МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КРИВОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій

Кафедра автоматизації, комп’ютерних наук і технологій

РОЗРАХУНКОВО-ГРАФІЧНА РОБОТА

з дисципліни «Веб-технології та веб-дизайн»

на тему: «Розробка системи онлайн тестування»

Виконав:

студент групи КН-18-1 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Криницький В.О.

Перевірив:

асистент кафедри АКНТ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Пилипенко О. В.

Консультант:

доцент кафедри КСМ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кузнєцов Д. І.

Оцінка/Бал \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / \_\_\_\_\_\_\_\_\_\_

Кривий Ріг

2021

|  |
| --- |
| *Криворізький національний університет* |
| (назва вищого закладу освіти) |

|  |  |
| --- | --- |
| Кафедра | *автоматизації, комп’ютерних наук і технологій* |

|  |  |
| --- | --- |
| Дисципліна | *Веб-технології та веб-дизайн* |

|  |  |
| --- | --- |
| Напрям підготовки | *122 Комп’ютерні науки* |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Курс | *третій* | Група | *КН-18* | Семестр | *шостий* |

ЗАВДАННЯ

на курсову роботу студента

|  |
| --- |
| *Криницький Владислав Олександрович* |
| (прізвище, ім’я, по батькові) |

|  |  |
| --- | --- |
| 1.Тема роботи | Розробка системи онлайн тестування |
|  | |
|  | |

|  |  |  |  |
| --- | --- | --- | --- |
| 2. Строк здачі студентом закінченої роботи | | *7 червня 2021* | |
| 3. Вихідні дані до роботи | *Тема: Розробка системи онлайн тестування* | |
|  | | |
|  | | |

|  |  |
| --- | --- |
| 4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці) |  |
| *Вступ* | |
| 1. *Теоретична частина* | |
| * 1. *Аналіз предметної області* | |
| * 1. *Визначення вимог до системи* | |
| 1. *Розробка WEB-системи* | |
| *2.1 Інструменти і технології розробки веб-сайтів* | |
| * + 1. *React* | |
| * + 1. *Angular* | |
| *2.1.2. Vue* | |
| *2.2. Практична реалізація веб-сайту* | |
| *2.2.1. Структура проекту* | |
| *2.2.2. Наповнення проекту* | |
| *2.3. Хостинг проекту* | |
| *Висновки* | |

|  |  |  |
| --- | --- | --- |
| 5.Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень) | |  |
| 6. Дата видачі завдання | 2 квітня 2021 | |

КАЛЕНДАРНИЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| №№ | Назва етапів курсової роботи | Строк виконання етапів роботи | Примітка |
| 11 | Отримання завдання на РГР роботу | 02.04.21 |  |
| 22 | Дослідження літератури та матеріалів на задану тума | 05.05.21 |  |
| 33 | Аналіз технічного завдання | 10.05.21 |  |
| 44 | Проектування структури та алгоритму роботи розроблюваної веб-системи | 20.05.21 |  |
| 55 | Розробка компонентів веб-системи | 25.05.21 |  |
| 66 | Розгортання та тестування розробленої веб-системи | 01.06.21 |  |
| 77 | Складання пояснювальної записки | 04.06.21 |  |
| 88 | Представлення закінченої роботи на перевірку | 07.06.21 |  |
| 99 | Захист розрахунково-графічної роботи | 07.06.21 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |
| --- | --- | --- |
| Студент |  | *Криницький В.О.* |
|  | (підпис) |  |

|  |  |  |
| --- | --- | --- |
| Керівник |  | *Пилипенко О.В.* |
|  | (підпис) | (прізвище, ім’я, по батькові) |

**ЗМІСТ**

[ВСТУП 5](#_Toc73916049)

[1. ТЕОРЕТИЧНА ЧАСТИНА 6](#_Toc73916050)

[1.1. Аналіз предметної області 6](#_Toc73916051)

[1.2. Визначення вимог до системи 7](#_Toc73916052)

[2. РОЗРОБКА WEB-СИСТЕМИ 10](#_Toc73916053)

[2.1. Інструменти і технології розробки веб-сайтів 10](#_Toc73916054)

[2.1.1. React 10](#_Toc73916055)

[2.1.2. Angular 12](#_Toc73916056)

[2.1.3. Vue 13](#_Toc73916057)

[2.1.4. Обґрунтування вибору веб-фреймворку 13](#_Toc73916058)

[2.2. Практична реалізація веб-сайту 14](#_Toc73916059)

[2.2.1. Структура проекту 15](#_Toc73916060)

[2.2.2. Наповнення проекту 17](#_Toc73916061)

[2.3. Хостинг проекту 21](#_Toc73916062)

[ВИСНОВОК 22](#_Toc73916063)

[СПИСОК ВИКОРИСТАННОЇ ЛІТЕРАТУРИ 23](#_Toc73916064)

[ДОДАТОК А – ПРИКЛАД ВИХІДНИГО КОДУ 24](#_Toc73916065)

# ВСТУП

Процес тестування знань учнів багато в чому може бути поліпшений за рахунок розробки автоматизованої онлайн-системи тестування. Дана робота описує особливості розробки системи подібного типу, які використовуються технології, а також аналізує сучасні підходи до розробки веб-сайтів. Об'єктом дослідження, які лежать в основі даної роботи, є автоматизація процесу тестування, а його предметом - розробка системи онлайн-тестування знань.

Завдання складання і проведення тесту завжди пов'язана з деякими матеріальними або часовими витратами. Проведення тестів з друкованими бланками пов'язано як з матеріальними, так і з часовими витратами, такими як папір, чорнило принтера, час пошуку аудиторії, час самого іспиту. Оцінка результатів таких тестів також пов'язана з часовими витратами зі збору бланків та підрахунку результатів. При цьому час, витрачений на підведення підсумків, зростає разом з кількістю учасників тесту. Для вчителів та інших людей, які займаються проведенням різних тестів регулярно, мінімізація витраченого на тестування часу та ресурсів може виявитися критично важливою.

Пандемія у 2020 році показала важливість дистанційної освіти та проблеми які постали під час організації цього процесу, особливо проблеми пов’язанні з практичними завданнями для перевірки знань. Існуючі на той момент системи не могли задовольнити потреби навчальних закладів. Тому системи онлайн тестування важливі в даний час і в майбутньому будуть ще більш важливі.

Мета даної роботи – розробити клієнтську частину системи онлайн-тестування з високою швидкістю роботи і аналітичними можливостями.

# ТЕОРЕТИЧНА ЧАСТИНА

## Аналіз предметної області

Проведення тестування з використанням паперових тестів давно є одним з пріоритетних способів оцінки успішності учнів в навчальних закладах. У школах і університетах тести можуть бути використані для проміжного контролю, а також для оцінки поведінки, психіки, профорієнтації або мотивацій учнів. Деякі державні іспити і олімпіади включають в себе тестову частину з вибором відповіді з декількох варіантів. Проведення таких тестів пов'язано з різними витратами, в першу чергу з часом на перевірку, пошук місця проведення, оповіщення учнів.

Для вирішення проблеми довгої перевірки тестів в останні кілька десятиліть стали набирати популярність типізовані тести зі спеціальними типографськими бланками. Такі бланки прискорюють процес перевірки та підведення результатів тестування, а відповіді в них записуються в певні поля, які потім розпізнаються за допомогою спеціального програмного забезпечення. Для якісного та об'єктивного аналізу результатів використовується аналітична система, яка обробляє результати, отримані в ході розпізнавання відповідей учасників. Такий підхід багато в чому прискорює процес проведення тестів, але він не є найефективнішим.

Система онлайн-тестування створюється з метою поєднання процесів створення, проведення і аналізу результатів тестів, тим самим підвищуючи ефективність самого процесу тестування.

У Європі та США паперові тести в освіті використовуються, в основному, для проміжної оцінки. Тести на оцінку поточної успішності відмінно підходять для негайного зворотного зв'язку, наприклад, для визначення того, наскільки учням або студентам зрозумілий недавно пройдений матеріал. У той же час тести, призначені для оцінки прогресу учнів за великий період часу, такі як фінальні іспити, можуть дати невідповідний дійсності результат. Незважаючи на суперечки, використання тестів як фінальних випробувань зростає, особливо в сфері вищої освіти. В Україні тестування займає серединне положення - воно використовується однаково часто як для оцінки проміжних результатів, так і для фінальних іспитів. Випускні іспити в школах після 9 і 11 класів (ЗНО) частково засновані на тестах з вибором відповіді з декількох варіантів. Окреме місце займають міжнародні тести в різних сферах. До них відносяться іспити на знання іноземної мови (IELTS, TOEFL), на загальний рівень знань та інші. Дані іспити включають як частини з тестом (c вибором відповіді), так і частини з вільним відповіддю, таким як твір, малюнок або рішення задачі.

В цілому, тести в освіті мають наступні характеристики:

* Поділ на секції
* Обмеженість за часом
* Різні типи питань
* Різні типи варіантів відповіді

До найбільш поширених типів питань відносяться:

* Текстові питання
* Питання з картинками
* Заповнення пропусків у тексті

Найбільш популярні типи відповідей до таких питань є:

* Єдина відповідь з числа перерахованих
* Множинний вибір з числа перерахованих
* Написання тексту
* Зіставлення

Проектована в рамках роботи система повинна мати більшість із зазначених поширених типів відповідей і питань.

## Визначення вимог до системи

Під системою онлайн-тестування мається на увазі веб-сайт, який надає його відвідувачам різні функції для успішного здійснення процесу тестування. Цей сайт, як правило, має кілька сторінок з динамічним вмістом, хоча може містити і статичні інформаційні сторінки. Здійснення взаємодії з користувачами проводиться за рахунок веб-сторінок з різними елементами управління, такими як посилання, поля для введення, кнопки, веб-форми та інші. Система тестування з аналітичними можливостями повинна володіти такими набором функцій:

* Реєстрація та авторизація користувачів
* Перегляд, створення, редагування тестів різних типів
* Віддалене проведення тестування учасників
* Перегляд і аналіз результатів

Розробляється в рамках роботи система повинна відповідати зазначеному вище опису.

Виходячи з мети роботи, головною вимогою до розроблюваної системі є швидкість використання. У поточний момент відомий один підхід до розробки веб-сайту, при якому взаємодія з ним стає найбільш швидким. Цей підхід має назву SPA (від англ. Single Page Application) і він полягає в тому, що веб-додаток знаходиться на одній сторінці, а інтерактивність здійснюється за рахунок зміни окремих її частин. Таким чином, при необхідності змінити наведену інформацію, веб-браузер не перезавантажує сторінку повністю, а лише замінює вміст певних елементів. [1]

Повне перезавантаження сторінки веб-браузером пов'язана з високими затратами часу, так як при кожному оновленні браузер змушений заново завантажувати розміщені на сторінці зображення, підключені скрипти JavaScript і каскадні таблиці стилів CSS. Більш того, при повній перезавантаження браузер заново перемальовує всі елементи на сторінці, що може сильно позначитися на швидкості роботи сайту, якщо кількість елементів на сторінці велике.

З огляду на той факт, що при створенні тесту на сторінці будуть генеруватися велике число інтерактивних елементів (поля для введення назви тексту, поля для додавання секцій, питань і відповідей, кнопки, списки, що випадають і т.п.), рекомендується використовувати SPA підхід як архітектури клієнтської частини системи, що розробляється. [3]

Отримання даних сайтом в SPA підході здійснюється за рахунок використання асинхронних HTTP-запитів до сервера, які підтримуються у всіх браузерах за замовчуванням. Запит здійснюється за допомогою JavaScript (скорочено JS) - мови, який використовується для маніпуляції елементами сторінки. Підхід до обміну даними між сервером і веб-додатком за допомогою асинхронних запитів в середовищі веб-розробки має назву AJAX (від англ. Asynchronous Javascript and XML). Хоча в назві і фігурує XML, в реальності більшість веб-додатків обмінюються даними в форматі JSON (від англ. JavaScript Object Notation). Дані, отримані клієнтською частиною програми в даному форматі не вимагають додаткової обробки, так як по суті вони є десеріалізованними JS-об'єктами. Таким чином, відображення веб-додатки в браузері може бути змінено в реальному часі ґрунтуючись на даних, отриманих з сервера, без перезавантаження сторінки. Існують кілька різних бібліотек на мові Javascript, які створені з метою спростити створення односторінкових веб-додатків. Як правило, вони надають необхідну для роботи сайту архітектуру і можуть бути доповнені різними іншими модулями виходячи з бізнес-вимог програми. Серед таких бібліотек таки відомі фреймворки, як React, Angular, Vue і інші. У другому розділі будуть розглянуті деякі з цих бібліотек і буде обґрунтований вибір однієї з них в якості основи клієнтської частини системи. [9]

# РОЗРОБКА WEB-СИСТЕМИ

## Інструменти і технології розробки веб-сайтів

За останні 3-5 років технологічне забезпечення в сфері веб-розробки швидко змінилася. Технології, які вважалися новими (HTML5, CSS3 і JQuery), були поступово витіснені потужними JavaScript-фреймворками, які дозволяють розробникам створювати інтерактивні і зручні у використанні програми. Деякі JS-бібліотеки надають різні програмні методи представлення даних, що значно підвищує загальну продуктивність сайту. Є кілька JS-бібліотек, які стали відомими за рахунок своїх переваг: React, Angular і View. Одна з них буде використовуватися в цьому проекті. [8]

### React

Цей JavaScript-фреймворк розроблений компанією Facebook і спрямований на те, щоб зробити маніпулювання даними на веб-сторінці якомога швидшим. В основі підходу React лежить поділ всієї веб-сторінки на більш дрібні компоненти. Компоненти можуть бути розміщені всередині один одного, але кожен з них буде мати свій власний контекст даних. Такий підхід спрощує оновлення інформації на веб-сторінці шляхом перемальовування компонента в залежності від динамічно змінюваних даних.

Ще однією перевагою React є його синтаксис розмітки, званий JSX. Ця мова дуже схожий на HTML і XML, але має деякі додаткові можливості. JSX дозволяє створювати динамічну компоновку даних в розмітці сторінки і вбудовувати компоненти в чистий HTML. Розробникам може бути зручніше писати скорочений код компонентів всередині HTML, який потім будуть замінений повним кодом вбудованого компонента. Наприклад, це поєднання HTML і JSX буде виводити заголовок з текстом «Тема»:

<h1>Тема</h1>

Крім того, JSX дозволяє вбудовувати код JS в макет для виведення динамічних даних. Для реалізації цього коду код повинен бути укладений у фігурні дужки. Наприклад, цей фрагмент коду видасть поточну дату:

<p> { Date() } </ p>

Компонентний код зберігається в об'єкті JavaScript, який обов'язково успадковує клас «React.Component» і імплементує функцію «render». Ця функція повертає JSX-код компонента, який описує, як він повинен відображатися в веб-браузері, і які динамічні дані слід вставляти в макет. Властивості, що передаються компоненту ззовні, можуть бути доступні в коді компонента за рахунок звернення до властивості props.propertyName. [4]

Кожен компонент має свій власний стан. Стан може використовуватися для зберігання різних параметрів компонента, в залежності від яких він буде перемальований веб-браузером заново, наприклад, рівень доступу користувача або його логін. Компоненти можуть бути вкладені одна в одну, створюючи деревовидну архітектуру програми, аналогічну дереву DOM (від англ. Document Object Model - об'єктна модель документа) веб-браузера. У React існує спеціальна логіка для управління деревом компонентів, звана Virtual DOM. Ця частина функціональності фреймворку React є механізмом зберігання даних про те, які елементи показуються відвідувачеві сайту. Ці дані, подібно DOM, зберігаються у вигляді дерева. Всякий раз, коли стан компонента змінюється, React викликає його функцію render () для перемальовування вмісту компонента. Потім він визначає, де саме дані повинні бути замінені, і робить вибіркові зміни як в Virtual DOM, так і в браузерному DOM-дереві. React робить найменшу можливу кількість маніпуляцій з DOM, зводячи до мінімуму використання ЦП та графічного процесора, що, в свою чергу, прискорює взаємодію з веб-додатком. Таким чином, використання стану компонента для управління його поданням спрощує створення додатків, що залежать від даних. React.JS також має функції для витягання призначеного для користувача введення з веб-форм і прив'язки вхідних значень до станів компонентів. Функціональність цього фреймворка також може бути легко розширена з використанням сторонніх бібліотек. [7]

### Angular

Структура цього фреймворка є комплексом складних відносин між багатьма об'єктами, які представляють собою окремі рівні архітектури коду. Це означає, що з точки зору розробників код логічно поділяється на різні категорії - інжектори, компоненти, шаблони, директиви та інші. Такий поділ на функціональні блоки дозволяє розробникам створювати легко масштабовані додатки.

Кожне веб-додаток Angular має, принаймні, один модуль, умовно званий «AppModule». Модулі допомагають організувати архітектуру програми в єдині функціональні блоки.

Компоненти, подібні React, є логічними елементами функціональності веб-додатки. Однак в Angular відсутня функція рендеринга, оскільки розмітка HTML зберігається в окремій суті, званої Template (шаблон).

Angular має розширений шаблонний синтаксис з безліччю логічних операторів, вбудованих в макет. Ці оператори дозволяють розробникам створювати інтерактивні макети, подібні JSX у React. У цій структурі компоненти управляються з різних модулів, які в кінцевому підсумку утворюють SPA. Зв'язок між компонентами і модулями виконується з використанням спеціальних об'єктів, званих метаданими. Вони визначають, який шаблон, селектор, модуль і постачальники присутні у кожного компонента.

Angular також має двосторонню прив'язку даних, яка автоматично змінює дані в шаблоні, показуваному користувачеві, коли змінюються дані в модулі. Ця прив'язка працює і у зворотному напрямку - Angular змінює дані в модулі, коли користувач робить введення даних в поля для введення або по-іншому взаємодіє з інтерактивними елементами управління.

Angular відомий своєю масштабністю і архітектурної строгістю. Він використовується багатьма сайтами з високим навантаженням і вважається одним з найскладніших веб-фреймворків.

### Vue

Vue - це прогресивний фреймворк, що підходить для створення користувацьких інтерфейсів. Назва фреймворка співзвучно з view, тобто з поданням.

Vue - це фреймворк, який використовується для вирішення завдань саме рівня уявлення, його просто інтегрувати з іншими бібліотеками та проектами. Іншими словами, Vue - це інструмент, який можна впроваджувати поступово. Можливість впровадження пов'язана з тим, що Vue прагне до прогресивності: підтримку Vue можна додати у вже існуючий проект, завдяки чому його функціональність буде значно розширена. І це відрізняє його від інших фреймворків.

Простота інтеграції - це одне з основних переваг даного фреймворка, особливо в поєднанні з можливістю інтеграції з бекенд фреймворками. Інші сильні сторони Vue - це простота в освоєнні і хороша документація, а також висока продуктивність.

Також Vue можна використовувати для створення односторінкових додатків. Центральна концепція Vue - це концепція компонентів, тобто невеликих частин інтерфейсу користувача, які можна використовувати повторно. Таким чином, і сам додаток складається з частин-компонентів. Один компонент може включати кілька інших компонентів, тобто використовується деревоподібна ієрархія. Vue - це реактивний фреймворк: уявлення (view) автоматично змінюється при зміні змінної (моделі). Vue має дуже хорошу екосистему, яка включає в себе безліч корисних сторонніх компонентів і інструментів.

### Обґрунтування вибору веб-фреймворку

Клієнтське рішення для цього проекту має забезпечувати високу швидкість роботи і мати можливість малювати безліч елементів на сторінці, ґрунтуючись на даних. Оскільки основним фактором успіху для цього проекту з точки зору веб-додатки є швидкість, то основними кандидатами є JavaScript-фреймворки React і Angular. Ці фреймворки виявляються найбільш ефективними для односторінкових додатків (SPA). Для реалізації цього проекту зручніше використовувати React, так як він має кілька особливих переваг. По-перше, React забезпечує зручну архітектуру, керовану компонентами, яка спростить рендеринг складних тестів з багатьма розділами і різними типами питань. По-друге, ця структура підвищує продуктивність веб-сайту з використанням JSX і Virtual DOM. Для проекту більш зручно використовувати вбудовування компонентів JSX, пропоноване React, оскільки модульна система, яка використовується в Angular, здається занадто складною для такого проекту, як онлайнова система тестування. React - це легка бібліотека, і це ідеальне рішення для потреб цього проекту, в той час як функціональність Angular здається надмірною. [6]

## Практична реалізація веб-сайту

Використовуваний в системі тестування фреймворк дозволяє створювати складні додатки за рахунок використання компонентів, які можна вбудовувати один в одного. Рендер сторінки відбувається після того, як браузер завантажить сторінку, тобто спочатку на сторінці не присутній нічого крім одного порожнього блоку, в який потім буде вставлено вміст програми. Те, що саме буде вставлено в блок, описано в виконуваному файлі "bundle.js". Цей файл є результатом компіляції іншого файлу, написаного з використанням спеціального синтаксису JSX, а також різних нових можливостей мови JavaScript, які ще не в повній мірі підтримуються браузерами. Вхідний файл до компіляції містить логіку по вставці якогось компонента в порожній блок на сторінці. Вбудований компонент може включати в себе інші компоненти, викликаючи відповідні методи по відображенні даних. У вхідному файлі "index.js" відбувається вставка певного компонента в порожній блок:

import React from 'react';

import ReactDOM from 'react-dom';

import './index.scss';

import 'bootstrap/dist/css/bootstrap.min.css';

import App from './App';

ReactDOM.render(

    <App />,

    document.getElementById('root')

);

У цьому файлі спочатку імпортуються всі необхідні залежності, а потім виконуються рендер компонента «App» у елемент з тегом «root».

### Структура проекту

На початку розробка SPA за допомогою мови програмування JavaScript та бібліотеки React необхідно створити структуру проекту для більш зручної розробки. На рисунку 2.2.1 показана структура проекту тестової системи.

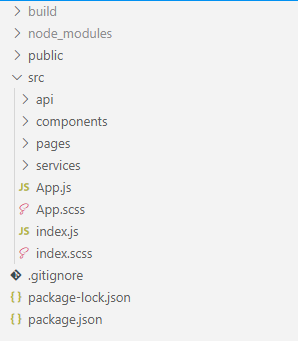


Рисунок 2.2.1. Структура проекту.

Весь вихідний код проекту знаходиться у папці src. У папці node\_modules знаходяться необхідні залежності (пакети) для зборки та роботи сайту, та у папці build знаходяться вже готові до публікації файли, тобто скомпільовані та оптимізовані.

У папці components знаходяться всі компоненти, що створені для функціювання сайту, також у папці pages знаходяться компоненти, що є сторінками.

Всі функції, класи та об’єкти необхідні для зв’язку з backend знаходяться у папці api, але при розробці даного проекту акцентується увага на створені клієнтської частини, тому backend нема і в папці api знаходяться так звані «фейкові» функції та об’єкти для «взаємодії» між клієнтом і сервером.

У папці services знаходяться допоміжні функції та об’єкти для роботи сайту, наприклад сервіс валідації, який перевіряє правильність даних, що вводить користувач.

Головним компонентом сайту є компонент App, за допомогою якого відображаються інші компоненти:

function App() {

    return (

        <AuthProvider>

            <HashRouter>

                <NavBar />

                <Container>

                    <Switch>

                        <Route exact path="/login" component={Login} />

                        <Route exact path="/register" component={Register} />

                        <PrivateRoute exact path="/">

                            <h1 className="bg-primary p-2 m-2 text-white text-center">Home</h1>

                        </PrivateRoute>

                        <PrivateRoute exact path="/about">

                            <h1 className="bg-secondary p-2 m-2 text-white text-center">About</h1>

                        </PrivateRoute>

                        <PrivateRoute exact path="/logout">

                            <Logout />

                        </PrivateRoute>

                        <PrivateRoute exact path="/completed-tests">

                            <PassedTests />

                        </PrivateRoute>

                        <PrivateRoute exact path="/user">

                            <UserProfile />

                        </PrivateRoute>

                        <PrivateRoute exact path="/tests/:id">

                            <TestViewer />

                        </PrivateRoute>

                        <PrivateRoute exact path="/assigned-tests">

                            <AssignedTests />

                        </PrivateRoute>

                        <Route>

                            <h1 className="bg-warning p-2 m-2 text-white text-center">Not found</h1>

                        </Route>

                    </Switch>

                </Container>

            </HashRouter>

        </AuthProvider>

    );

}

AuthProvider – це компонент-оболонка, цей компонент слугує для того, щоб передавати дані користувача всім дочірнім компонентам. Ця передача здійснюється через механізм контексту бібліотеки React.

Router служить для того, щоб при переході по посиланнях в браузері змінювався URL підмінявся і зберігалася історія відвідування з можливістю подальшого повернення назад без перезавантаження сторінки. Кожен окремий шлях в додатку описується компонентом Route, що задає правило шляху і конкретний компонент, який буде завантажений при переході по шляху.

### Наповнення проекту

Головна сторінка знаходиться за адресую "/" починаючи від кореня веб-додатки і відкривається автоматично при спробі перейти в корінь сайту. На цій сторінці розташована інформація про

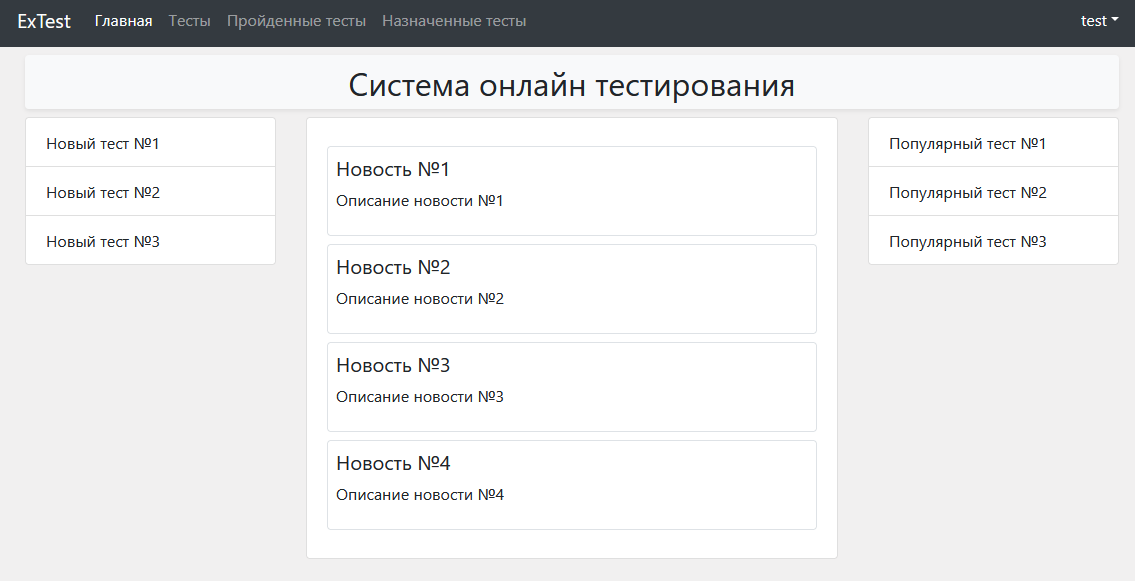


Рисунок 2.2.2. Головна сторінка

У разі, якщо користувач не авторизувався, йому показується форма введення логіна і пароля (рис. 2.2.3).

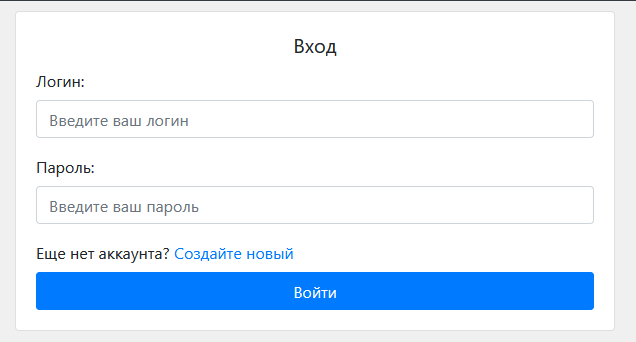


Рисунок 2.2.3. Форма входу

Якщо користувач не має акаунту є можливість створити новий натиснувши кнопку «Створити новий», після цього відкриється форма реєстрації (рис. 2.2.4), та з’явиться можливість зареєструватися як учень або викладач.

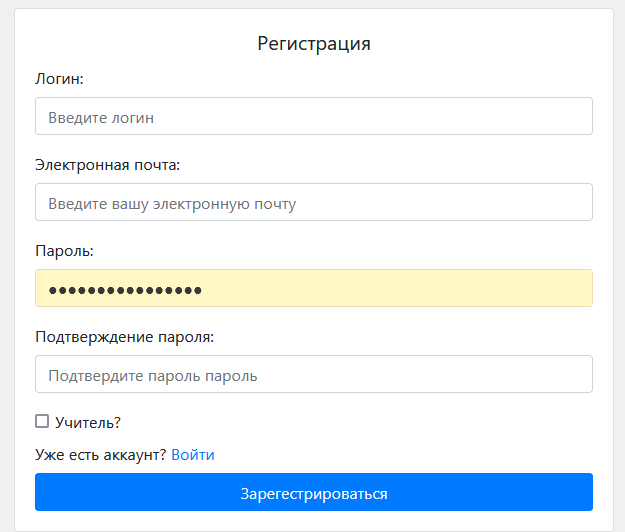


Рисунок 2.2.4. Форма реєстрації

Після того як користувач зареєструвався у нього з’являється можливість, якщо це учень, проходити тести та дивитися результати вже пройдених тестів, якщо це вчитель, створювати нові тести та назначати тести учням.

Приклад проходження тесту:

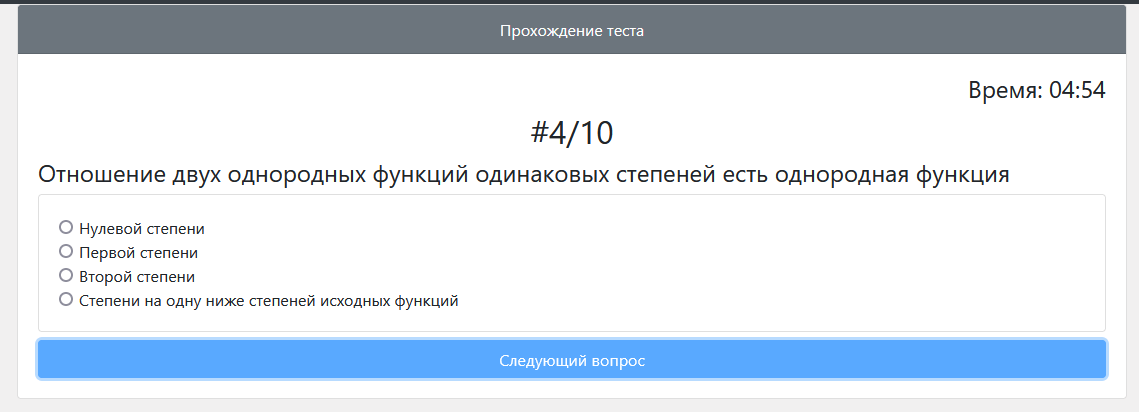


Рисунок 2.2.5. Проходження тесту

Всі сторінки, що відображаються, складаються з компонентів. Наприклад, сторінка пройдених тестів (рис. 2.2.6) складається з таких компонентів: Navbar, CardWrapper, TestList, CardList та TestItem.

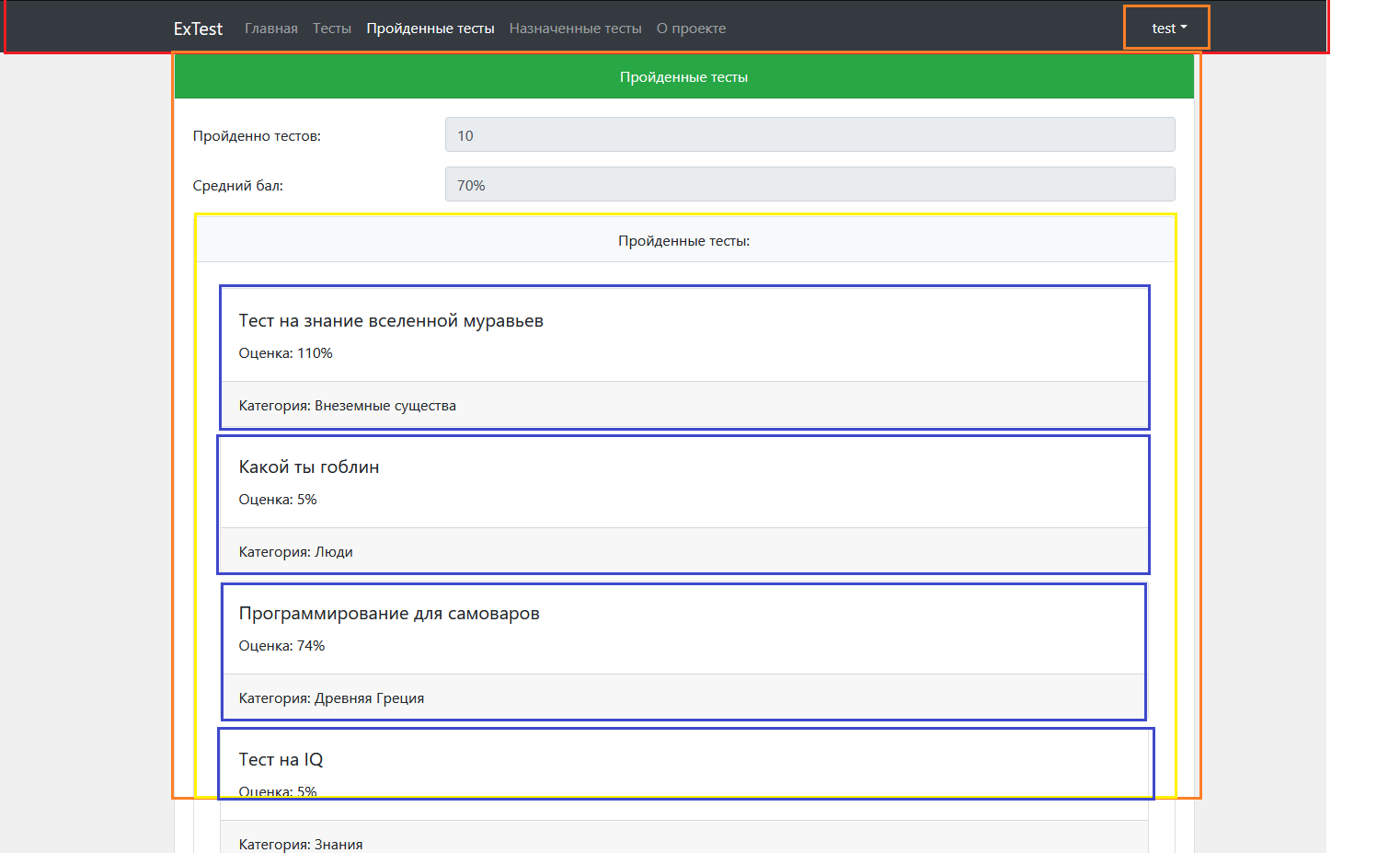


Рисунок 2.2.6. Приклад розбиття на компоненти.

Після проходження тесту є можливість переглянути всі відповіді, що зробив учень, та сумарну кількість балів (рисунок 2.2.7).

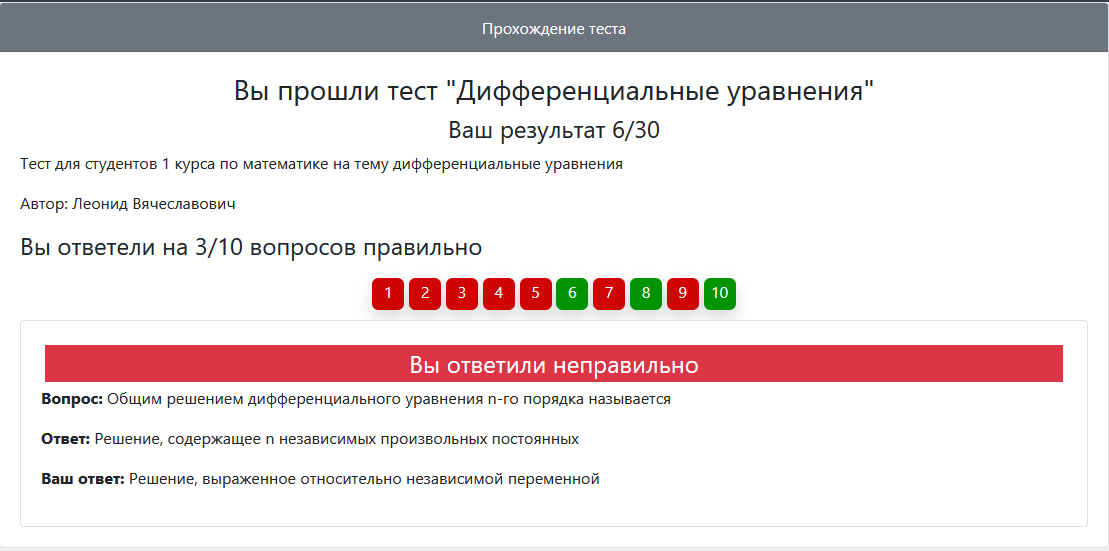


Рисунок 2.2.7. Результати тестів.

У користувача зареєстрованого як вчитель є можливість створювати нові тести, на спеціальній сторінці є можливість задати назву, обрати категорію, та додати питання, до кожного питання можна додавати відповіді та обрати яка з них буде правильною:

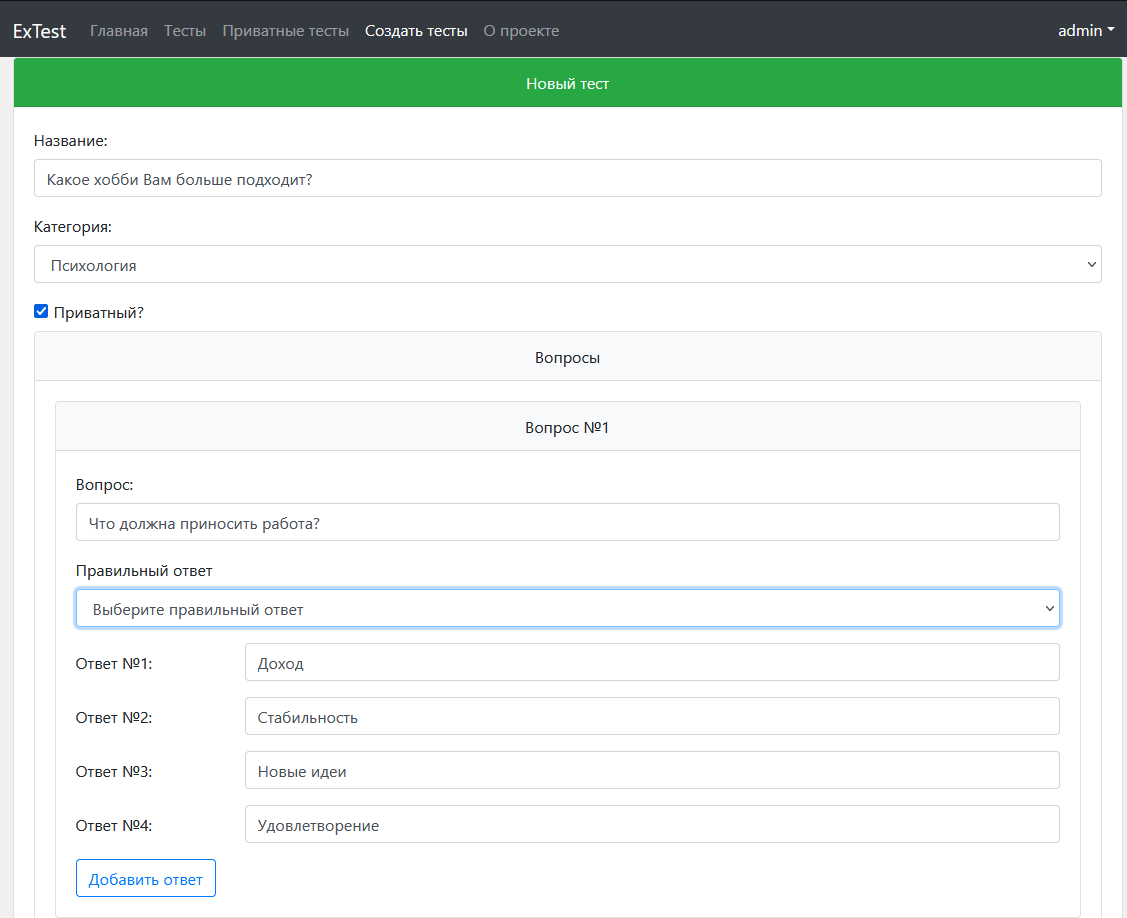


Рисунок 2.2.8. Створення тестів

Для створення функціоналу сайту також використовувались сторонні бібліотеки. Для обробки вводу та валідації форм використовувалась бібліотека Formik. Для можливості переходити по різним сторінкам react-router-dom. Для можливості компіляції препроцессору мови SASS – node-sass, та для використання готових компонентів react зі застосуванням bootstrap – react-bootstrap.

Приклад вихідного коду сайту наведено у Додатку А.

Github: <https://github.com/Neadv/rgr>

## Хостинг проекту

Для того щоб сайт функціонував користувачу потрібно лише завантажити один html файл та один js файл, то є можливість використати безплатний хостинг через Github – Githup pages, який дозволяє створити репозиторій та загрузити туди статичні файли для сайту, які потім будуть віддаватися клієнтам по запиту на сайт. Для того щоб швидко загрузити сайт на Github Pages використовувався пакет gh-pages. Адрес сайту - <https://neadv.github.io/rgr/#/>

# ВИСНОВКИ

На підставі проблем, пов'язаних з проведенням тестування з використанням паперових бланків, були виділені вимоги до розроблюваної системі онлайн-тестування. Було прийнято рішення про максимізації швидкості роботи призначеного для користувача інтерфейсу для досягнення переваги перед існуючими онлайн-системами.

В роботі було дано обґрунтований вибір набору технологій, що використовуються для розробки веб-сайту і аналітичної системи. Аналіз двох фреймворків React, Angular і Vue дозволив зробити висновки про те, що перший з них куди більше зручний для розробки системи онлайн-тестування. Він відрізняється швидкістю за рахунок використання розумного механізму по маніпуляції DOM, а також широкими можливостями по роботі з даними і розширенню сторонніми бібліотеками.

В результаті цієї роботи створено онлайн-додаток, що дозволить його користувачам створювати, проходити і переглядати результати тестів.

# СПИСОК ВИКОРИСТАННОЇ ЛІТЕРАТУРИ

1. Современный учебник JavaScript [Електронний ресурс] - <https://learn.javascript.ru/>
2. *Фаулер, Мартин.* Рефакторинг кода на JavaScript: улучшение проекта существующего кода, 2-е изд. — М.: «Диалектика», 2019. — 464 с.
3. *Флэваrан, Дэвид.* JavaScrirt: карманный справочник, 3-е изд. — М., 2013. — 320 с.
4. React – JavaScript-библиотека для создания пользовательских интерфейсов [Електронний ресурс] - <https://ru.reactjs.org/>
5. Create React App Docs [Електронний ресурс] - <https://create-react-app.dev/docs/getting-started>
6. *Томас Марк Тиленс.* React в действии. — СПб.: «Питер», 2019. — С. 368.
7. *Кирупа Чиннатамби.* Изучаем React. — СПб.: «Питер», 2019. — С. 368.
8. *Фримен Эрик, Фримен Элизабет.* Изучаем HTML, XHTML и CSS = Head First HTML with CSS & XHTML. — П.: «Питер», 2010. — 656 с.
9. *Стивен Шафер.* HTML, XHTML и CSS. Библия пользователя, 5-е издание = HTML, XHTML, and CSS Bible, 5th Edition. — М.: «Диалектика», 2010. — 656 с.
10. Руководство по React [Електронний ресурс] - <https://metanit.com/web/react/>

# ДОДАТОК А – ПРИКЛАД ВИХІДНИГО КОДУ

App.js

import { Container } from 'react-bootstrap';

import { HashRouter, Route, Switch } from 'react-router-dom';

import './App.scss';

import { AuthProvider } from './components/general/AuthProvider';

import { NavBar } from './components/NavBar/NavBar';

import { PrivateRoute } from './components/general/PrivateRoute';

import { Login } from './pages/Login/Login';

import { Logout } from './pages/Logout/Logout';

import { Register } from './pages/Register/Register';

import { UserProfile } from './pages/UserProfile/UserProfile';

import { TestViewer } from './pages/TestViewer/TestViewer';

import { PassedTests } from './pages/PassedTests/PassedTests';

import { AssignedTests } from './pages/AssignedTests/AssignedTests';

import { Category } from './pages/Category/Category';

import { SelectedCategorty } from './pages/Category/SelectedCategory';

import { AddTest } from './pages/AddTest/AddTest';

import { Home } from './pages/Home/Home';

function App() {

    return (

        <AuthProvider>

            <HashRouter>

                <NavBar />

                <Container>

                    <Switch>

                        <Route exact path="/login" component={Login} />

                        <Route exact path="/register" component={Register} />

                        <PrivateRoute exact path="/">

                            <Home />

                        </PrivateRoute>

                        <PrivateRoute exact path="/logout">

                            <Logout />

                        </PrivateRoute>

                        <PrivateRoute exact path="/completed-tests">

                            <PassedTests />

                        </PrivateRoute>

                        <PrivateRoute exact path="/user">

                            <UserProfile />

                        </PrivateRoute>

                        <PrivateRoute exact path="/tests/:id">

                            <TestViewer />

                        </PrivateRoute>

                        <PrivateRoute exact path="/assigned-tests">

                            <AssignedTests />

                        </PrivateRoute>

                        <PrivateRoute exact path="/category">

                            <Category />

                        </PrivateRoute>

                        <PrivateRoute exact path="/category/:id">

                            <SelectedCategorty />

                        </PrivateRoute>

                        <PrivateRoute exact path="/test/add">

                            <AddTest />

                        </PrivateRoute>

                        <Route>

                            <h1 className="bg-warning p-2 m-2 text-white text-center">Not found</h1>

                        </Route>

                    </Switch>

                </Container>

            </HashRouter>

        </AuthProvider>

    );

}

export default App;

Home.jsx

import { useEffect, useState } from "react";

import { Card, Col, Container, ListGroup, Row } from "react-bootstrap";

import { useHistory } from "react-router";

import \* as homeApi from '../../api/homeApi';

function Home() {

    const [newTests, setNewTests] = useState([]);

    const [popularTests, setPopularTests] = useState([]);

    const [news, setNews] = useState([]);

    const history = useHistory();

    function handleClick(id){

        history.push("/tests/" + id);

    }

    useEffect(() => {

        setNewTests(homeApi.newTests);

        setPopularTests(homeApi.popularTests);

        setNews(homeApi.news);

    }, []);

    return (

        <Container className="p-2">

            <h2 className="text-center bg-light rounded shadow-sm p-2">Система онлайн тестирования</h2>

            <Row>

                <Col sm={3}>

                    <ListGroup>

                        {newTests.map((n, index) => (

                            <ListGroup.Item key={index} style={{cursor: "pointer"}} onClick={() => handleClick(index)}>{n}</ListGroup.Item>

                        ))}

                    </ListGroup>

                </Col>

                <Col>

                    <Card>

                        <Card.Body>

                            {news.map((n, id) => (

                                <div key={id} className="my-2 border p-2 rounded news" style={{cursor: "pointer"}}>

                                    <h5>{n.title}</h5>

                                    <p>{n.desc}</p>

                                </div>

                            ))}

                        </Card.Body>

                    </Card>

                </Col>

                <Col sm={3}>

                <ListGroup>

                        {popularTests.map((n, index) => (

                            <ListGroup.Item key={index} style={{cursor: "pointer"}} onClick={() => handleClick(index)}>{n}</ListGroup.Item>

                        ))}

                    </ListGroup>

                </Col>

            </Row>

        </Container>

    );

}

export { Home };