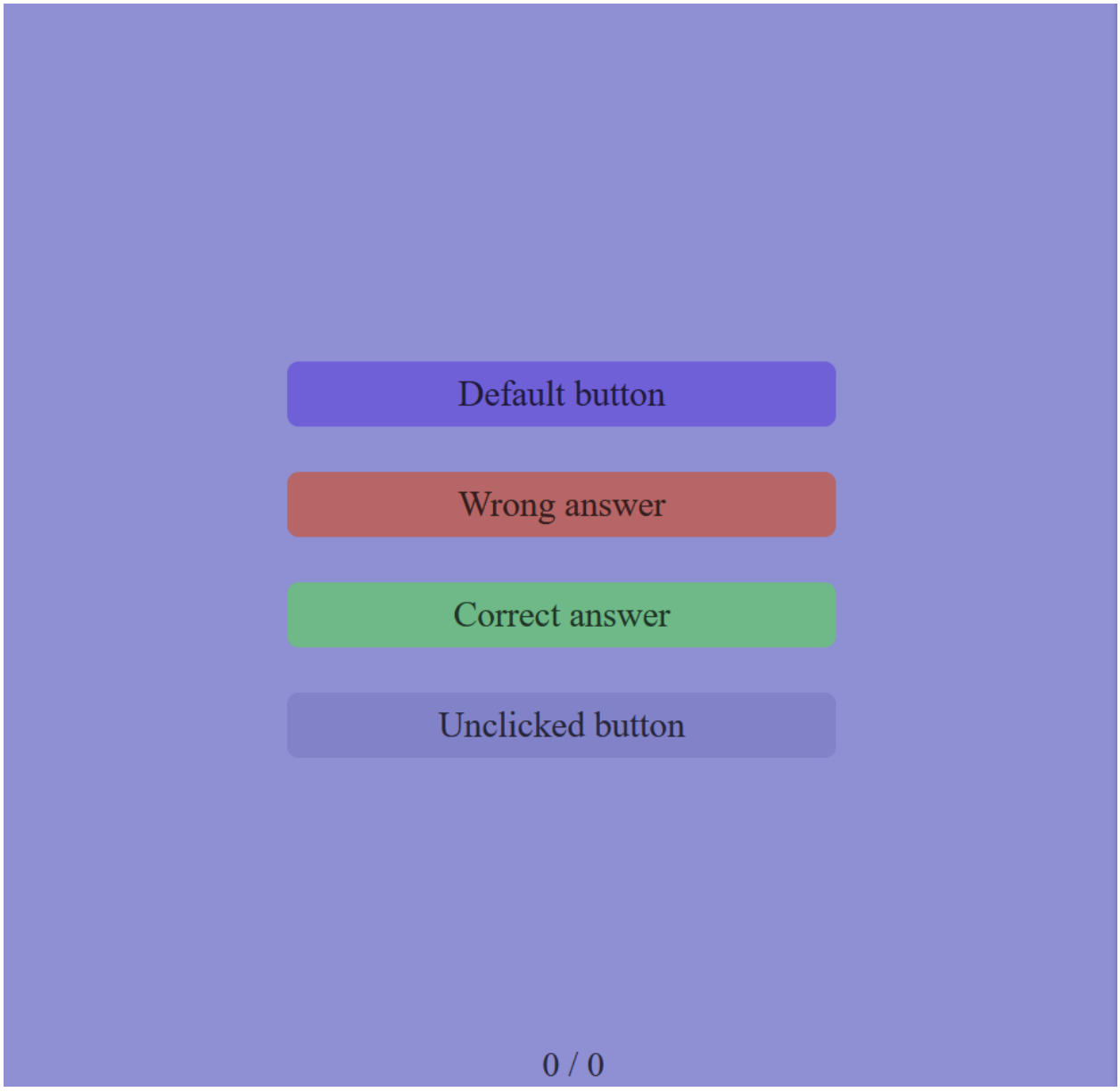

Рис. 4.4 – Вигляд сторінки без підключення скрипту

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Тест</title>
    <link rel="stylesheet" type="text/css" href="app.css" />
  </head>
  <body>
    <div class="wrapper">
      <main class="main">
        <div class="quiz__head">
          <div class="head__content" id="head"></div>
        </div>
        <div class="quiz__body">
          <div class="buttons">
            <div class="buttons__content" id="buttons">
              <button class="button">Default button</button><br />
              <button class="button button_wrong">Wrong answer</button><br />
              <button class="button button_correct">Correct answer</button><br />
              <button class="button button_passive">Unclicked button</button><br />
            </div>
          </div>

          <div class="quiz__footer">
            <div class="footer__content" id="pages">0 / 0</div>
          </div>
        </div>
      </main>
    </div>
    <script src="app.js"></script>
    <script src="q-a.js"></script>
  </body>
</html>
```

Рис. 4.5 – Код сторінки app.html

```css
.quiz__head {

    margin: 10px;
    margin-bottom: 50px;
}

.head__content  {
    padding: 5px;
    font-size: 35pt;
}

.quiz__body {
    margin: 10px;
}

.quiz__footer  {
    position: absolute;
    bottom: 0;
    display: block;
    width: 100%;
}

.footer__content  {
    padding: 5px;
}

.button {
    font-family: 'Nunito-regular';
    border-radius: 10px;
    background: rgba(89, 60, 221, 0.575);
    color: rgba(0, 0, 0, 0.719);
    padding: 10px 25px;
    width: 50%;
    font-size: 23pt;
    display: block;
    margin: 2px auto;
    border: 0;
}

.button:hover {
    cursor: pointer;
    box-shadow: 0 4px 2px 0 rgba(97, 72, 206, 0.644);
    transition: transform 150ms;
    transform: scale(1.01);
}
```

Рис. 4.6 – Деякі стилі з файлу app.css

Перейдемо до створення скриту app.js. Для того щоб взаємодіяти з

елементами сторінки і прописати логіку програми, було прописано:

```
const headElem = document.getElementById("head");
const buttonsElem = document.getElementById("buttons");
const pagesElem = document.getElementById("pages");
```

Рис. 4.7 – Отримання елементів по їх ідентифікаторам

Далі прописали класи. Клас Quiz містить конструктор, який створює вікторину з питаннями, результатами, балами, результатом і поточним станом. Він також містить методи для кліку на відповіді (Click()), переходу до наступного питання (Next()) і завершення тесту (End()). Клас Question містить конструктор, який задає текст питання і відповіді. Клас Answer містить конструктор, який задає текст відповіді та її значення. Клас Result містить конструктор, який задає текст результату та його значення. Він також має метод Check() для перевірки того, чи достатньо балів заробив користувач.

```
class Quiz {
    constructor(questions, results) {
        this.questions = questions; //Масив із питаннями
        this.results = results; //Масив з можливими результатами
        this.score = 0; //Кількість набраних балів
        this.result = 0; //Номер відповіді з масиву
        this.current = 0; //Номер поточного питання
    }

    Click(index) { //Додаємо бали
        let value = this.questions[this.current].Click(index);
        this.score += value;
        let correct = -1;

        //Якщо додано 1 бал, то відповідь правильна
        if (value >= 1) {
            correct = index;
        }
        else {
            //Інакше шукаємо правильну відповідь\
            for (let i = 0; i < this.questions[this.current].answers.length; i++) {
                if(this.questions[this.current].answers[i].value >= 1) {
                    correct = i;
                    break;
                }
            }
        }

        this.Next();
        return correct;
    }

    Next() { //Перехід до наступного питання
        this.current++;
        if (this.current >= this.questions.length) {
            this.End();
        }
    }

    End() { //Вивід результату в кінці програми
        for (let i = 0; i < this.results.length; i++) {
            if(this.results[i].Check(this.score)) {
                this.result = i;
            }
        }
    }
}
```

Рис. 4.8 – Клас Quiz

```
class Question {
    constructor(text, answers) {
        this.text = text;
        this.answers = answers;
    }

    Click(index) {
        return this.answers[index].value;
    }
}


class Answer {
    constructor(text, value) {
        this.text = text;
        this.value = value;
    }
}


class Result {
    constructor(text, value){
        this.text = text;
        this.value = value;
    }

    Check(value) { //Перевірка, чи достатньо балів отримав користувач
        if (this.value <= value) {
            return true;
        }
        else {
            return false;
        }
    }
}
```

Рис. 4.9 – Класи Question, Answer, Result

Потім було створено функцію, що оновлює питання та варіанти відповідей для тесту. Якщо тест завершився, то результати зберігаються у localStorage.

```
function Update() {
    //Перевірка, чи є ще питання
    if (quiz.current < quiz.questions.length) {
        headElem.innerHTML = quiz.questions[quiz.current].text; //Якщо є, змінюємо саме питання
        buttonsElem.innerHTML = ""; //Видаляємо старі варіанти відпоідей

        //Створення кнопок для нових варіантів відповідей
        for (let i = 0; i < quiz.questions[quiz.current].answers.length; i++) {
            let btn = document.createElement("button");
            btn.className = "button";
            btn.innerHTML = quiz.questions[quiz.current].answers[i].text;
            btn.setAttribute("index", i);
            buttonsElem.appendChild(btn);
        }

        pagesElem.innerHTML = (quiz.current + 1) + " / " + quiz.questions.length; //Вивід номеру поточного питання
        Init(); //Виклик функції, що прикріпляє відповіді до кнопок
    }
    else {
        //Якщо тест завершився, вивід результату
        buttonsElem.innerHTML = "";
        headElem.innerHTML = `Ваш результат: ` + quiz.results[quiz.result].text + ` бали/балів
        <br><button class="btn-end" onclick="location.reload()">Розпочати знову</button>
        <br><button class="btn-end" onclick="main()">На головну</button>`;

        pagesElem.innerHTML = "Кількість правильних відповідей: " + quiz.score;

        localStorage.setItem("grade", quiz.results[quiz.result].text);
        localStorage.setItem("result", quiz.score);
    }
}
```

Рис. 4.10 – Функція Update()

Було створено функцію, яка знаходить усі кнопки на сторінці з іменем класу button. Потім вона додає подію до кожної кнопки, щоб при натисканні на неї викликати функцію Click() і передати індекс кнопки як аргумент.

```
function Init() {
    let btns = document.getElementsByClassName("button"); //Знаходимо всі кнопки

    for (let i = 0; i < btns.length; i++) {
        btns[i].addEventListener("click", function (e) { Click(e.target.getAttribute("index")); });
    }
}
```

Рис. 4.11 – Функція Init()

І остання з найважливіших функцій Click(). Ця функція використовується для перевірки. Вона приймає індекс відповіді, яку обрав користувач, та перевіряє, чи це правильна відповідь.

```
function Click(index)  {
    let correct = quiz.Click(index); //Отримуємо номер правильної відповіді
    let btns = document.getElementsByClassName("button"); //Знаходимо всі кнопки

    for (let i = 0; i < btns.length; i++) { //Робимо кнопки сірими
        btns[i].className = "button button_passive";
    }

    //Підсвічуємо правильну відповідь зеленим, а неправильну – червоним
    if (correct >= 0) {
            btns[correct].className = "button button_correct";
    }

    if (index != correct) {
            btns[index].className = "button button_wrong";
    }

    else { //Інакше просто підсвічуємо зеленим відповідь користувача
        btns[index].className = "button button_correct";
    }

    setTimeout(Update, 1000); //Чекаємо секунду і оновлюємо тест
}
```

Рис. 4.12 – Функція Click()

Нарешті переходимо до скрипту q-a.js. В ньому створюються масив з критеріями оцінювання *results* (що є анонімними об’єктами класу Result) та масив з питаннями і відповідями *questions* (що є анонімними об’єктами класів Question і Answer). А також створюється новий об’єкт quiz з параметрами questions та results. Викликається функція Update().

```
const results = [
    new Result("Повторіть тему! 0", 0),
    new Result("1", 2),
    new Result("2", 4),
    new Result("3", 6),
    new Result("4", 8),
    new Result("5", 10)
];
```

Рис. 4.13 – Масив results

```
const questions = [
    new Question("Що таке ідентифікатор?",
    [
        new Answer("Змінна", 0),
        new Answer("Програма, яка реалізує процес обробки даних, оперує з різними типами даних, на які необхідно якимось чином посилатися.", 0),
        new Answer("Це будь-яка кінцева послідовність літер, цифр й символу підкреслення, яка починається з букви.", 1),
        new Answer("Опис дій над даними", 0)
    ]),

    new Question("Чи розрізняються великі і малі літери в ідентифікаторах при роботі програми?",
    [
        new Answer("Так", 0),
        new Answer("Ні", 1)
    ]),

    new Question("Розділ операторів починається службовим словом ...",
    [
        new Answer("type", 0),
        new Answer("begin", 1),
        new Answer("label", 0),
        new Answer("end", 0)
    ]),

    new Question("Як виглядає оператор присвоєння?",
    [
        new Answer(":=", 1),
        new Answer("=", 0),
        new Answer("==", 0),
        new Answer("!=", 0)
    ]),

    new Question("Заголовок програми починається зі слова ...",
    [
        new Answer("heading", 0),
        new Answer("program", 1),
        new Answer("title", 0)
    ]),
```

Рис. 4.14 – Частина масиву questions

```
const quiz = new Quiz(questions, results);
Update();
```

Рис. 4.15 – Запуск всього коду

Весь код app.html наведений в додатку А, app.js – в додатку Б.

Керівництво користувача

Створена програма призначена для оцінювання знань студентів по мові програмування Pascal. Також в будь-який момент можна замінити тему, питання, варіанти відповідей та критерії оцінювання.

Щоб запустити програму, достатньо відкрити файл index.html.

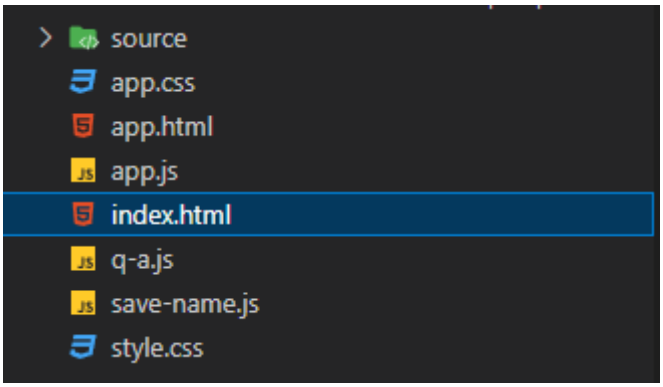

Рис. 5.1 – Структура проекту

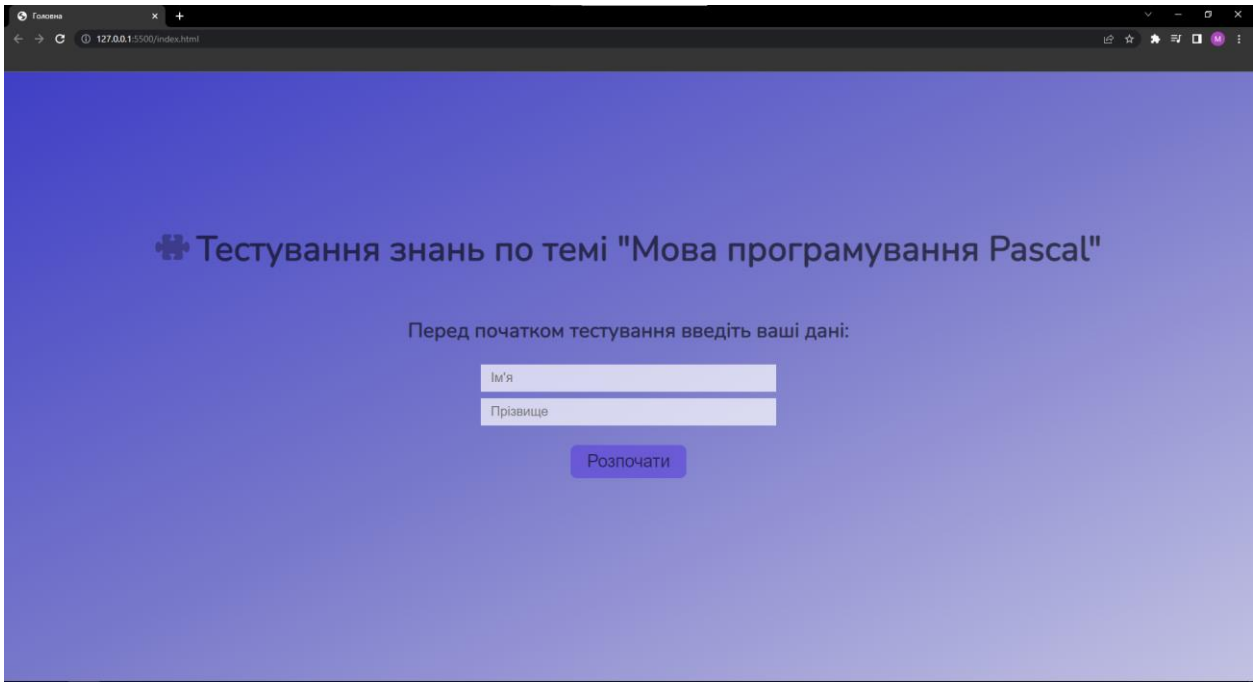

Рис. 5.2 – Головна сторінка програми

На цьому кроці потрібно ввести своє ім’я та прізвище. І натиснути на кнопку розпочати.

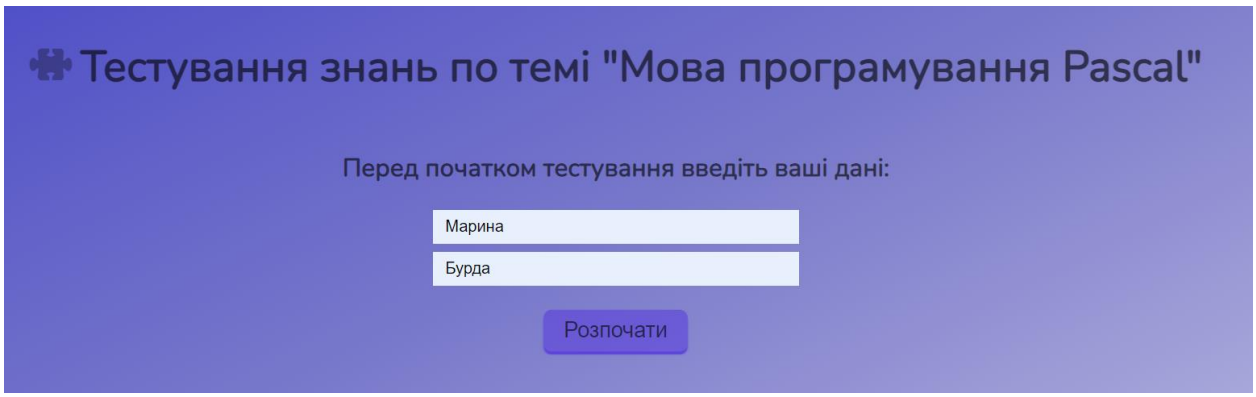

Рис. 5.3 – Приклад заповнення форми

Потім відкривається сторінка тест. І можна побачити питання та варіанти відповідей.

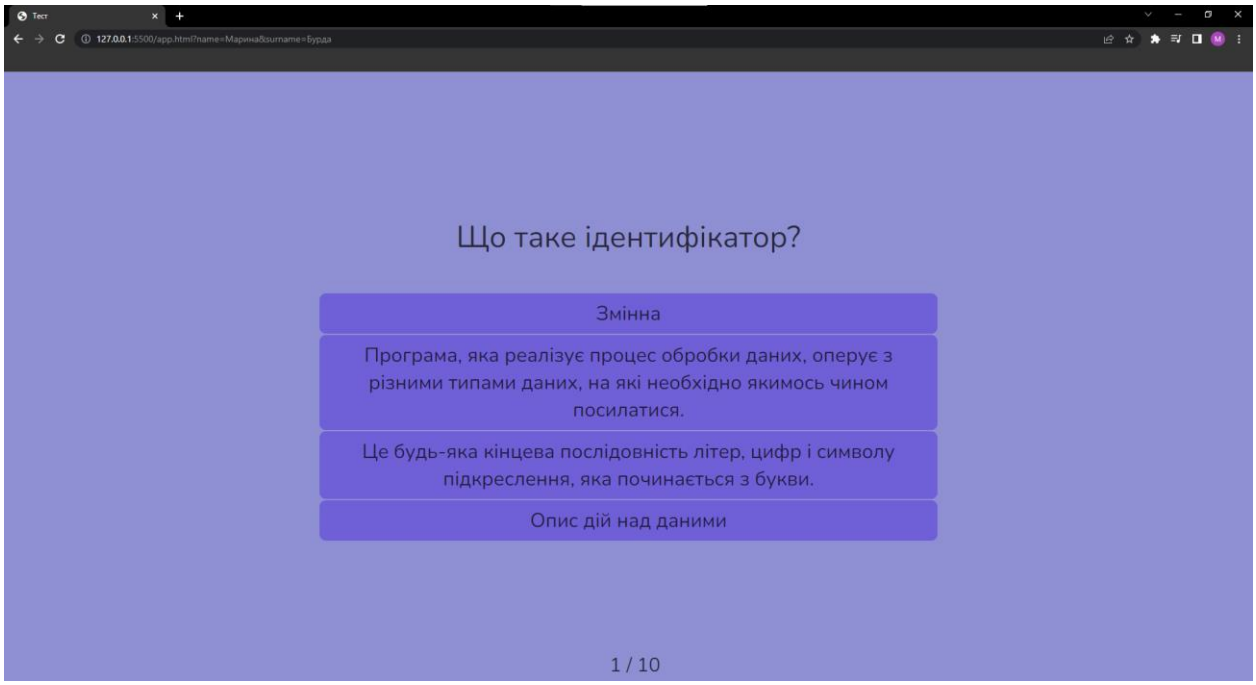

Рис. 5.4 – Сторінка тестування

При натисканні на будь-яку кнопку, підсвічуються правильні та хибні варіанти відповідей.

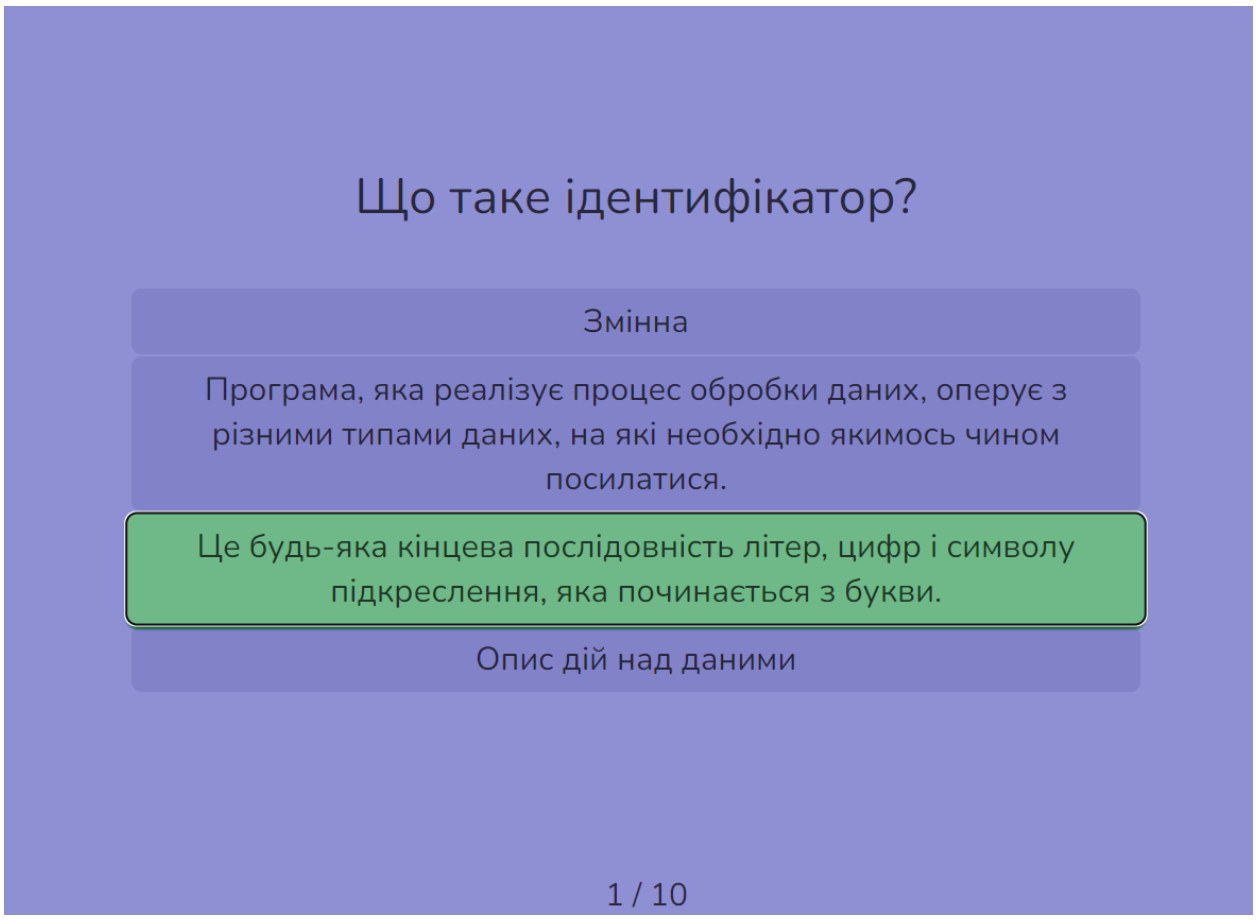

Рис. 5.5 – Вибір правильної відповіді

Після кожної відповіді з’являються нові запитання та варіанти відповідей аж поки не закінчаться питання.

Рис. 5.6 – Вибір неправильної відповіді (також підсвічується правильний варіант відповіді)

Вкінці тестування виводиться результат тестування, кількість правильних відповідей та кнопки, щоб перейти на головну сторінку або заново пройти тест.

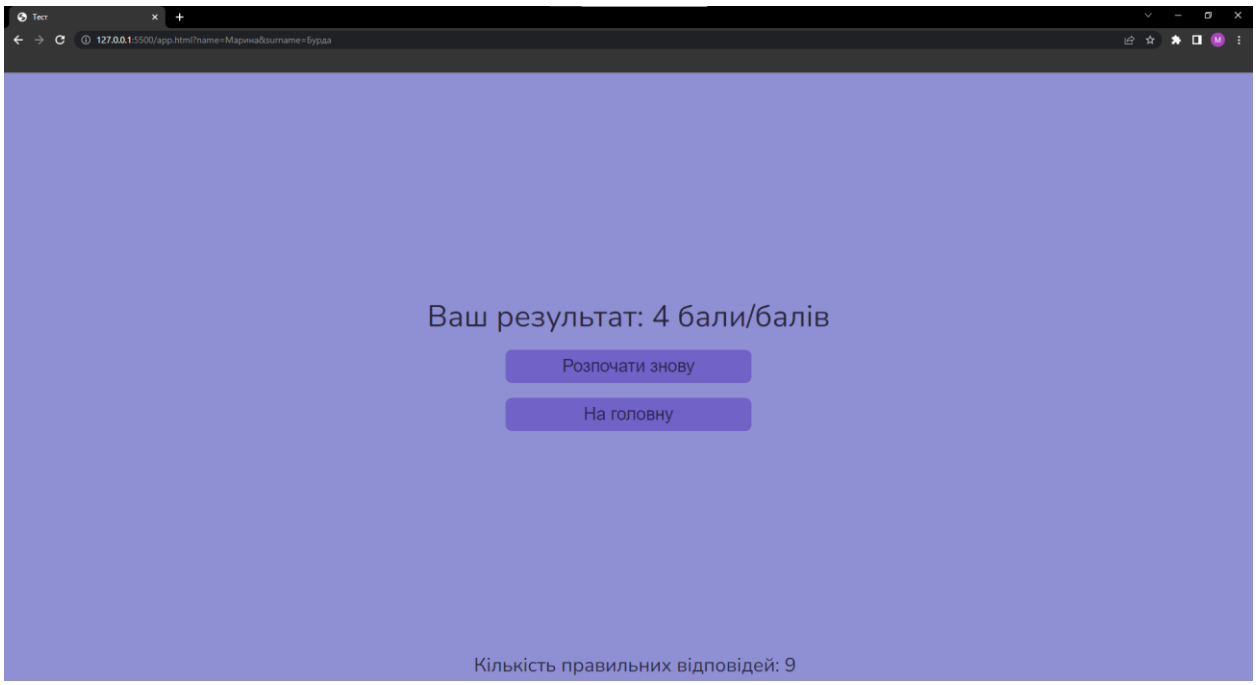

Рис. 5.7 – Результат тестування

При розробці ПЗ було прописано, що ім’я, прізвище, результат та кількість правильних відповідей зберігаються в Local Storage. Тепер можна це перевірити. Це може бут корисним, якщо користувач випадково вийшов.

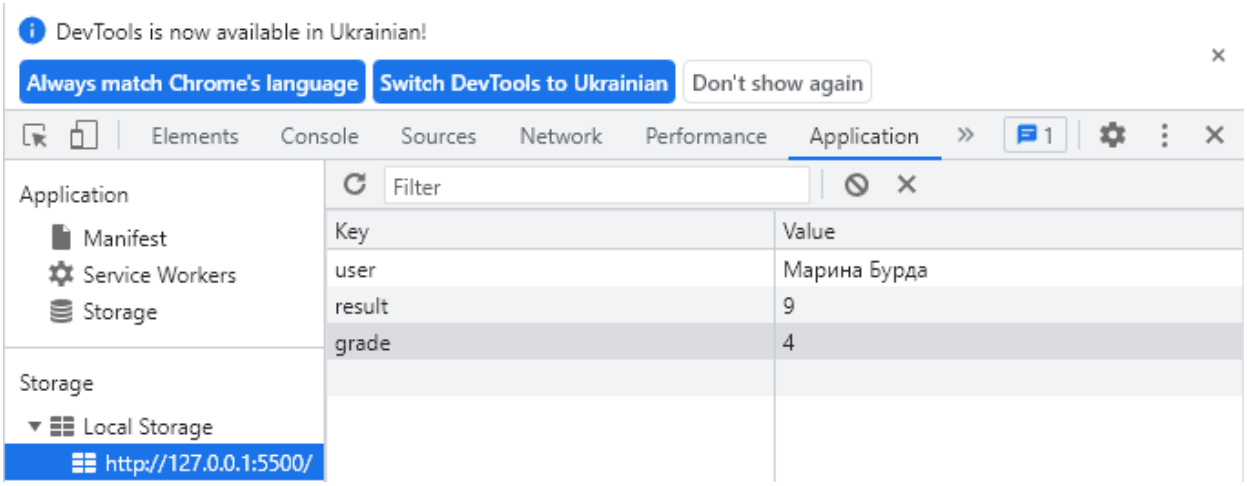

Рис. 5.8 – Збереження інформації в Local Storage

ВИСНОВКИ

У процесі виконання даної курсової роботи було успішно розроблено додаток для тестування з теми «Мова програмування Pascal».

Проведено аналіз задачі розробки програмного забезпечення для тестування. Також визначено, що таке програмне забезпечення є необхідним для ефективного тестування знань. Основним методом вирішення задачі стало розроблення програмного забезпечення з використанням JavaScript, що включало в себе взаємодію з користувачем та збереження результатів тестування в localStorage.

Спроектовано загальний алгоритм роботи програми, в якому описано класи, які доцільно створити і їх призначення. Створено блок-схему основної функції.

Також було розроблено функціональні алгоритми роботи програми, де було детальніше описані вимоги до додатку, коротко описаний його функціонал, створено дві блок-схеми потрібних функцій.

При розробці програмного забезпечення, розбили роботу на кроки, щоб було простіше і зручніше виконати завдання, і, безпосередньо, був написаний код.

В створенні керівництва користувача було детально описано, які дії потрібно виконати, щоб запустити додаток і як ним користуватися.

В результаті виконання курсової роботи було успішно досягнуто мети, а саме розроблено програмне забезпечення для тестування знань на мові JavaScript, яке дозволяє ефективно виконувати тестування та зберігати результати на стороні клієнта.

СПИСОК ЛІТЕРАТУРИ

1. Duckett, J. JavaScript and jQuery: Interactive Front-End Web Development. Wiley Publishing, 2014 – 640 с.
2. Niederst Robbins, J. Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics. O'Reilly Media, 2018 – 528 с.
3. GitHub: веб-сайт. URL: https://github.com/getify/You-Dont-Know-JS (дата звернення 15.02.2023)
4. MDN Web Docs: веб-сайт. URL: https://developer.mozilla.org (дата звернення: 27.01.2023)
5. W3Schools Online Web Tutorials: веб-сайт. URL: htpps://www.w3schools.com (дата звернення 05.03.2023)

ДОДАТКИ

Додаток А

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Тест</title>
    <link rel="stylesheet" type="text/css" href="./css/app.css" />
  </head>
  <body>
    <div class="wrapper">
      <main class="main">
        <div class="quiz__head">
          <div class="head__content" id="head"></div>
        </div>
        <div class="quiz__body">
          <div class="buttons">
            <div class="buttons__content" id="buttons">
              <button class="button">Default button</button><br />
              <button class="button button_wrong">Wrong answer</button><br />
              <button class="button button_correct">Correct answer</button><br />
              <button class="button button_passive">Unclicked button</button><br />
            </div>
          </div>
          <div class="quiz__footer">
            <div class="footer__content" id="pages">0 / 0</div>
          </div>
        </div>
      </main>
    </div>
    <script src="app.js"></script>
    <script src="q-a.js"></script>
  </body>
</html>
```

Додаток Б

```
const headElem = document.getElementById("head");
const buttonsElem = document.getElementById("buttons");
const pagesElem = document.getElementById("pages");
class Quiz {
	constructor(questions, results) {
		this.questions = questions;
		this.results = results;
		this.score = 0;
		this.result = 0;
		this.current = 0;
	}
	Click(index) {
		let value = this.questions[this.current].Click(index);
		this.score += value;
		let correct = -1;
		if (value >= 1) {
			correct = index;
		}
		else {
			for (let i = 0; i < this.questions[this.current].answers.length; i++) {

	if(this.questions[this.current].answers[i].value >= 1) {
					correct = i;
					break;
				}
			}
		}
		this.Next();
		return correct;
	}
	Next() {
		this.current++;
		if (this.current >= this.questions.length) {
			this.End();
		}
	}
	End() {
		for (let i = 0; i < this.results.length; i++) {

	if(this.results[i].Check(this.score)) {
				this.result = i;
			}
		}
	}
}
class Question {
	constructor(text, answers) {
		this.text = text;
		this.answers = answers;
	}
	Click(index) {
		return this.answers[index].value;
	}
}
class Answer {
	constructor(text, value) {
		this.text = text;
```

```
            this.value = value;
        }
}
class Result {
        constructor(text, value){
            this.text = text;
            this.value = value;
        }
        Check(value) {
            if (this.value <= value) {
                return true;
            }
            else {
                return false;
            }
        }
}
function Update() {
        if (quiz.current < quiz.questions.length) {
            headElem.innerHTML = quiz.questions[quiz.current].text;
            buttonsElem.innerHTML = "";
            for (let i = 0; i < quiz.questions[quiz.current].answers.length; i++) {
                let btn = document.createElement("button");
                btn.className = "button";
                btn.innerHTML = quiz.questions[quiz.current].answers[i].text;
            btn.setAttribute("index", i);
        buttonsElem.appendChild(btn);
            }
            pagesElem.innerHTML = (quiz.current + 1) + " / " + quiz.questions.length;
            Init();
        }
        else {
            buttonsElem.innerHTML = "";
            headElem.innerHTML = `Ваш результат: ` + quiz.results[quiz.result].text + ` бали/балів
            <br><button class="btn_end" onclick="location.reload()">Розпочати знову</button>
            <br><button class="btn_end" onclick="main()">На головну</button>`;

            pagesElem.innerHTML = "Кількість правильних відповідей: " + quiz.score;

            localStorage.setItem("grade", quiz.results[quiz.result].text);
            localStorage.setItem("result", quiz.score);
        }
}
function Init() {
        let btns = document.getElementsByClassName("button");
        for (let i = 0; i < btns.length; i++) {

        btns[i].addEventListener("click", function (e) { Click(e.target.getAttribute("index")); });
        }
}
```

```
function Click(index)  {
        let correct = quiz.Click(index);
        let btns =
document.getElementsByClassName("butto
n");
        for (let i = 0; i < btns.length; i++) {
                btns[i].className = "button
button_passive";
        }
        if (correct >= 0) {

        btns[correct].className = "button
button_correct";
        }
        if (index != correct) {
                        btns[index].className
= "button button_wrong";
        }
        else {
                btns[index].className =
"button button_correct";
        }
        setTimeout(Update, 1000);
}
function main() {
        window.location.assign("./index.html
");
}
function shuffle(array) {
        array.sort(() => Math.random() -
0.5);
}
```