Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мультимедійних систем факультету інформатики

Розробка веб-інтерфейсу системи тестування знань з використанням бібліотек React і Redux

Текстова частина до курсової роботи за спеціальністю «Комп'ютерні науки» 122

Керівник курсової роботи
Демківський Є.О.
_____
(підпис)
"____" _____ 2020 р.

Виконав студент
Возбранний Р.С.
"____" _____ 2020 р.

Київ 2020

Тема: Розробка веб-інтерфейсів з використанням бібліотеки React.js

Календарний план виконання роботи:

№	Назва етапу	Термін виконання	Примітка
1.	Отримання теми курсової роботи	08.10.2019	
2.	Вивчення предметної області	01.11.2019	
3.	Огляд існуючих рішень	01.12.2019	
4	Огляд засобів реалізації	10.01.2020	
5.	Вивчення технологій для розробки	01.02.2020	
6.	Побудова технічного завдання	15.02.2020	
7.	Написання першої частини курсової роботи	06.03.2020	
8.	Написання другої частини курсової роботи	25.03.2020	
9.	Написання третьої частини курсової роботи	15.04.2020	
10.	Написання висновків курсової роботи	25.04.2020	
11.	Перегляд змісту роботи з керівником	01.05.2020	
12.	Внесення змін до курсової роботи відповідно до зауважень наукового керівника	06.05.2020	
13.	Створення презентації	10.05.2020	
14.	Захист роботи		

Студент Возбранний Р.С.
Керівник Демківський Є.О.

“____” _____

Зміст

Вступ

Для того щоб оцінити знання співробітників або кандидатів на посаду, компанії використовують різні методи. Багато компаній використовують тестові завдання для перевірки практичних знань. Це найкращий спосіб проаналізувати компетентність кандидата для вакантної посади, оцінити темп роботи, продуктивність, психологічну стійкість майбутнього співробітника. Проте, часто люди відмовляються від виконання тестового завдання через брак часу або недостатній рівень довіри до компанії. Інший спосіб перевірки знань, який обирають роботодавці – це тестування безпосередньо на співбесіді. В цьому випадку роботодавець ризикує витратити зайвий час на кандидатів, які не здатні пройти тест.

Автоматична система тестування допоможе кандидатам скоротити час на проходження тестів в різних компаніях, пройти тест у зручний час, використати результати тестування повторно. Після проходження тесту користувачу буде запропоновано зробити результат публічним. А роботодавці зможуть використовувати інформацію про пройдені користувачами тести під час підбору персоналу.

React.js – відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка покликана розв'язувати проблеми часткового оновлення вмісту веб-сторінки, з якими стикаються в розробці односторінкових застосунків. За даними Google Trends на 2020 рік, React.js – найпопулярніший серед розробників інструмент для створення веб інтерфейсів. Саме це визначило мій вибір інструментів розробки.

Об’єкт дослідження: бібліотеки React.js та Redux. Мета даної роботи полягає у вивченні та засвоєнні нових навичок у роботі з React.js та допоміжними інструментами: Redux, Styled-components, Formik, Prism.js, React Bootstrap.

Розділ 1. Огляд існуючих рішень

1.1 Опис функцій існуючих систем тестування

1.1.1 Google Форми

Найрозповсюдженішим способом тестування є складання тестів в Google Формах. Це зручний інструмент, за допомогою якого можна легко і швидко складати опитування та анкети, а також збирати іншу інформацію.

Google Форми – безкоштовний інструмент з інтуїтивно зрозумілим дизайном і великою кількістю функцій, необхідних для проведення тестування та аналізу результатів великої кількості користувачів.

Доступні налаштування тесту:

- опції показу оцінки за тести (одразу після надсилання форми або пізніше, після перевірки вручну);
- налаштування видимості інформації про запитання (видимість незарахованих відповідей, правильних відповідей та кількості балів за запитання);
- можливість встановити вагу кожного питання, яка буде враховуватись при виставленні фінальної оцінки за тест.

Всі типи питань форми (рисунок 1.1.1.1):

- з короткими відповідями;
- абзац;
- з варіантами відповіді;
- прапорці:
- спадний список;
- лінійна шкала;
- таблиця з варіантами відповіді;
- сітка прапорців;
- дата;

- час;
- завантаження файлу.

Типи питань, для яких доступна автоматична перевірка:

1. Короткий текст.
2. З одним правильним варіантом відповіді.
3. З кількома правильними варіантами.
4. Спадний список.
5. Таблиця з одним правильним варіантом відповіді.
6. Таблиця з кількома правильними варіантами.

Переглядати й аналізувати результати користувачів може адміністратор, обрані адміністратором користувачі або всі, кому було надане посилання на перегляд. Результати користувачів доступні у вкладці «Відповіді», або у прикріпленій до форми Google таблиці. [11]

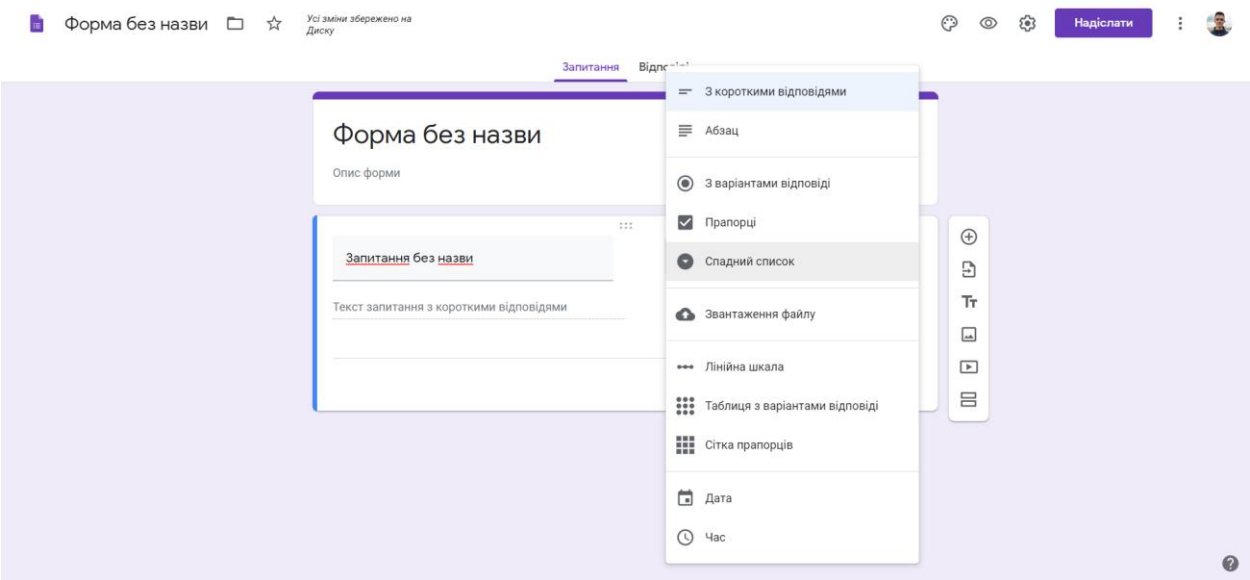

Рис. 1.1.1.1. Типи запитань Google форм

1.1.2 Платформа Techgig

Techgig – це платформа, призначена виключно для IT галузі, яка надає широкий спектр послуг, ключовими з яких є проведення тестувань і процес підбору працівників на основі їхніх результатів. Автоматизована оцінка знань користувачів доступна для більше ніж 52-х мов програмування. Система пропонує лише два типи запитань: з вибором однієї та декількох правильних відповідей (рисунок 1.1.2.1).

Користувач може пройти тест за обраним напрямком раз на місяць. Нова спроба стає доступною першого числа кожного місяця. Мінімальний пакет послуг коштує $979,05 за рік.

Основний функціонал системи тестувань:

- генерація URL для створення запрошення на проходження тесту;
- велика, структурована бібліотека питань;
- просте створення тесту з нуля або з використанням запропонованих системою питань;
- механізм для запобігання списуванню під час проходження тестів кандидатами;
- панель управління з ролями для зручної роботи декількох спеціалістів по підбору персоналу. [12]

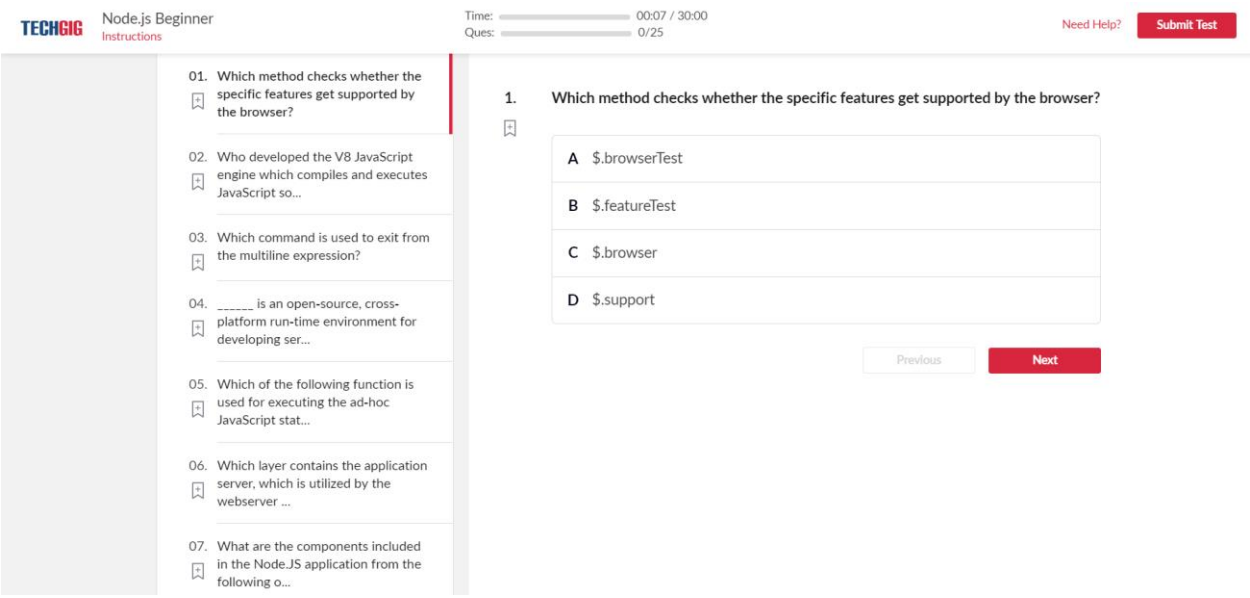

Рис. 1.1.2.1. Сторінка тесту поатформи Techgig

1.1.3 Платформа iSpring

iSpring – платформа для онлайн-навчання і тестування співробітників. Тест створюється на платформі та призначається співробітникам. iSpring перевіряє відповіді та показує у звітах, хто набрав прохідний бал і які помилки в тесті допустив. Редактор тестів включає 14 типів питань (рисунок 1.1.3.1).

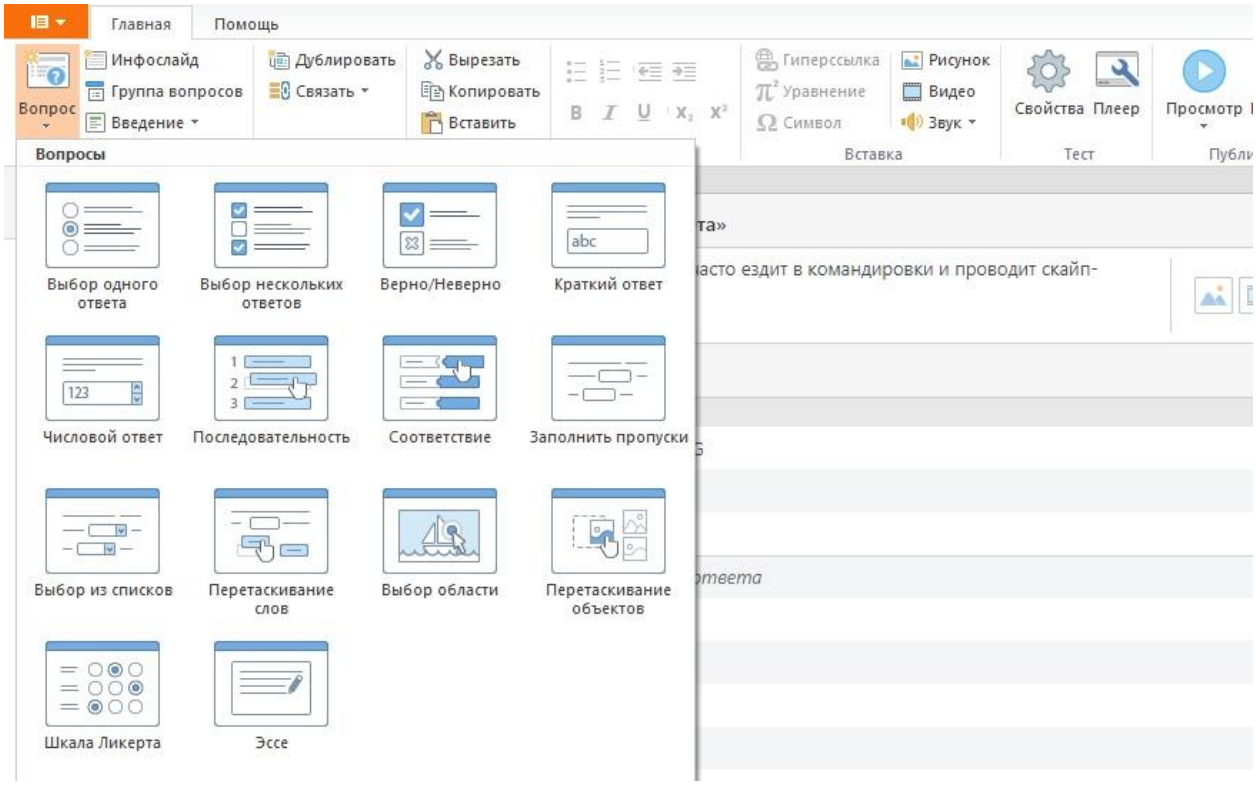

Рис. 1.1.3.1. Типи запитань платформи iSpring

Доступні налаштування тесту: адміністратор може задати бали і штрафи, вказати кількість спроб і обмежити час відповіді на кожне питання.

Система дозволяє увімкнути автоматичне відправлення результатів тесту на електронну пошту, сервер або систему дистанційного навчання. Адміністратор може налаштувати, в якому форматі відправити звіт про проходження тесту (тільки набрані бали або детальний звіт з правильними відповідями та коментарями по кожному питанню).

Доступна функція випадкового вибору питань. Якщо тест включає питання з різних тем, їх можна структурувати і об'єднати в групи для того щоб відобразити користувачу випадкову вибірку груп або питань з кожної групи. Це дозволяє створити свій варіант тесту для кожного користувача. Мінімальний пакет послуг коштує $370 за рік. [13]

1.1.4 Вступний тест Binary Studio Academy

Деякі компанії самі створюють системи тестування для власних потреб. Так зробила компанія Binary Studio. Тест використовується для попереднього відбору охочих отримати можливість навчатися в академії.

Всього доступно п’ять напрямків: JavaScript, PHP, QA, .NET, Java. Тест має обмеження у часі і систему виявлення нечесних дій користувача (покидання сторінки або копіювання тексту питання (рисунок 1.1.4.1)). Використовується питання з одною і декількома правильними відповідями. Тест доступний у визначений проміжок часу. [14]

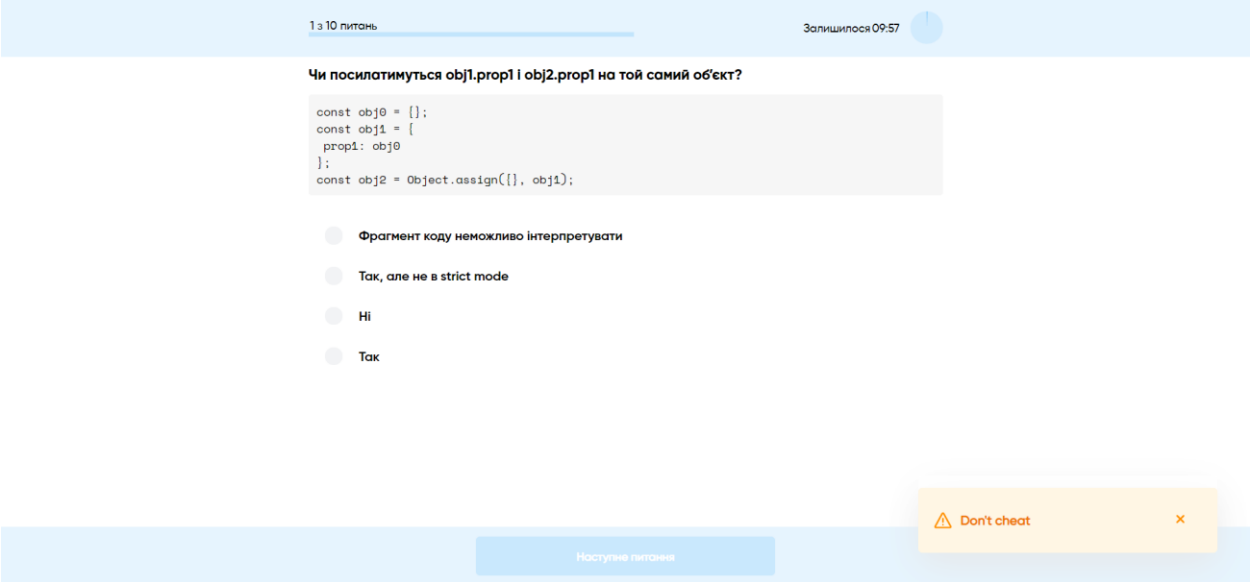

Рис. 1.1.4.1. Сторінка тесту поатформи Binary Studio Academy

1.2 Аналіз функцій існуючих рішень та визначення доцільності розробки нової платформи

Дуже часто процес відбору працівників у компанію забирає багато часу і зусиль. Невеликі компанії не можуть дозволити собі розробку власних рішень для відбору працівників, а готові рішення є достатньо дорогими. Кандидати, у свою чергу, під час пошуку роботи надсилають свої резюме в багато різних

компаній одночасно і витрачають купу часу і зусиль на проходження співбесід, тестів і виконання тестових завдань.

В результаті аналізу функцій описаних систем було виявлено, що основним завданням більшості є автоматизація створення унікальних тестів для окремих замовників. Натомість, ключовою ідеєю власної розробки є створення системи для проходження уніфікованих тестів, результати проходження яких зможуть переглядати і аналізувати представники різних компаній.

Завданням розробленої системи є забезпечення якісного тестування знань користувачів з метою подальшого спрощення процесу працевлаштування. Для того щоб точність результатів тестування можна було вважати достатньою, в системі мають бути використані механізми для запобігання нечесного виконання завдань. Для автоматичного оцінювання робіт, було вирішено залишити лише ті типи запитань, які можуть бути перевірені без втручання людини: питання з вибором однієї або декількох відповідей. Користувачу заборонено проходити тест на одну тему частіше ніж визначений часовий інтервал. Після проходження тесту і ознайомлення з результатом, йому буде запропоновано зробити результат публічним. Після підтвердження публікації, оцінка стане доступною для перегляду на відповідній сторінці з результатами користувачів.

Розділ 2. Огляд використаних технологій та інструментів

2.1 React.js

React.js – одна з найпопулярніших бібліотек для створення складних Frontend застосунків. Одна з головних особливостей React.js – свобода дій, існує величезна кількість підходів до побудови додатків з його допомогою (redux, mobx і інші). Спочатку React призначався для вебу, для створення

веб-сайтів, проте пізніше з'явилася платформа React Native, яка вже призначалася для мобільних пристроїв.

React – ідеальний інструмент для створення масштабованих веб-додатків, особливо SPA (односторінкових застосунків). React відносно простий в освоєнні, має зрозумілий та лаконічний синтаксис.

React використовує віртуальний DOM (VDOM) – це концепція програмування, в якій «віртуальне» уявлення призначеного для користувача інтерфейсу зберігається в пам'яті і синхронізується з «справжнім» DOM за допомогою бібліотеки ReactDOM. Цей процес називається узгодженням. Ви вказуєте, в якому стані повинен перебувати призначений для користувача інтерфейс, а React забезпечує відповідність реального DOM цьому стану. Віртуальний DOM представляє легку копію звичайного DOM. І відмінною рисою React є те, що дана бібліотека працює саме з віртуальним DOM, а не звичайним. У підсумку така схема взаємодії з елементами веб-сторінки працює набагато швидше і ефективніше, ніж якби ми працювали з JavaScript з DOM безпосередньо.

Іншою відмінною рисою бібліотеки є концентрація на компонентах – ми можемо створити окремі компоненти і потім їх легко переносити з проекту в проект. Ще одна особливість React – використання JSX. JSX представляє комбінацію коду JavaScript і XML і надає простий і інтуїтивно зрозумілий спосіб для визначення коду візуального інтерфейсу. React JSX трансформує XML-подібний синтаксис в JavaScript. [1]

При розробці було вирішено використовувати тільки функціональні компоненти. Функціональні компоненти на основі хуків більш наочні за класи та мають майже всі можливості реакт компоненту. З хуками з’являється можливість перевикористання логіки з відстеження стану без зміни ієрархії компонентів. Через це стає легко ділитися хуками поміж іншими компонентами.

2.2 Бібліотека Redux

Redux – відкрита JavaScript бібліотека призначена для управління сховищем програми. Найчастіше використовується разом з React або Angular для побудови інтерфейсів користувача. Redux використовує три основні концепції: єдине джерело істини, стан тільки для читання, зміна стану за допомогою функцій.

При використанні Redux, основні дані для всього додатку представлені єдиним JavaScript об'єктом з посиланням на стан або дерево станів. Цей об'єкт може бути простим або складним, залежно від вимог програми. Незалежно від розміру застосунку, всі дані стану зберігаються в одному об'єкті.

Стан призначений тільки для читання і єдиний спосіб його змінити – відправити action (об’єкт який описує що сталося). Оскільки редьюсери – це функції, можна контролювати порядок їх надсилання, передавати додаткові дані або створювати повторювані редьюсери (reducers). Редьюсери визначають як повинен змінитись стан після того як відбулась подія.

За замовчуванням action creators в Redux не підтримують асинхронні дії, такі як отримання даних, тому було використано Redux Thunk. Це middleware бібліотека, яка дозволяє викликати action creator, повертаючи при цьому функцію замість об'єкта. Функція приймає методи dispatch як аргумент, щоб після того, як асинхронна операція завершиться, використовувати його для діспатчінга звичайного синхронного екшену, всередині тіла функції. [2][4]

2.3 Бібліотека Styled Components

Styled Components – бібліотека-надбудова над CSS. Вона розбирає стилі CSS які визначені в JavaScript і створює відповідні JSX елементи. Цей

підхід дозволяє писати CSS, який буде застосований лише для одного компонента, уникаючи використання пов’язування елемента зі стилем за допомогою className.

Styled-components покладаються на літерали шаблону JavaScript, що дозволяє писати справжній CSS з використанням основних переваг препроцесорів, таких як вкладення та багато інших. Бібліотека автоматично створює унікальне ім’я класу для заданих стилів, тому при використанні доводиться перейматися про конфлікти імен класів. Styled-components дозволяє динамічно змінювати стилі на основі пропсів (вхідних даних компонента) або глобальної теми. Бібліотека дозволяє стилізувати теги, або готові реакт компоненти, стилі яких будуть унаслідувані. [8]

2.4 Бібліотека Formik

Форми завжди були складним елементом інтерфейсу користувача. Їх потрібно валідувати, відправляти, ініціалізувати з початковими даними. В React немає простих DOM-форм, на onSubmit яких можна отримати об’єкт з даними. Дотримуючись ідеї компонентної архітектури, потрібно створювати компонент форми, який буде зберігати значення всіх полів і стан відправлення. Поля, у свою чергу – контрольовані компоненти, які можуть мати асинхронну валідацію, виведення помилок, внутрішню логіку і до того ж можуть бути перевикористані.

Для спрощення роботи з формами використовується Formik. Бібліотека базується на техніці render props. Доступно два варіанти використання: через HOC або компонент з render prop. Основні компоненти: Formik (Компонент з усією логікою та render prop), withFormik (High-Order Component), Field (модернізований компонент <input>).

Formik дає повну свободу вибору бібліотеки валідації. Валідація за замовчуванням проводиться на кожну зміну поля і на зміну фокуса. В

документації використовується бібліотека yup для створення валідаційних схем. Для ситуації, коли валідація в інтерфейсі вдала, але сервер відповів помилкою, можна скористатися вбудованим функціоналом setStatus або setErrors, щоб передати об’єкт помилок з середини функції надсилання форми.

Для обробки надсилання форми потрібно написати функцію, яка приймає values та formikBag як аргументи. Values – це дані полів форми. FormikBag – набір функцій для маніпулювання станом форми. Використовуючи formik, потрібно самому описати функціонал зміни стану форми під час зміни етапів надсилання. Наприклад, після надсилання даних на сервер необхідно викликати setSubmitting(true), після отримання відповіді — setSubmitting(false). [7]

2.5 Бібліотека Prism.js

Prism.js – це легка JavaScript бібліотека для забезпечення виділення коду на веб-сайтах. Prism підтримує 210 різних мов програмування (HTML, XML, CSS, PHP, JavaScript і багато інших). На офіційному сайті представлено 8 тем оформлення коду. Всі вони у відкритому доступі. Також з бібліотекою можна використовувати додаткові скрипти та стилі, які дозволяють додати нумерацію рядків, відображення назви мови, в якій відображається код, та багато інших. Особливої уваги заслуговує плагін «Autoloader», який дозволяє автоматично підвантажувати потрібні стилі для підсвічування синтаксису блоків коду. [9]

2.6 Бібліотека React-bootstrap

Bootstrap – це безкоштовний набір інструментів з відкритим кодом для розробки за допомогою HTML, CSS і JS, який містить шаблони для форм, кнопок, навігації та інших компонентів інтерфейсу, а також

додаткові розширення JavaScript. Він спрощує розробку динамічних веб-сайтів і веб-додатків.

Bootstrap має модульну структуру і складається переважно з наборів таблиць стилів LESS, які реалізують різні компоненти цього набору інструментів. Розробники можуть самостійно налаштовувати файли Bootstrap, обираючи компоненти для свого проекту.

Основні інструменти Bootstrap:

- сітки (grid) – наперед задані, готові до використання колонки;
- типографіка (typography) – опис та визначення класів для шрифтів, таких як шрифти для коду, цитат тощо;
- мультимедіа (media) – засоби управління зображеннями та відео;
- таблиці (table) – засоби оформлення таблиць, які зокрема забезпечують сортування;
- форми (form) – класи для оформлення як форм, так і деяких подій;
- навігація (nav, navbar) – класи для оформлення вкладок, сторінок, меню і панелей навігації;
- сповіщення (alert) – класи для оформлення діалогових вікон, підказок і спливаючих вікон.

React-Bootstrap створений для заміни Bootstrap JavaScript в React проектах. Кожен елемент був переписаний як React компонент без використання бібліотеки jQuery. [6]

Розділ 3. Програмна реалізація платформи

3.1 Технічне завдання

3.1.1 Основні вимоги

3.1.1.1 Ролі користувачів системи

Ролі користувачів системи: гість, авторизований користувач, адміністратор. Функціонал, доступний для неавторизованих користувачів (гостей сайту):

- перегляд всіх доступних для проходження тестів;
- авторизація.

Для авторизованих користувачів:

- перегляд всіх доступних для проходження тестів;
- проходження тестів;
- перегляд власних результатів;
- публікація власних результатів;
- перегляд списку всіх публічних результатів користувачів сайту.

Для адміністраторів:

- перегляд всіх доступних для проходження тестів;
- проходження тестувань;
- перегляд власних результатів;
- публікація власних результатів;
- перегляд списку всіх публічних результатів користувачів сайту;
- створення тестів.

3.1.1.2 Опис функціоналу сторінок та елементів інтерфейсу

Сторінка каталогу доступних тестів:

- список доступних тестів;

- відображення короткої інформації про кожен тест у списку (назва, короткий опис, час на проходження тесту);
- для кожного елементу списку повинна бути кнопка переходу на сторінку перегляду детальної інформації про тест;
- пошук по тестах (за назвою).

Сторінка перегляду детальної інформації про тест:

- назва;
- опис;
- правила проходження тесту;
- кнопка «Start», для запуску тесту;
- після натискання кнопки «Start» користувачу має бути показане модальне вікно для підтвердження початку тестування.

Сторінка тесту:

- список всіх питань тесту з можливістю перемикатися по ним;
- текст питання;
- код з підсвіченим синтаксисом;
- варіанти відповіді з можливістю вибору однієї або декількох правильних відповідей;
- кнопки навігації по питаннях;
- кнопка надсилання тесту на оцінювання;
- панель інформації, яка має містити назву тесту, зворотний відлік часу до закінчення і кнопка скасування проходження тесту;
- при натисканні кнопки скасування проходження тесту, користувачу має бути показане модальне вікно для підтвердження скасування спроби;
- відслідковування переходу зі сторінки та копіювання тексту, для запобігання нечесного проходження тесту.

Сторінка перегляду результату пройденого тесту:

- назва тесту;

- дата і час проходження;
- набрані бали або статус «Canceled», якщо тест був скасований або завершений некоректно;
- кнопка для повернення до головної сторінки сайту.

Сторінка перегляду власних результатів проходження тестів:

- список результатів пройдених тестів авторизованого користувача;
- для кожного результату зі списку повинна бути відображена назва тесту, дата проходження і результат (або статус «Canceled»);
- для кожного результату зі списку має бути перемикач, який буде відображати та контролювати публічність результату;
- після перемикання перемикача публічності, користувачу повинно бути показане модальне вікно з підтвердженням зміни статусу публічності.

Сторінка перегляду результатів тестування всіх користувачів:

- список результатів всіх користувачів;
- кожен елемент списку повинен містити назву тесту, оцінку за тест, дату проходження, ім’я користувача, результат якого відображається.

Сторінка створення та редагування тесту:

- сторінка доступна тільки адміністратору;
- має містити поля для вводу інформації про тест;
- до тесту можна додати будь-яку кількість питань;
- до кожного питання можна додати будь-яку кількість відповідей;

Сторінка профілю має містити інформація про обліковий запис авторизованого користувача.

Головна панель навігації:

- має бути присутньою на всіх сторінках;
- не відображається під час проходження тесту;
- для не авторизованого користувача містить посилання на головну сторінку, сторінку перегляду всіх тестів та на сторінки авторизації;

- для авторизованого користувача – посилання на головну сторінку, сторінку перегляду всіх тестів, сторінку перегляду власних результатів проходження тестів, сторінку перегляду всіх опублікованих результатів користувачів, профіль та кнопку виходу з облікового запису;
- для адміністратора – посилання на головну сторінку, сторінку перегляду всіх тестів, сторінку перегляду власних результатів проходження тестів, сторінку створення тесту, сторінку перегляду всіх опублікованих результатів користувачів, профіль та кнопку виходу з облікового запису.

3.1.2 Заплановані допрацювання

Розробка REST API для застосунку:

- авторизація користувачів з використанням JWT (JSON Web Token);
- CRUD (create, read, update, delete) – 4 базові функції управління даними «створення, зчитування, зміна і видалення»;
- система тестування, основними функціями якої є забезпечення перевірки виконаних тестів і контроль дозволів на проходження тестів користувачами.

3.1.3 Перспективи розвитку

Можливі покращення:

- розробка панелі адміністратора;
- розширення списку типів завдань, зокрема завданням з написання коду користувачем;
- додавання ролі роботодавця та функціоналу пов’язаного з можливістю комунікації роботодавців та працівників за допомогою сайту;

- показ рекомендацій (ресурсів для навчання або пропозицій від роботодавців) на основі результатів тестування.

3.2 Розробка системи

3.2.1 Початкові налаштування

3.2.1.1 Створення проекту за допомогою Create React App:

Для створення проекту було використано command line interface create-react-app. Create React App – зручний інструмент для швидкого створення React застосунку. Цей інструмент створює проект з вже налаштованими Webpack, Babel та іншими інструментами розробки. Команда для створення проекту: `npx create-react-app app-name.`

3.2.1.2 Підключення маршрутизації React Router:

Після того як проект створено, необхідно налаштувати маршрутизацію. У React є своя система маршрутизації, яка дозволяє зіставляти запити до додатка з певними компонентами. Ключовою ланкою в роботі маршрутизації є модуль react-router, який містить основний функціонал по роботі з маршрутизацією. Однак для роботи у браузері, знадобиться ще й модуль react-router-dom. Команда для установки: `npm install --save react-router-dom.`

React router пропонує три основні компоненти для налаштування маршрутизації в додатку: BrowserRouter, Route, Link, Switch. Для початку необхідно обгорнути головний компонент застосунку у файлі App.jsx в BrowserRouter для того щоб всередині став доступним об’єкт History, який дозволяє відстежувати адресу сторінки.

Компонент Route повинен приймати два параметри: path і component. Фактично це визначає який компонент повинен бути відображений при

переході по вказаному маршруті у браузері. Компонент Switch дозволяє вибрати один компонент для рендерингу. Без нього Router може показувати кілька компонентів для одного маршруту, якщо вони відповідають вказаному запиту.

Для навігації по сторінках застосунку використовується компонент Link. Якби посилання були створені за допомогою звичайного посилання (тегу a), натискання на них призвело б до перезавантаження сторінки. Компонент Link допомагає це попередити. При натисканні на Link URL-адреса буде оновленою, а контент сторінки буде оновлений без перезавантаження сторінки.

Оскільки було вирішено використовувати функціональні компоненти, для подальшої роботи з маршрутизацією будуть використовуватись хуки React Router 5. Всі вони імпортуються з модуля react-router-dom. Хуки – це функції, що дозволяють використовувати стан і функції життєвого циклу класу в функціональних компонентах. Для того щоб отримати доступ до об’єкта history буде використано функцію useHistory. Для отримання адреси використовується useLocation. Для отримання параметрів з рядка адреси використовується useParams. [3]

3.2.1.3 Встановлення Redux:

Після налаштування маршрутизації, саме час забезпечити застосунок сховищем стану. Необхідно завантажити Redux і допоміжні модулі:

- Redux – основна бібліотека;
- React Redux – модуль для роботи Redux з React;
- Redux Thunk – асинхронний middleware для Redux.

Команда для установки: `npm i redux redux-thunk react-redux`. В Redux загальний стан застосунку представлений одним об’єктом JavaScript – state (стан) або state tree (дерево станів). Незмінне дерево станів доступне тільки для читання. Зміни можливі тільки при надсиланні action (дії). Дія

(action) – це JavaScript-об’єкт, який коротко описує суть зміни. Єдиною вимогою до цього об’єкта є наявність властивості type, значенням якої зазвичай є рядок. Генератори дій (actions creators) – це функції, які створюють дії. Редюсер (reducer) – це чиста функція, яка обраховує наступний стан дерева на основі його попереднього стану і застосованої дії. Редюсер повертає новий об’єкт дерева станів, яким замінюється попередній. Сховище (store) – це об’єкт, який зберігає стан застосунку. Отримати стан можна викликавши функцію getState(). Єдиним способом оновити стає є виклик функції dispatch. Сховище дозволяє підписуватись і слідкувати за змінами в стані за допомогою функції subscribe. Схема роботи redux: (рисунок 3.2.1.3.1). [2][4][5]

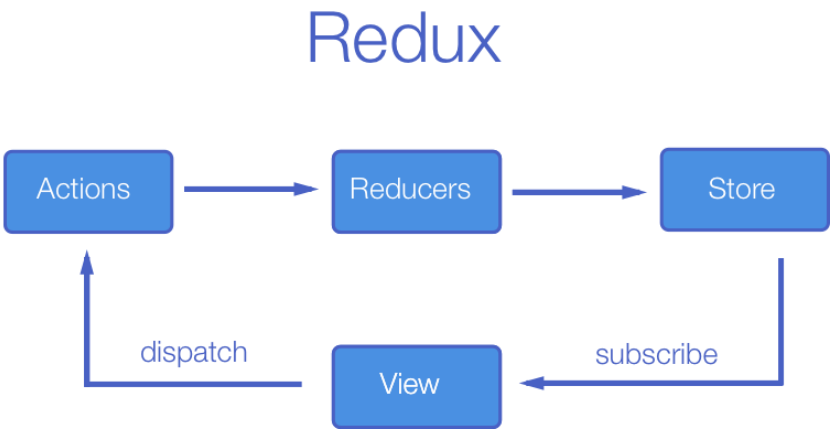

Рис. 3.2.1.3.1. Схема роботи redux

3.2.1.4 Встановлення React-bootstrap:

Для використання фреймворку bootstrap необхідно встановити відповідні пакети: react-bootstrap bootstrap. Після установки, у файлах index.js або App.js необхідно під'єднати стилі до проекту: `import 'bootstrap/dist/css/bootstrap.min.css';`.

3.2.2 Налаштування сховища Redux

У файлі index.js необхідно обгорнути головний компонент застосунку App провайдером. Компонент Provider, який імпортовано з бібліотеки react-redux, повинен приймати об’єкт сховища (рисунок 3.2.2.1).

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';
import { Provider } from 'react-redux'
import configureStore from './store/configureStore';

const store = configureStore();

ReactDOM.render(
  <Provider store={store}>
    <App />
  </Provider>,
  document.getElementById( elementId: 'root')
);
```

Рис. 3.2.2.1. Файл index.js

Функція для налаштування сховища (configureStore) була винесена в окремий файл. Бібліотечна функція createStore приймає rootReducer, який буде створено пізніше, initialState (початковий стан сховища), якщо такий вказаний та функцію applyMiddleware, яка дозволяє під'єднати redux-thunk (рисунок 3.2.2.2).

```
import { createStore, applyMiddleware } from 'redux';
import thunk from 'redux-thunk';
import rootReducer from './reducers/rootReducer';

export default function configureStore(initialState) {
  return createStore(
    rootReducer,
    initialState,
    applyMiddleware(thunk)
  );
}
```

Рис. 3.2.2.2. Функція configureStore

Файл rootReducer створений для поєднання всіх редюсерів застосунку в один. Для його створення використовується функція combineReducers, яка імпортується з бібліотеки redux і приймає на вхід об’єкт в якому визначені всі редюсери застосунку. (рисунок 3.2.2.3) [2][4][5]

```
import authReducer from './authReducer';
import testReducer from './testReducer';
import testInfoReducer from './testInfoReducer';
import testScoreReducer from './testScoreReducer';
import { combineReducers } from 'redux';

const rootReducer = combineReducers( reducers: {
  auth: authReducer,
  test: testReducer,
  testInfo: testInfoReducer,
  testScore: testScoreReducer
});

export default rootReducer;
```

Рис. 3.2.2.3. Функція combineReducers

3.2.3 Авторизація

Найпростіший підхід для реалізації аутентифікації при використанні це надсилання логіна і пароля при кожному запиті. Зрозуміло, що такий

спосіб не забезпечує достатній рівень безпеки даних, особливо при використанні незахищеного протоколу. Для уникнення проблем з безпекою було використано JWT.

JSON Web Token – відкритий стандарт для створення токенів доступу, які засновані на форматі JSON. Використовуються для передачі даних для аутентифікації в клієнт-серверних застосунках. Токени створюються сервером, підписуються секретним ключем і передаються клієнту, який використовує їх для подальшого підтвердження особистості.[10]

Для роботи з авторизацією користувача створено reducer (authReducer.js) і action creator (authActions.js). authReducer (рисунок 3.2.3.1) змінює стан сховища, для відображення необхідної інформації про стан авторизації користувача:

- авторизація (AUTH_USER) – дозволяє записати інформацію про користувача в змінну сховища user;
- помилка при авторизації (AUTH_ERROR) – записує текст помилки, яка виникла при обробці запиту на сервері;
- запит в процесі обробки (AUTH_LOADING) – записує в змінну isLoading значення true/false.

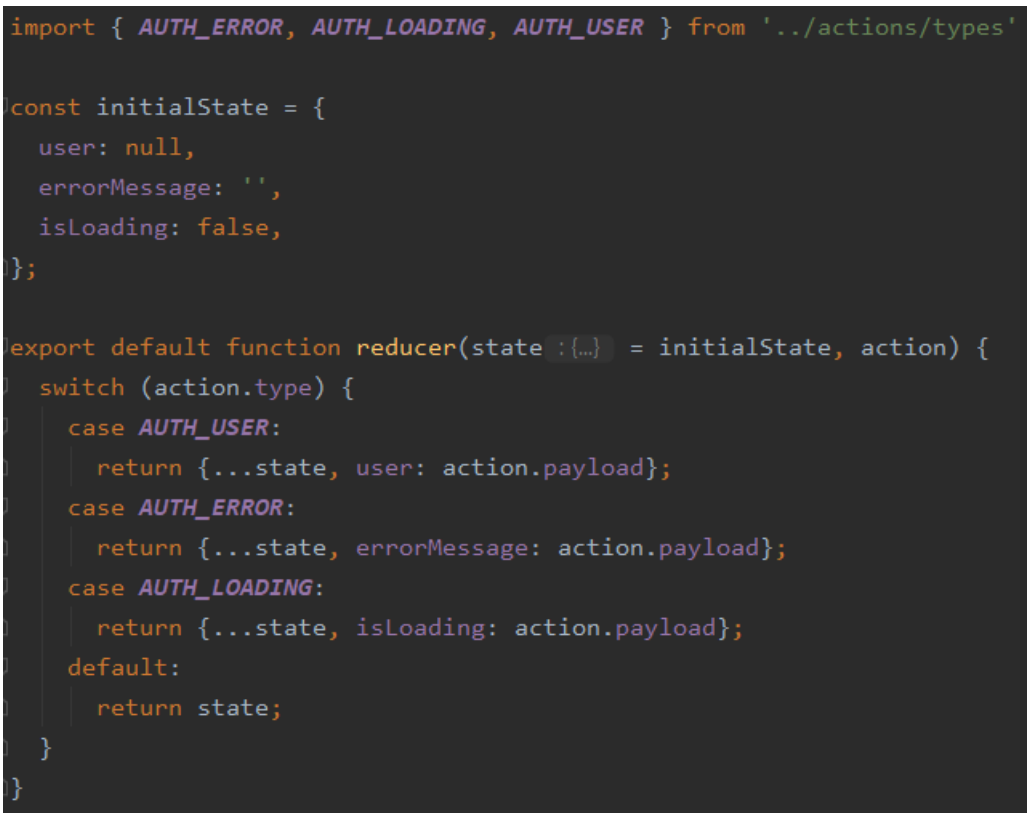

Рис. 3.2.3.1. Файл authReducer

У файлі authActions відбуваються всі запити до сервера, і створюються дії, для зміни стану. При логіні (рисунок 3.2.3.2) або реєстрації за допомогою redux-thunk та axios виконується асинхронний запит до сервера. Якщо сервер повертає помилку, викликається дія authError, в яку записується текст помилки. Якщо авторизація пройшла успішно, сервер надсилає у відповідь токен, і об’єкт користувача. Токен відразу записується до localstorage.

Рис. 3.2.3.2. Функція loginFetch

Функція fetchUser (рисунок 3.2.3.3), створена для отримання інформації про уористувача, токен якого зберігається в localstorage. Ця функція викликається кожен раз при завантаженні застосунку. Для цього в компоненті App.jsx використано метод useEffect. Якщо токен недійсний, відбувається очищення localstorage, і logout користувача.

Рис. 3.2.3.3. Функція fetchUser

Для авторизації та реєстрації створені дві окремі сторінки (компоненти Login і SignUp). Форми на цих сторінках реалізовані за допомогою Formik і схеми для валідації yup.

Панель навігації для авторизованих і неавторизованих користувачів відображаєтьсья по різному. Авторизованим користувачам доступні посилання на сторінку перегляду списку всіх доступних тестів, сторінку перегляду власних результатів тестування, і сторінку перегляду всіх публічних результатів користувачів сайту, але недоступні кнопки для переходу на сторінки аутентифікації. Для гостей сайту відображаються лише посилання на сторінку всіх тестів і кнопки переходу на сторінки аутентифікації. Було створено різні компоненти для відображення

посилань для авторизованих і неавторизованих користувачів: GuestLinks і UserLinks, відображення яких контролюється компонентом AppBar (рисунок 3.2.3.4).

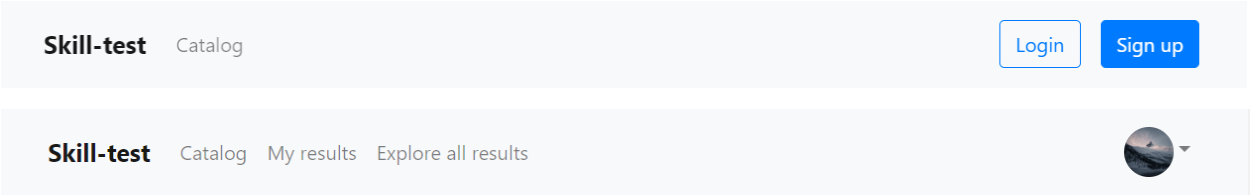

Рис. 3.2.3.4. Варіанти відображення компонента AppBar

На сторінках, які повинні бути відображені лише гостям або лише авторизованим користувачам, виконується перевірка ролі. У разі неспівпадіння її з дозволеною, відбувається перенаправлення: `user ? <Redirect to="/"/>`.

3.2.4 Список тестів

Сторінка для перегляду каталогу тестів (рисунок 3.2.4.1) доступна всім користувачам (авторизованим і гостям сайту). На сторінці реалізований пошук по назві тесту.

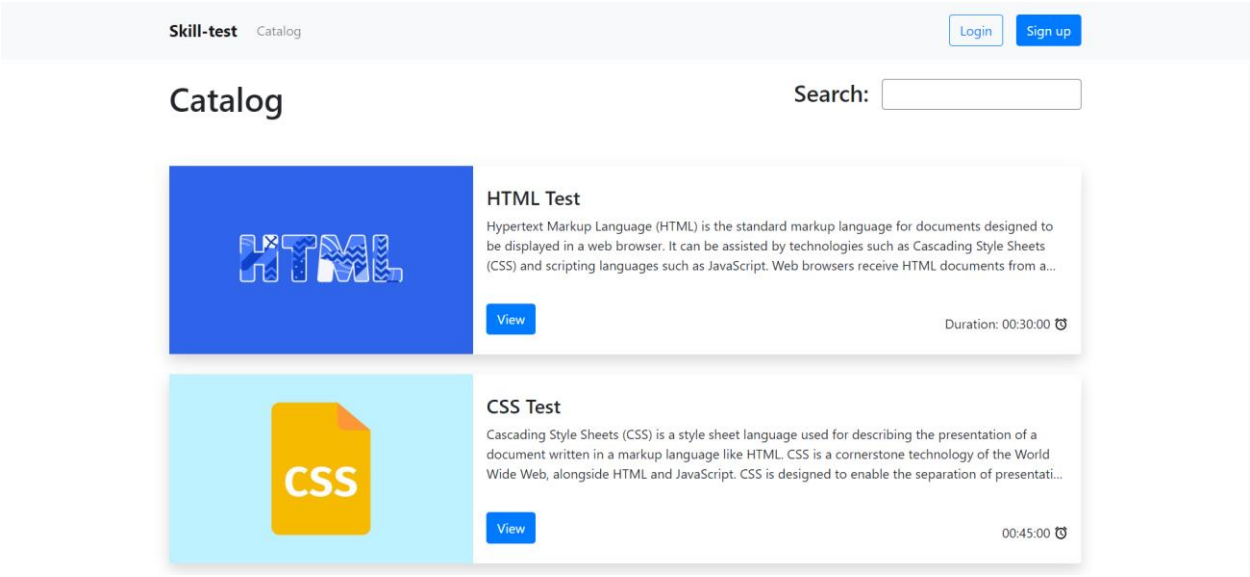

Рис. 3.2.4.1. Сторінка каталогу тестів

При завантаженні сторінки здійснюється запит на сервер для отримання списку тестів. Це відбувається у файлі testInfoActions.js, за отримання списку відповідає функція testInfoListFetch (рисунок 3.2.4.2).

```
export const testInfoListFetch = () => {
  return (dispatch, getState) => {
    dispatch(testInfoListLoading( bool: true));
    axios.get( url: `http://localhost:8000/skillful/tests-info/`) Promise<AxiosResponse<T>>
      .then((response : AxiosResponse<T> ) => {
        dispatch(testInfoListSet(response.data));
        dispatch(testInfoListLoading( bool: false));
      }) Promise<AxiosResponse<T>>
      .catch((error) => {
        if(error.response) {
          dispatch(testInfoListError(error.response.data.errorText));
        } else {
          dispatch(testInfoListError( errorMessage: "Something went wrong."));
        }
        dispatch(testInfoListLoading( bool: false));
      });
  }
};
```

Рис. 3.2.4.2. Функція testInfoListFetch

Якщо при завантаженні даних виникає помилка, викликається функція testInfoListError, яка дозволяє записати текст помилки у відповідну змінну redux сховища. Це відбувається для того, щоб повідомити користувача про помилку (рисунок 3.2.4.3). Інформування користувача про помилку на всіх сторінках здійснюється в такий спосіб.

Рис. 3.2.4.3. Повідомлення про помилку під час отримання списку тестів

3.2.5 Детальна інформація про тест

Сторінка для перегляду детальної інформації про тест доступна всім користувачам сайту (рисунок 3.2.5.1). Неавторизованим користувачам при натисканні на кнопку початку тесту буде запропоновано здійснити авторизацію або зареєструватися.

Рис. 3.2.5.1. Сторінка для перегляду детальної інформації про тест

3.2.6 Проходження тесту

Вся інформація про проходження тесту зберігається в redux сховищі. За це відповідає файл testReducer.js, в якому власне і визначені змінні для зберігання інформації про тест, всі питання, обрані користувачем відповіді, інформація про статус надсилання запиту і текст помилки, яка могла виникнути під час надсилання. У файлі testAction.js визначені дії, які використовуються під час проходження тесту.

Після підтвердження початку тесту, викликається функція testStart, яка надсилає на сервер запит з інформацією про початок (id користувача, id тесту). Сервер перевіряє доступність тесту для проходження користувачем (після минулої спроби повинна пройти достатня кількість

часу) та у відповідь надсилає питання тесту. Якщо сервер поверне помилку, її текст буде записаний у відповідну змінну у сховищі для подальшого відображення користувачу.

Після отримання питань та інформації про тест, здійснюється завантаження безпосередньо сторінки тесту (рисунок 3.2.6.1). Для зв‘язку компоненти React з redux сховищем використовується хук useSelector, імпортований з бібліотеки react-redux. Для виконання дій і зміни стану сховища, використовується хук useDispatch.

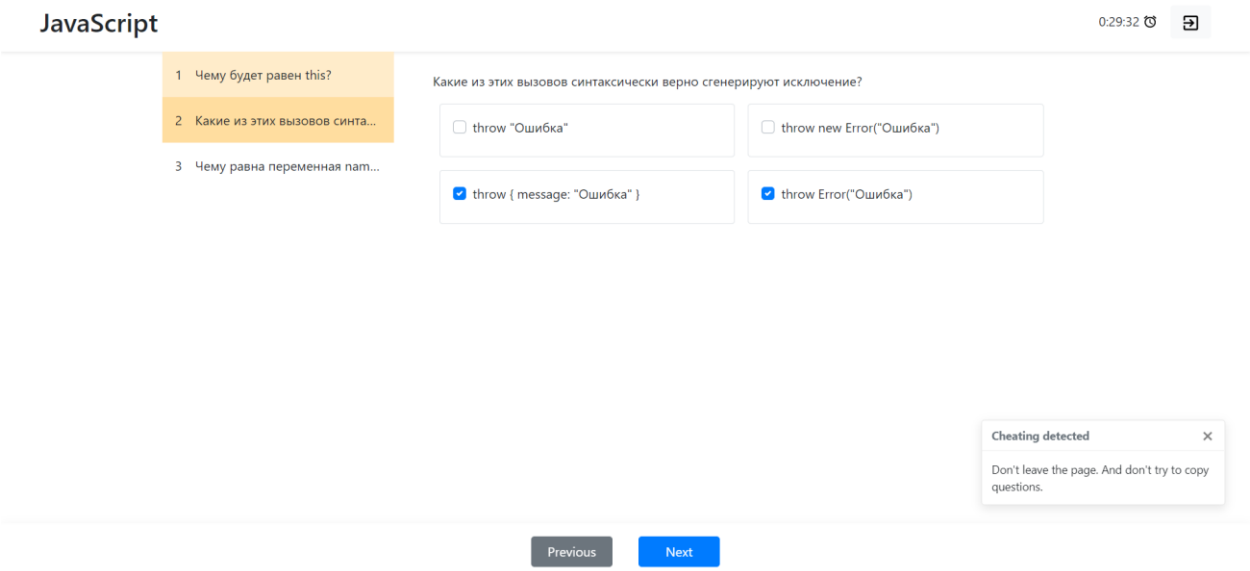

Рис. 3.2.6.1. Сторінка тесту

Об'єкт тесту, який повертає сервер у відповідь на запит про початок, містить масив питань тесту, в кожному з яких може бути код з визначеною мовою програмування. Для кращого читання цього коду виконується підсвічення синтаксису за допомогою бібліотеки Prism.js. Для цього в компонент, який відображає питання тексту, імпортується тема, додаткові мови програмування для відображення і сам об’єкт Prism, який містить метод highlightAll, який потрібно викликати для підсвічування синтаксису. Тож, для того щоб викликати цю функцію відразу після отримання коду компнонентом, використовується функція useEffect, в яку першим

аргументом потрібно передати виклик функції Prism.highlightAll(), а другим – масив об’єктів стану, до зміни значень яких і буде прив’язаний виклик функції підсвічування. Для відображення коду використовується тег pre з тегом code. Для коректної роботи Prism.js, тег code має містити клас, який буде відповідати потрібній мові програмування (рисунок 3.2.6.2).

```
<pre>
  <code className={"language-"+question.progLang}>
    {question.code}
  </code>
</pre>
```

Рис. 3.2.6.2. Фрагмент коду для налаштування Prism.js

Під час проходження тесту відстежується покидання сторінки, і копіювання тексту питання. Для того щоб відстежити покидання сторінки використовується подія blur, який встановлюється на елемент window при завантаженні компоненти тесту. Для відстеження копіювання тексту питання використовуються івенти onCopy та onCut, які є частиною SyntheticEvent (крос-браузерної оболонки для базових подій браузера).

Для запобігання випадкового закривання вкладки з тестом, використовується подія window beforeunload. Це дозволяє викликати вікно підтвердження закриття вкладки з відповідним повідомленням користувачу.

На початку тесту сервер надсилає час початку тесту, також в об’єкті тест зберігається інформація про тривалість тесту. В інтерфейсі реалізований таймер зворотного відліку. Після завершення часу відбувається автоматичне надсилання тесту на перевірку.

Якщо користувач вирішить покинути сторінку, він може натиснути кнопку завершення тесту, або закрити вкладку. При натисканні кнопки завершення, надсилається запит на сервер про скасування проходження тесту. При закриванні вкладки також буде надісланий запит про скасування тесту.

При натисканні кнопки надсилання тесту на оцінювання, користувачу показується модальне вікно для підтвердження надсилання. Якщо залишились питання, для якого не обрана відповідь, користувач також буде проінформований про це. Після підтвердження надсилання тесту на оцінювання, на сервер буде відправлений запит з об’єктом, який містить пари значень (id питання та id відповіді). Сервер перевіряє час надсилання відповіді та порівнює його з очікуваним часом прийняття відповіді і якщо все добре, здійснює перевірку тесту і надсилає користувачу результат.

3.2.7 Управління результатами тестів

Для авторизованих користувачів доступна сторінка перегляду власних результатів (рисунок 3.2.7.1). Користувач може переглядати власні результати, та змінювати їх статус публічності. При натисканні на перемикач публічності, на сервер надсилається запит для зміни статусу результату.

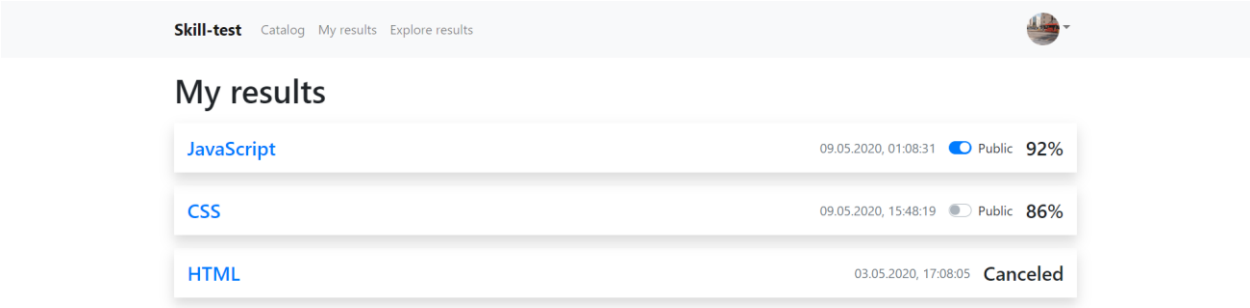

Рис. 3.2.7.1. Інтерфейс перегляду та управління результатами тестувань

3.2.8 Створення та редагування тесту

Сторінка для створення та редагування тесту (рисунок 3.2.8.1) доступна лише адміністраторам. Форма для створення тесту містить всі, необхідні для створення, поля (назва, опис, час між спробами, час на проходження). Також

форма дозволяє додавати запитання до тесту, а також додавати відповіді до запитання. Для реалізації цього функціоналу в компоненті створена змінна для зберігання масиву запитань і змінна для інформації для тесту. Для відображення списку тестів використовується функція "map". Для додавання нового питання у масиві питань створюється новий об'єкт з пустими полями. Додавання відповіді працює аналогічно.

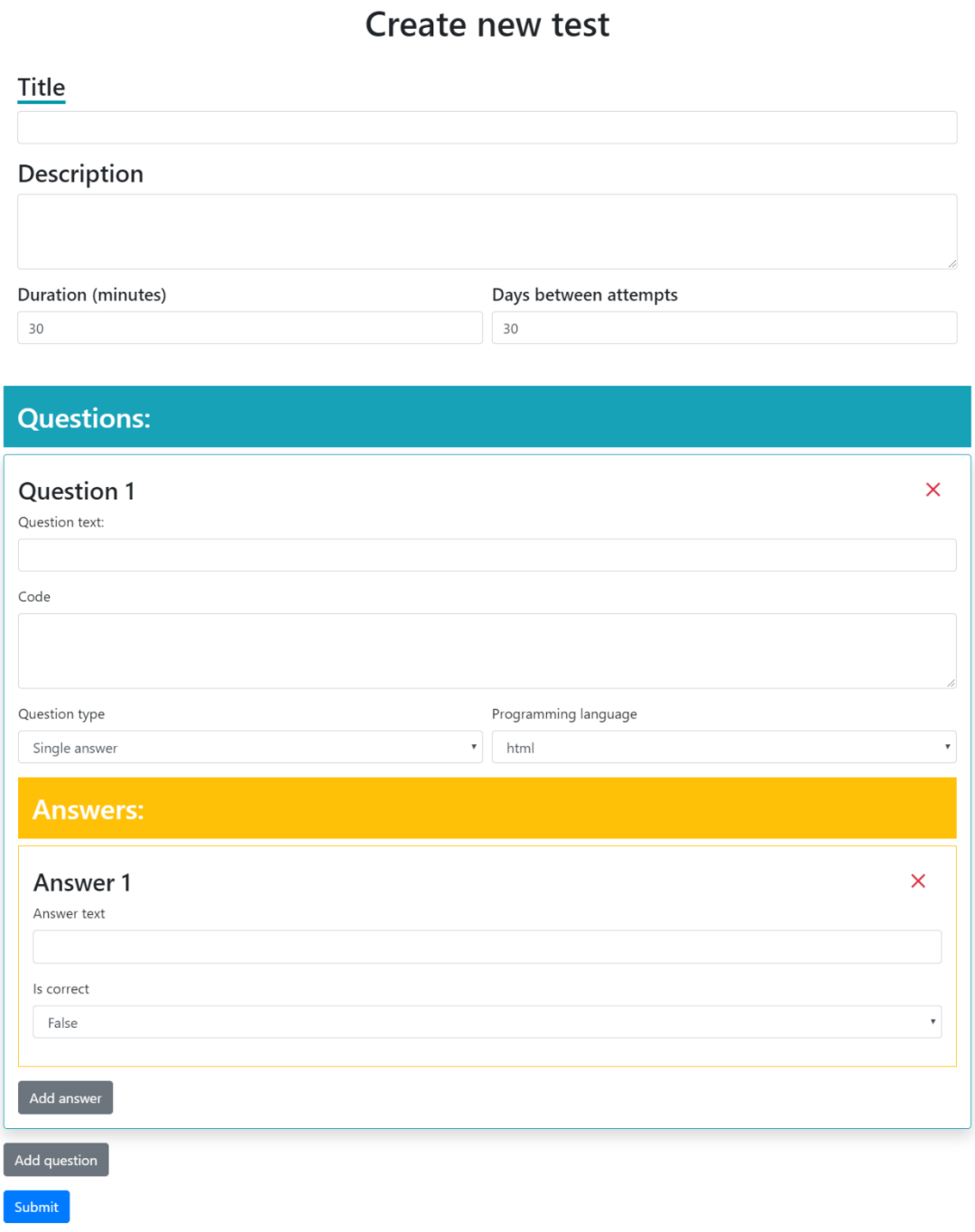

Рис. 3.2.8.1. Сторінка для створення та редагування тесту

Для редагування тесту використовується той же React компонент, але при завантаженні сторінки, поля форми автоматично заповнюються інформацією про тест.

Висновки

Під час виконання курсової роботи був проведений аналіз функцій представлених на ринку систем тестування знань. Також, були досліджені основні підходи до створення веб-застосунків за допомогою бібліотек React.js та Redux. При розробці системи було використано допоміжні інструменти та бібліотеки для додавання нових функцій та прискорення розробки платформи. В результаті виконання роботи було реалізовано інтерфейс користувача для системи тестування знань. Плануються допрацювання системи шляхом реалізації REST API.

Перспективами розвитку проекту є розширення списку типів завдань, зокрема завданнями з написання коду користувачем. Також, додавання ролі роботодавця та функціоналу для комунікації представників компаній та кандидатів на платформі дозволить гарантувати достатній рівень захисту особистих даних і контактів користувачів. Фінансовими перспективами розвитку проекту є показ рекомендацій щодо ресурсів для навчання або пропозицій від компаній на основі результатів тестування.

Список використаної літератури та електронних ресурсів

1. Документація бібліотеки React.js. [Електронний ресурс]. – Режим доступу: https://uk.reactjs.org/
2. Документація бібліотеки Redux. [Електронний ресурс]. – Режим доступу: https://redux.js.org/
3. Навчальні матеріали для роботи з react-router. [Електронний ресурс]. – Режим доступу: https://reacttraining.com/react-router/
4. Основи Redux. [Електронний ресурс]: codeguida.com – 2016. – Режим доступу: https://codeguida.com/post/624
5. Managing your React state with Redux. [Електронний ресурс]: medium.com – 2016. – Режим доступу: https://medium.com/the-web-tub/managing-your-react-state-with-redux-affab72de4b1
6. Документація бібліотеки React Bootstrap. [Електронний ресурс]. – Режим доступу: https://react-bootstrap.github.io/
7. Документація бібліотеки Formik. [Електронний ресурс]. – Режим доступу: https://jaredpalmer.com/formik/
8. Документація бібліотеки Styled Components. [Електронний ресурс]. – Режим доступу https://styled-components.com/docs
9. Документація бібліотеки Prismjs. [Електронний ресурс]. – Режим доступу https://prismjs.com/
10. React + Redux - JWT Authentication Tutorial & Example. [Електронний ресурс]: jasonwatmore.com – 2019. – Режим доступу: https://jasonwatmore.com/post/2017/12/07/react-redux-jwt-authentication-tutorial-example
11. Офіційна сторінка платформи «Google Форми». [Електронний ресурс]. – Режим доступу: https://www.google.com/intl/uk_ua/forms/about/
12. Офіційна сторінка платформи «Techgig». [Електронний ресурс]. – Режим доступу: https://www.techgig.com/home

13. Офіційна сторінка платформи «iSpring». [Електронний ресурс]. – Режим доступу: https://www.ispring.ru/
14. Офіційна сторінка платформи «Binary Studio Academy». [Електронний ресурс]. – Режим доступу: https://academy.binary-studio.com/ua/