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

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему:

«Інформаційно-аналітична система оцінювання комплексного рівня підготовки фахівців навчальним закладом»

Виконала: студентка 2-го курсу, групи 1КІ-20м
спеціальності 123 — Комп’ютерна інженерія
Степанова Т. М.
(прізвище та ініціали)

Керівник: к.т.н., проф. каф. ОТ
Захарченко С. М.
(прізвище та ініціали)
«____» _____ 2021 р.

Опонент: к.т.н., доцент каф. МБІС
Карпінець В. В.
(прізвище та ініціали)
«____» _____ 2021 р.

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф. Азаров О.Д.
(прізвище та ініціали)
«____» _____ 2021 р

Вінниця 2021

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

Освітньо-кваліфікаційний рівень — магістр

Спеціальність 123 — «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ

Азаров О. Д.

«____»

_____2021 р

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Степановаій Тетяні Миколаївні

1 Тема роботи: «Інформаційно-аналітична система оцінювання комплексного рівня підготовки фахівців навчальним закладом», керівник роботи: к.т.н., професор кафедри ОТ Захарченко Сергій Михайлович затверджені наказом Вінницького національного технічного університету від 24.09.2021 року № 277.

2 Строк подання студентом роботи 10.12.2021

3 Вихідні дані до роботи: існуючі системи-аналоги для оцінювання, методи тестування, технології для розробки.

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): огляд та аналіз існуючих систем оцінювання, проектування системи комплексного оцінювання, програмна реалізація системи.

5 Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень): макет інтерфейсу користувача, блок-схема алгоритму, лістинг програми.

6 Консультанти розділів роботи приведені в таблиці 1.

Таблиця 1 — Консультанти

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1,2,3	Захарченко С. М., к.т.н., професор каф. ОТ		
4	Кавецький В. В., к.т.н., доцент каф. ЕПВМ		

7 Дата видачі завдання _____

8 Календарний план представлений у таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів виконання магістерської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Постановка задачі роботи	04.10.2021	Виконано
2	Огляд і аналіз існуючих систем оцінювання	15.10.2021	Виконано
3	Огляд методики реалізації комплексного оцінювання знань студентів	29.10.2021	Виконано
4	Розгляд існуючих технологій для реалізації програмної частини	04.11.2021	Виконано
5	Вибір бази даних	17.11.2021	Виконано
6	Програмна реалізація частини системи.	21.11.2021	Виконано
7	Стилізація клієнтської частини	30.11.2021	Виконано
8	Оформлення пояснювальної записки	03.12.2021	Виконано
9	Перевірка якості виконання магістерської роботи	10.12.2021	Виконано

Студент _____Степанова Т. М._____

(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____Захарченко С. М.___

(підпис) (прізвище та ініціали)

Консультант з економічної частини _____Кавецький В. В.___

(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.4

Степанова Т. М. Інформаційно-аналітична система оцінювання комплексного рівня підготовки фахівців навчальним закладом. Магістерська робота зі спеціальності 123 — комп'ютерна інженерія, освітня програма — комп'ютерна інженерія. Вінниця: ВНТУ, 2021 ст.

В магістерській дипломній роботі було досліджено та проаналізовано існуючі системи оцінювання рівня підготовки, їх переваги та недоліки. У теоретичній частині розглянуто що таке оцінювання, види оцінювання та підходи. Розглянуто методику реалізації комплексного оцінювання знань. Також для програмної реалізації обрано технології, які є найновішими та найуживанішими на ринку. Розроблено частину інформаційно-аналітичної системи, в якій викладач та студент можуть виконувати базові операції. Також оглянуто та описано реалізовані функціональні можливості. Розроблено інструкцію користувача.

Інформаційно-аналітична система оцінювання комплексного рівня підготовки фахівців навчальним закладом розроблена для використання її у браузері на будь-якій операційній системі за допомогою середовища розробки WebStorm та стеку технологій MERN.

Ключові слова: інформаційно-аналітична система, оцінювання, тестування, технології розробки, JavaScript, React, Node.js, MongoDB.

ABSTRACT

UDC 004.4

Stepanova T. M. Information-analytical system for assessing the complex level of training of specialists in educational institutions. Master's thesis in specialty 123 — computer engineering, educational program — computer engineering. Vinnytsia: VNTU, 2021

In the master's thesis the existing systems of assessment of the level of training, their advantages and disadvantages were researched and analyzed. The theoretical part considers what is evaluation, types of evaluation and approaches. The method of realization of complex assessment of knowledge is considered. Technologies that are the latest and most widely used on the market have also been selected for software implementation. A part of the information-analytical system in which the teacher and the student can perform basic operations has been developed. Implemented functionality is also reviewed and described. User manual developed.

The information-analytical system for assessing the complex level of training of educational institutions is designed for use in a browser on any operating system using the WebStorm development environment and the MERN technology stack.

Keywords: information-analytical system, evaluation, testing, development technologies, JavaScript, React, Node.js, MongoDB.

ВСТУП

Актуальністю даного дослідження є підвищити ефективність освітньої системи. Перше, що згадується, коли мова йде про освіту — це здобуття знань в школі, і в подальшому в університеті. Освіта — це можливість. Можливість знати і захищати свої права та обов'язки, це можливість володіти необхідною інформацією. Освіта збагачує наш світогляд, надає нові можливості для розвитку, кращого життя, це можливість робити світ кращим. Бути освіченим означає не просто закінчити навчальний заклад, а здобути знання, які будуть корисними в житті. Освіта забезпечує швидкий процес розвитку людства.

З іншого боку сучасний рівень розвитку цифрового суспільства вимагає розробки інструментів, що дозволяють певним чином формалізувати процес оцінювання знань та навичок. Цей механізм реалізується як правило у вигляді тестів, або складних систем, що дозволяють перевіряти практичні навички, наприклад в пакеті Cisco Packet Tracer є можливість створювати спеціалізовані завдання, що зберігаються з розширенням .pka, і дозволяють контролювати вірність виконання практичних завдань в цій системі.

Слід звернути увагу, що сучасні системи оцінювання спрямовані виключно на оцінювання знань з певної теми або певної дисципліни, з іншого боку є проблема формалізованого оцінювання всього спектру знань і навичок. Використання для комплексного оцінювання виписки з диплома з десятками оцінок з різних несистематизованих дисциплін є занадто складним і неефективним процесом. З іншого боку середній бал теж не несе інформації про рівень знань, оскільки одна групу дисциплін студент може знати краще, а іншу гірше.

Система комплексного оцінювання може бути застосована як учасниками навчального процесу, так і потенційними споживачами випускників навчального закладу:

— навчально-педагогічним складом для визначення рівня підготовки студента за окремими напрямками дисциплін (програмний, апаратний, тощо);

— власне самими студентами для самооцінювання;

— потенційними працедавцями для визначення посад, на які доцільніше з таким багажем знань працевлаштовувати молодого спеціаліста.

Метою роботи є створення інформаційно-аналітичної системи комплексного оцінювання рівня підготовки фахівців навчальним закладом.

Відповідно до мети були поставлені такі завдання:

— провести аналіз існуючих систем оцінювання знань;

— розробити структуру системи комплексного оцінювання;

— обґрунтувати вибір технологій для реалізації системи;

— виконати програмну реалізацію системи;

— підготувати рекомендації для роботи з системою.

Об’єктом дослідження є процес автоматизації комплексного оцінювання рівня знань та практичних навичок суб’єкта навчального процесу.

Предметом дослідження є методи та програмні засоби комплексного оцінювання рівня знань та практичних навичок суб’єкта навчального процесу.

Наукова новизна магістерської роботи полягає у вдосконаленні системи оцінювання рівня знань та навичок студента за рахунок застосування комплексних показників, що враховують як прямі так і непрямі ознаки отриманих знань та практичних навичок, що дозволяє підвищити об’єктивність оцінювання.

Практичне значення отриманих результатів магістерської роботи полягає в створенні програмного забезпечення, що реалізує базові функції системи комплексного оцінювання.

Отримані результати магістерської кваліфікаційної роботи **апробовано** на науково-технічній конференції підрозділів Вінницького національного технічного університету «Молодь в науці: дослідження, проблеми, перспективи (МН-2022)».

1 ОГЛЯД І АНАЛІЗ ІСНУЮЧИХ СИСТЕМ ОЦІНЮВАННЯ

1.1 Загальне поняття про оцінювання

Оцінювання — це аналітичний інструмент або процедура, призначення якої є вимірювання прямих ефектів, результативності наданого матеріалу. В загальному розумінні оцінка — це систематичний процес і всестороннє вивчення конкретного предмета, системи, об'єкта, процесу, результат якого буде використовуватись для рекомендацій, покращення результату в майбутньому, для аналізу і підвищення ефективності.

В даний час, у зв'язку з актуалізацією завдання забезпечення якості вищої освіти, функція оцінювання починає набувати нового сенсу та розміщуватись в іншому контексті. Насамперед змінюється розуміння мети, яку обслуговує процедура оцінювання.

Сьогодні функція оцінювання не зводиться тільки до виявлення недоліків, а насамперед розглядається як критичний аналіз освітнього процесу, що передбачає насамперед точне визначення напрямів покращення. Важливо підкреслити, що йдеться не так про зміну засобів оцінювання, як про зміну цілей оцінювання.

Оцінювання — це не фіксація підсумків, а відправна точка, за якою слідує новий виток розвитку, а отже, і підвищення якості освіти. Іншими словами, головне завдання цієї процедури — поліпшення якості роботи конкретної людини і через це досягнення вищих цілей. Також поліпшення якості навчальних програм, в які залучені люди, що оцінюються, і досягнення нової якості роботи всієї організації загалом.

Таким чином, оцінювання починає інтерпретуватися як конструктивний зворотний зв'язок. Тому бачення результатів роботи дуже впливає на подальший розвиток та засвоєння матеріалу.

Існує декілька варіантів, чому аналіз і оцінка допомагають студенту:

— знаючи свої помилки, студент буде вчитись на них;

— оцінка допомагає зрозуміти студенту, що є важливим;

— оцінка показує студенту, що у нього виходить добре, а що ні;

— оцінка допомагає студенту виявити, що він/вона не знає;

— допомагає самостійно відслідковувати свій власний прогрес/регрес.

Оцінка часто ділиться на початкову, формуючу, та підсумкові категорії з метою розгляду різних версій.

Початкова оцінка — створена для використання її студентами відповідно до попередніх досягнень або особистих можливостей у найбільш підходящій навчальній стратегії або з підходящим вчителем. Вона проводиться за допомогою проміжного тестування, яке визначає ступінь готовності до навчання. Цей тип оцінки використовується, щоб дізнатися, який рівень знань студента з предмета. Це допомагає вчителю ефективніше пояснювати матеріал. Ці оцінки не виставляються за шкалою.

Формуюча оцінка — ця оцінка зазвичай проводиться протягом усього курсу або проекту, використовується для допомоги у навчанні. В освітньому середовищі формуюче оцінювання може бути проведене викладачем і студентом. Метою є отримати зворотний зв'язок про роботу, не обов'язково використовується для виставлення оцінок.

Формуючі оцінки можуть набувати форми діагностичних, стандартних тестів, вікторин, усних питань або чернеток. Формують оцінки одночасно з інструкціями. Результат можна зарахувати. Формуюче оцінювання націлено на те, щоб побачити, чи студенти розуміють інструкцію, перед тим як виконувати підсумкове оцінювання.

Сумативне оцінювання — зазвичай проводиться наприкінці курсу чи проекту. В освітньому середовищі підсумкові оцінки зазвичай застосовуються для присвоєння студентам оцінок за курс. Підсумкові оцінки призначені для узагальнення того, що дізналися студенти, щоб визначити, чи вони розуміють предмет. Цей тип оцінки зазвичай виставляється за шкалою від 0 до 100. Можуть набувати форми тестів, іспитів або проектів.

Сумативне оцінювання часто використовується для визначення, здав студент залік чи ні. Таким чином, визначається рівень засвоєння матеріалу [1].

Діагностична — така оцінка має справу з усіма труднощами. Ця оцінка проводиться в кінці навчального процесу.

Оцінка (підсумкова чи формуюча) часто поділяється на об'єктивну чи суб'єктивну. Об'єктивна оцінка — це форма питань, які є єдиною правильною відповіддю. Суб'єктивна оцінка — це форма питань, яка може бути зазначена більше одного виразу правильної відповіді (або більше одного виразу правильної відповіді).

Є різні типи об'єктивних та суб'єктивних питань. Типи об'єктивних питань включають справжні або помилкові відповіді, множинний вибір, множинні відповіді та питання на відповідність. Суб'єктивні питання включають питання з розширеною відповіддю та есе. Об'єктивне оцінювання добре підходить для все більш популярного комп'ютеризованого формату або формату онлайн-оцінювання. Різниця між об'єктивними та суб'єктивними оцінками не є ні корисною, ні точною, тому що насправді не існує таких речей, як об'єктивна оцінка [2].

Тест результатів можна порівняти зі встановленим критерієм, або з успішністю інших студентів, або з попередньою успішністю. Існує кілька видів оцінки: оцінка, заснована на критеріях, оцінка, заснована на нормах, іпсативна оцінка. Розглянемо кожний вид детальніше.

Оцінка, заснована на критеріях проводиться зазвичай з використанням тесту, заснованого на критеріях. При такому оцінюванні кандидати оцінюються за певними об'єктивними критеріями. Критеріальне оцінювання часто, але не завжди, використовується для визначення компетентності людини, визначається чи може вона щось робити. Найвідомішим прикладом оцінки, заснованим на критеріях є іспит з водіння, коли студенти оцінюються за рядом явних критеріїв.

Оцінка, заснована на нормах, зазвичай з використанням еталонного тесту не вимірюється за певними критеріями. Цей тип відноситься до студентської спільноти, яка проводить оцінку. Це є ефективним способом порівняння студентів. Тест IQ — це найвідоміший приклад оцінки, що

базується на нормах. Багато вступних випробувань засновані на нормах, що дозволяє пройти фіксовану частку студентів. Це означає, що стандарти можуть змінюватися рік у рік залежно від якості когорти; оцінка, заснована на критеріях, не змінюється з року в рік.

Іпсативна оцінка — це самостійне порівняння або в той же час, або в порівнянні з іншими областями в рамках того ж студента.

Оцінка може бути формальною або неформальною. Формальна має на увазі письмовий документ, такий як тест, вікторина чи робота. Формальній оцінці надається числовий бал або оцінка, заснована на успішності студента. Тоді як неформальна оцінка не впливає на підсумкову оцінку студента. Неформальна оцінка зазвичай відбувається у більш невимушеній манері. Це може бути інвентаризація, контрольні списки, шкали оцінок, рубрики, оцінки ефективності та портфоліо, участь у колегіумі, самооцінка, а також обговорення [3].

Моніторинг — це поняття є близьке до оцінювання, про те це трохи інше. Моніторинг напряму пов’язаний з тим, чи буде оцінювання ефективним. Поняття моніторингу стало вживаним в різних сферах людської діяльності.

Моніторинг — це система постійного спостереження за явищами і процесами, що проходять в навколишньому середовищі і суспільстві, результати якого служать для обґрунтування рішень, які будуть прийняті до об’єкту моніторинга.

Вперше моніторинг був використаний в грунтознавстві, потім в екології та інших суміжних науках. В даний час він вивчається і використовується в технічних, в соціальних науках, і в різних сферах практичної діяльності. Є підстави говорити, що залишилося мало областей діяльності, де в тій чи іншій мірі не використовувався б моніторинг. Тому при оцінюванні студента потрібно завжди моніторити ситуацію із засвоєними заннями. Метою моніторингу в системі освіти є постійний контроль, що несе виховну, освітню, розвиваючу функції [4].

В процесі оцінювання дуже важливо, щоб студент умів самостійно, зі здоровим глуздом, оцінювати свої знання, умів співвідносити їх до поставлених вимог. Поєднання самоконтролю з боку студента і контролю з боку викладача значно підсилить ефект заходів оцінювання.

На сьогодні існує велике різноманіття форм контролю якості навчання та ступінь засвоєного матеріалу. Найпоширенішими методами контролю є: усний контроль, письмовий, тестовий, графічний, програмований контроль, практична перевірка, а також методи самоконтролю і самооцінки.

Усний контроль (усне опитування) — це найпоширеніший метод у навчальній практиці. Його використання сприяє опануванню логічного мисленням, виробленню і розвитку навичок аргументування, висловлювлення своєї думки грамотно, образно, емоційно, обстоювати власну думку. Здійснюють його на семінарських, практичних і лабораторних заняттях, а також колоквіумах, лекціях і консультаціях.

Метою письмового контролю є з'ясування в письмовій формі ступеня оволодіння студентами знаннями, вміннями та навичками з предмета, визначення їх якості, правильності, точності, усвідомленості, вміння застосувати знання на практиці.

Тестовий контроль застосовується для визначення рівня сформованості знань і вмінь з навчальної дисципліни користуються методом тестів. Виокремлюють тести відкритої форми (із вільно конструйованими відповідями) і тести закритої форми (із запропонованими відповідями).

Тести відкритої форми передбачають короткі однозначні відповіді, які ґрунтуються переважно на відтворенні вивченого матеріалу, або складні (комплексні) відповіді, які потребують розвинутого логічного мислення, вміння аналізувати.

Тести закритої форми передбачають вибір відповіді з певної кількості варіантів. Серед таких тестів виокремлюють тест-альтернативу, тест-відповідність.

Тест-альтернатива вимагає вибору однієї з двох запропонованих відповідей. Застосовують його під час контролю таких показників засвоєння, як уміння визначати використання фактів, законів, підводити під поняття, встановлювати причину якогось явища. Недолік цього виду тесту полягає в тому, що він не дає змоги вільно конструювати, формулювати відповідь. Його перевагою є те, що він допомагає швидше орієнтуватися в матеріалі, знаходити спільне та відмінне у явищах, легше класифікувати конкретні явища за певними видами.

Тест-відповідність, як правило, складається з двох частин, між якими слід встановити відповідність. Застосовують його для виявлення таких результатів засвоєння, як уміння визначати використання речовин, апаратів, процесів, встановлювати зв'язок між абстрактним і конкретним поняттями, класифікувати їх тощо. Перевага тестів-відповідностей полягає у компактній формі завдання, завдяки якій протягом короткого часу вдається перевірити засвоєння великого обсягу навчального матеріалу. Недоліком є обмеженість безпосередньої мети контролю і ускладнення при доборі матеріалу.

Сутність графічного контролю полягає у створенні студентом узагальненої наочної моделі, яка відображає відношення, взаємозв'язки певних об'єктів або їх сукупності. Наочна модель — це графічне зображення умови задачі, креслення, діаграми, схеми, таблиці (наприклад, схема історичної битви, нанесені на контурні карти певні географічні та історичні об'єкти). Графічна перевірка може бути самостійним методом контролю або органічним елементом усної чи письмової перевірки.

Практичну перевірку застосовують з навчальних дисциплін, які передбачають оволодіння системою практичних професійних умінь та навичок, і здійснюють під час проведення практичних і лабораторних занять з цих навчальних дисциплін, у процесі проходження різних видів виробничої практики. Така перевірка дає змогу виявити, якою мірою студент усвідомив теоретичні основи цих дій.

Суттю методу самоконтролю є усвідомлене регулювання студентом своєї діяльності задля забезпечення таких її результатів, які б відповідали поставленим завданням, вимогам, нормам, правилам, зразкам. Мета самоконтролю — запобігання помилкам і виправлення їх. Показником сформованості самоконтролю є усвідомлення студентом правильності плану діяльності та її операційного складу, тобто способу реалізації цього плану.

Метод самооцінки передбачає критичне ставлення студента до своїх здібностей і можливостей, об'єктивне оцінювання досягнутих результатів [5].

Всі ці методи контролю можна проводити як усно, так і в письмовій формі чи використовуючи машинні засоби, такі як комп'ютер.

Використання машинних засобів дозволяє швидше та якісніше здійснювати контроль за діяльністю студентів. Спеціально створені програми дозволяють виконати перевірку, провести моніторинг та скласти результат у зручний презентабельний вигляд для викладача та студента. Також може бути підключена аналітика, яка буде моніторити всі отримані результати для повноти бачення ситуації.

1.2 Аналіз існуючих систем оцінювання якості знань

Розглядаючи наявні програмні засоби для проведення тестового контролю, необхідно зазначити, що будь-який програмний засіб, що використовується у навчальному процесі, повинен відповідати загальним вимогам педагогічних програмних засобів, зокрема: інтерфейс програми повинен бути виконаний рідною мовою студента; програмне забезпечення повинно бути ліцензійним, тобто законно придбаним.

Враховуючи специфіку тестового контролю, програмні засоби мають задовольняти такі вимоги:

— можливість використання кількох типів питань; можливість створення питань і відповідей, що можуть містити формули, малюнки, схеми;

— можливість вибору наступного питання випадковим чином з наявної сукупності тестових завдань;

— відображення варіантів відповідей у випадковому порядку для кожного тестуючого;

— збереження результатів тестування після завершення виконання тесту;

— збереження усіх відповідей для забезпечення зворотного зв’язку із тестуючим;

— можливість проведення аналізу тестових завдань, загалом усього тесту й аналізу відповідей кожного тестуючого зокрема;

— можливість експорту результатів тестування в інші програмні засоби для більш детального аналізу результатів тестування.

Як правило, сучасні системи комп’ютерного тестування складаються з кількох функціональних модулів, які можуть бути об’єднані в єдине ціле і інсталюватись на комп’ютери або існувати окремо у вигляді виконуваних файлів. Найчастіше до складу стандартної системи тестування входять:

— редактор тестів (модуль, призначений для створення тестів);

— модуль тестування;

— модуль для обробки результатів тестування;

— довідкова система;

— модуль, за допомогою якого можна здійснювати мережеве тестування.

На сьогоднішній день існує чимало програм для тестування знань студентів. Всі вони мають свої переваги та недоліки.

1.2.1 easyQuizzy

Програма дозволяє легко створювати тести в форматі виконуваних файлів, для запуску яких не буде потрібно використання додаткового програмного забезпечення.

За допомогою програми можна додавати в тест питання трьох різних типів: з єдиним правильним відповіддю із запропонованих варіантів, з

кількома вірними відповідями і самостійним введенням відповіді. Всі питання можуть включати в себе графічні файли, спеціальні символи і формули.

Малого того, автор тесту вільний налаштувати демонстрацію докладної статистики про хід проходження тесту по його закінченні і додати обмеження за часом [6]. Програма дозволяє захищати готові файли від редагування за допомогою пароля і має зручний інтерфейс, який показано на рисунку 1.1.

1.2.2 MyTest

Перша з таких програм MyTest це — система програм (програма тестування учнів, редактор тестів і журнал результатів) для створення і проведення комп'ютерного тестування, збору і аналізу результатів, виставляння оцінки за вказаною в тесті шкалою [7].

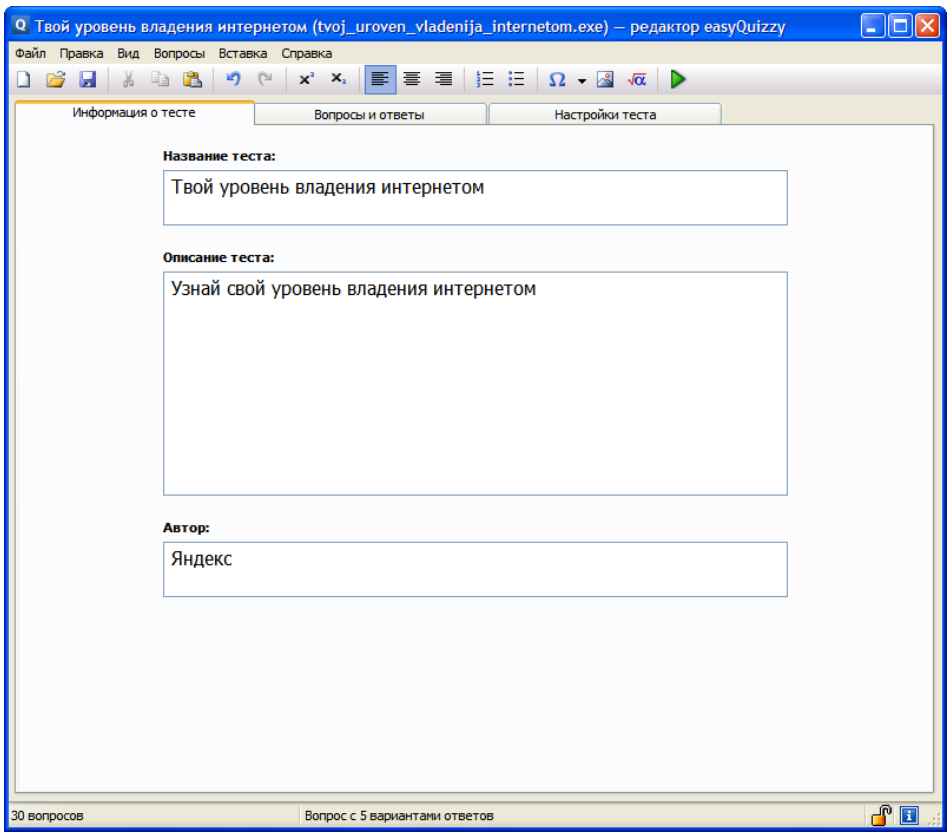


Рисунок 1.1 — Редактор тестів програми easyQuizzy

За допомогою пакету можна легко створювати тести з будь-яких предметів шкільної програми, з будь-яких вузівських дисциплін, тести для

професійного тестування, психологічні тести. Інтерфейс показано на рисунку 1.2.

Програма MyTest працює з вісьма типами завдань: одиночний вибір, множинний вибір, встановлення порядку проходження, встановлення відповідності, вказівка істинності або помилковості тверджень, ручне введення числа, ручне введення тексту, вибір місця на зображенні.

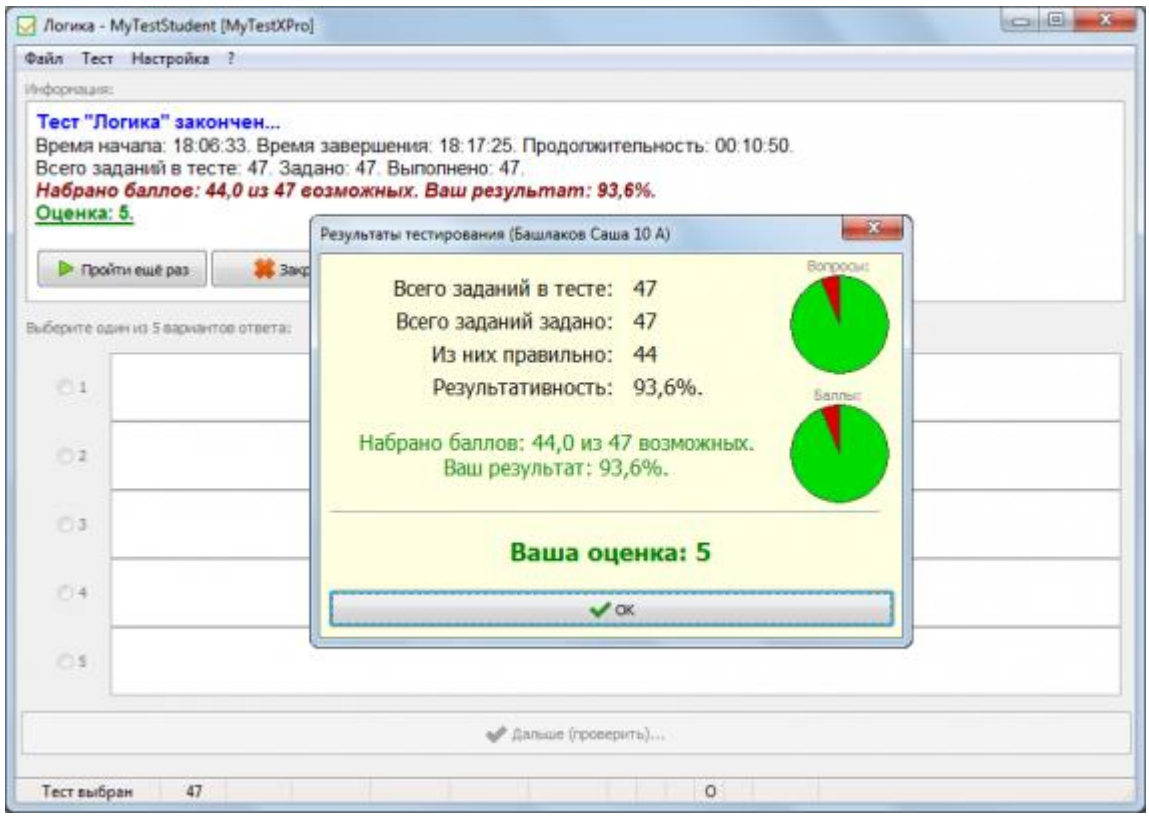


Рисунок 1.2 — Вікно програми MyTest

1.2.3 MyTestXPro

MyTestXPro — це система програм для створення і проведення комп'ютерного тестування знань, збору і аналізу результатів.

За допомогою програми MyTestXPro можлива організація і проведення тестування, іспитів в будь-яких освітніх установах як з метою виявити рівень знань по будь-яким навчальним дисциплінам, так і з навчальними цілями. Підприємства та організації можуть здійснювати атестацію та сертифікацію своїх співробітників, це зображено на рисунку 1.3.

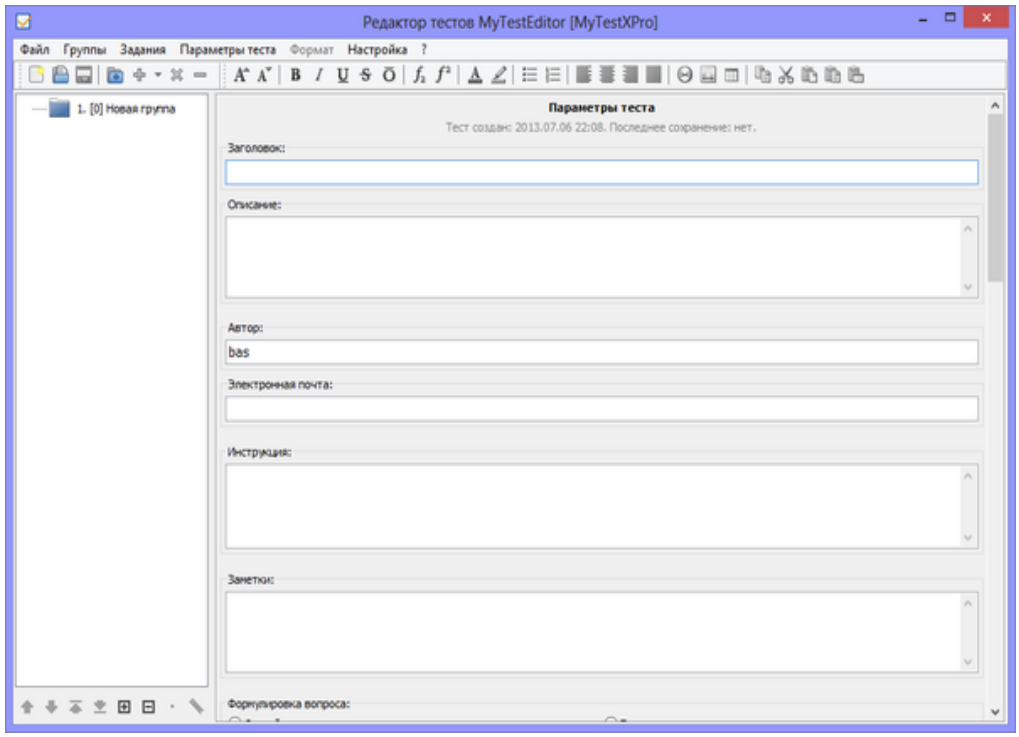


Рисунок 1.3 — Редактор тестів програми MyTestXPro

Програма MyTestXPro працює з десятьма різними типами завдань. У тесті можна використовувати як завдання одного типу, так і завдання різних типів. Кількість груп і завдань у тесті не обмежена. Питання з варіантами відповіді можуть включати до десяти варіантів. Для кожного завдання можна задати до п'яти формулювань питання.

Текст питання і варіантів відповіді (там, де вони можливі) підтримують можливості форматування тексту, вставки малюнків, таблиць, символів. У програмі є зручний вбудований текстовий редактор. Форматувати текст, вставляти таблиці, малюнки і символи можна не тільки в питання, але і в варіанти відповідей.

Програма підтримує декілька незалежних один від одного режимів тестування. Використовуючи різні режими і параметри тестування, можливо ефективно вирішувати різноманітні завдання, як навчання, так і перевірки знань [8].

При неможливості провести комп'ютерне тестування з електронного тесту можна швидко сформувати і роздрукувати «паперовий тест». Для

зручності поширення тестів серед студентів можна створювати «автономні тести» — програми, що містять один тест і настройки модуля тестування в одному виконуваному exe-файлі.

1.2.4 INDIGO

Система тестування INDIGO є багатофункціональним комплексом програмного забезпечення, що дозволяє автоматизувати процес проведення тестування і обробки результатів. Система INDIGO є універсальним інструментом, який можна використовувати для вирішення широкого спектра завдань: визначення рівня готовності учнів шкіл до ДПА та ЗНО, тестування і контроль знань студентів з різних дисциплін, визначення професійного рівня співробітників (в тому числі при прийомі на роботу), автоматизація психологічних тестів (в тому числі профорієнтаційних), проведення опитувань (соціологічних, маркетингових, виявлення домінуючої точки зору і т.д.), автоматизація проведення вікторин та олімпіад, зображено на рисунку 1.4.

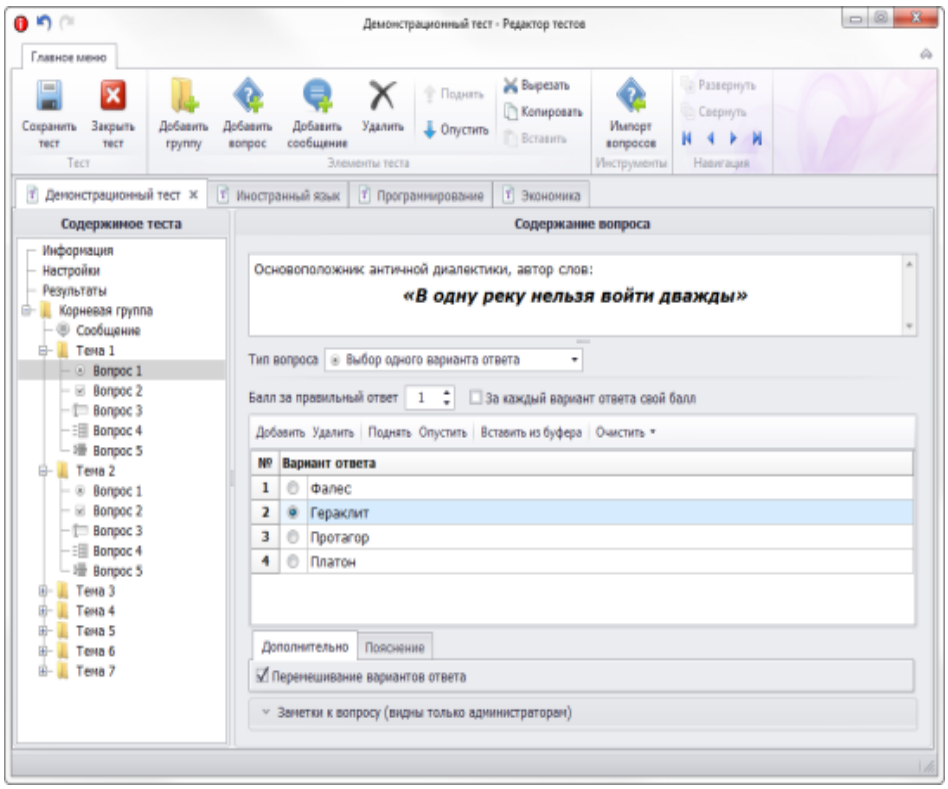


Рисунок 1.4 — Редактор тестів програми INDIGO

Система тестування встановлюється на один комп'ютер-сервер за допомогою інсталяційного пакета. З можливістю працювати як на ізольованому комп'ютері, так і в локальній мережі або через Інтернет. Всі дані зберігаються централізовано в базі даних системи, адміністратори працюють через програму клієнт [9].

Користувачі працюють через web-браузери (Google Chrome, Mozilla Firefox, Opera, Internet Explorer, Safari та інші). Є підтримка браузерів на мобільних пристроях.

1.2.5 Let's test

Система тестування Let's test дозволяє проводити онлайн тестування знань через інтернет. Вона є не просто конструктором тестів, а володіє широким набором функціональних можливостей, завдяки яким можна побудувати цілу інфраструктуру тестів [10]. Let's test дозволяє побудувати систему перевірки знань за допомогою тестів, витративши при цьому мінімум зусиль. Систему тестування наведено на рисунку 1.5.

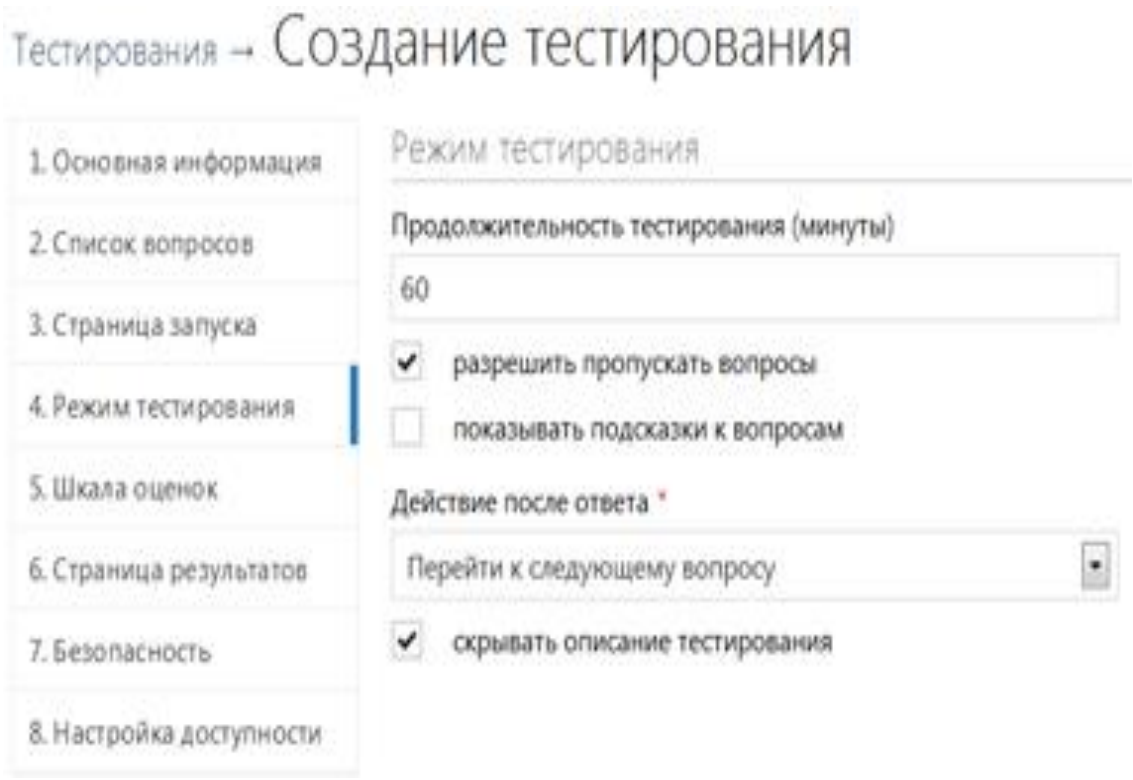


Рисунок 1.5 — Редактор тестів сервісу Let's test

1.2.6 Порівняння програм аналогів

Таблиця 1.1 — Порівняння програм для тестування знань

	easyQuizzy	MyTest	MyTestXPro	INDIGO	Let's test
Редактор тестів	+	+	+	+	+
Програма-сервер	-	-	+	+	+
Програма-клієнт	+	+	+	-	-
WEB-Інтерфейс	-	-	-	+	+
Система збору та збереження статистики	-	-	+	+	+
Налаштування шкали оцінювання	+	-	+/-	+	+
Шифрування даних	+/-	+/-	+	+	+
Імпорт питань	-	-	+/-	+	-

1.3 Загальні вимоги

Приведемо перелік загальних вимог, що описують структуру майбутньої системи та спосіб взаємодії між її елементами:

— система повинна бути побудована на клієнт-серверні технології;

— необхідно реалізувати можливість централізованої роздачі тестових запитань із одного комп’ютера на решту, не залежно від їх кількості;

— система повина містити сертифікати, дипломи по різних спеціальностях;

— система повина містити результати з навчання по всіх дисципліах;

— повинна виставляти коефіцієнти різним предеметам;

— проводити моніторинг.

Вимоги до бази та редактора тестів доповнюють загальні вимоги до системи. Так, наприклад, якщо система має забезпечувати шифрування файлу із запитаннями, то очевидно, що це мусить бути реалізоване саме в редакторі.

Крім того редактор повинен відповідати наступним вимогам:

— зручна систематизація запитань по темах;

— можливість копіювати та переміщати питання між темами;

— можливість відкривати та зберігати загальну статистику проходження тестів;

— система підказок та моніторингу помилок при створенні тестів.

Слід зазначити, що сервер мусить працювати по власному прикладному протоколу (поверх рівня TCP протокольного стека TCP/IP), який також повинен розуміти і клієнт, це допоможе створити між прикладними процесами сервера і клієнта буде надійний канал передачі даних. Окрім цього, до серверної частини системи висуваються наступні вимоги:

— повинен забезпечувати можливість одночасного проходження кількох тестів;

— можливість задавати перелік студентів та кількість спроб проходження;

— система відображення статистики на поточному етапі тестування;

— можливість відключення скомпрометованих питань;

— забезпечення автоматичного знаходження серверу клієнтом в мережі.

Клієнтську частину потрібно зробити максимально простою та зрозумілою. Крім того слід відзначити, що клієнтська частина, в цілях безпеки, не отримує вірну відповідь, перевірка відбувається на сервері.

Крім того необхідно реалізувати наступні можливості:

— масштабування вкладених ілюстрацій;

— відображення кількості правильних та неправильних відповідей;

— відображення кількості часу, що залишився до завершення тестування.

2. ПРОЕКТУВАННЯ СИСТЕМИ КОМПЛЕКСНОГО ОЦІНЮВАННЯ

2.1 Обгрунтування доцільності комплексного оцінювання

ВНЗ обіцяють здобувачам знань висококваліфіковану підготовку та отримання якісних знань. Проте не завжди є можливість це забезпечити, оскільки університети не аналізують здібності студентів, та не проводять комплексне оцінювання при вступі та по завершенні навчання. Тому процес вибору ІТ напряму фактично може тривати протягом всього навчання в університеті. Також, після закінчення навчання не завжди зрозуміло, в якому напрямку студент розуміється краще. Це питання є актуальним як для самого навчального закладу з точки зору самоаналізу, так і для майбутніх працедавців.

Зараз існує доволі багато ресурсів для знань, якими користуються студенти під час навчання. Деякі з них дають поради для подальшого вивчення, деякі дають сертифікати і дипломи, що затверджують отриманні знання, деякі нічого не дають.

Головною ідеєю інформаційно-аналітичної система комплексного оцінювання рівня підготовки фахівців навчальним закладом є зібрати всі досягнення, оцінки і результати студента для формування більш інформативного показника, або системи показників, що буде характеризувати рівень отриманих знань та навичок. Також ця інформація буде корисною для студента і допоможе йому визначитись в якому напрямку потрібно розвиватись, чому варто приділити більше уваги і зусиль.

Якщо брати всі роки навчання в університеті, то на будь-якій спеціальності студенти вивчають доволі багато дисциплін з різних напрямків. Для прикладу, візьмемо спеціальні дисципліни, що вивчають студенти напряму підготовки 123 Комп’ютерна інженерія. Умовно всі спеціальні дисципліни можна поділити за напрямами, як показано на рисунку 2.1.

Наприклад, до дисципліни, що спрямовані на формування математичного базису можна віднести:

— алгоритми та методи обчислень;

— дискретна математика;

— комп’ютерна логіка.

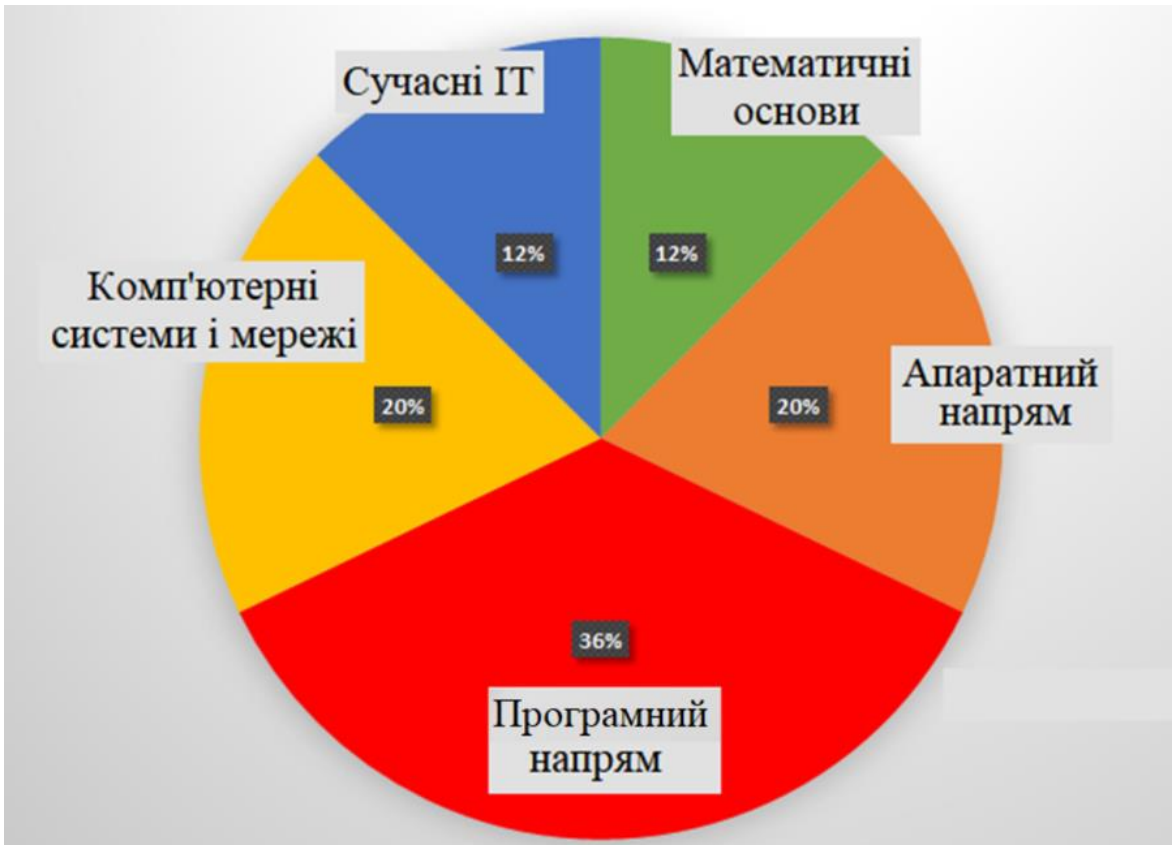


Рисунок 2.1 — Розподіл спеціальних дисциплін спеціальності 123 за ІТ напрямами

Дисципліни прогрмамного напряму:

— об’єктно-оієнтоване програмування;

— паралельні та розподілені обчислення;

— системне програмування;

— програмування інтернет-додатків;

— проектування додатків мобільних пристроїв;

— візуальне програмування;

— операційні системи.

Апаратного напряму:

— архітеткура комп’ютерів;

— аналого-цифрова техніка;

— комп’ютерна електроніка;

— комп’ютерна схемотехніка;

— периферійні пристрої комп’ютерів.

Напряму сучасних IT технологій:

— тестування та верифікація програмного забезпечення;

— комп’ютерна графіка;

— комп’ютерна обробка мультимедійних даних;

— хмарні технології;

— web-технології та web-дизайн.

Напряму комп’ютерних систем і мереж:

— комп’ютерні системи;

— корпоративні і загальнодоступні мережі;

— основи мультимедіа і безпеки в комп’ютерних мережах, основи інтернету речей.

Звичайний набір оцінок по всіх дисциплінах, розкиданий по різних журналах чи просто збережений в одному місці, не дає можливості формалізувати процес узагальненого оцінювання. Якщо просто брати середній бал студента, то це також не завжди точний і коректний показник, оскільки у студента можуть бути високі оцінки за дисципліни програмного напрямку і низькі оцінки за дисципліни апаратного напрямку або ж навпаки. З іншого боку не враховуються додаткові досягнення студента. Тому винникла ідея узагальнити результати навчання шляхом розрахунку середнього балу по кожній з груп дисциплін, а крім того врахувати додаткові фахові досягнення.

Наприклад, комплексний показник може враховувати такі показники:

— набір середніх оцінок за напрямами;

— наявність дипломів і сертифікатів (перемоги в професійних конкурсах і олімпіадах, сертифікати про проходження додаткових курсів за фахом, індустріальні сертифікати про отримання певної кваліфікації);

— професійна активність (досвід роботи за спеціальністю);

— інноваційна активність (патенти, свідоцтва про авторський твір);

— наукова активність (публікації в спеціальних виданнях).

Слід звернути увагу, що перелік показників не є сталим і може змінюватись, оскільки певні показники можуть виявитись неефективними, їх доцільно буде вилучити, щоб не ускладнювати систему показників і не робити її занадто громіздкою.

2.2 Методика реалізації комплексного оцінювання знань студентів

Як згадано вище, комплексний показник може містити багато компонентів. Виходячи з цього, комплексне оцінювання може бути реалізовоно наступним чином. Як було показано вище, крім власне оцінок, отриманих в процесі навчання є різні джерела інформації про рівень підготовки фахівця, наприклад:

— дипломи і нагороди, сертифікати про навчання на навчальних ресурсах;

— професійні сертифікати про рівень знань, наприклад CCNA в комп’ютерних мережах;

— перемоги в різноманітних олімпіадах і конкурсах професійного напрямку;

— відзиви працедавців;

— досвід роботи.

Завданням системи є зібрати всі ці показники і вивести для студента загальну картину, щоби мати уявлення в якому напрямку краще розвиватись. Щоб ставити цілі перед студентом, які він буде досягати, для отримання результату. А викладач на основі цих даних буде аналізувати рівень засвоєння, відповідно до цього буде готувати матеріал для подачі. Тобто на вхід системи

будуть подаватись різні дані, окрім того необхідно курувати вагою кожної складової цієї системи. Це керування може бути реалізоване через механізм метрик або вагових коефіцієнтів. Таким чином, комплексна оцінка може бути розрахована як:

$$M_{compl} = \frac{\sum_1^n M_{j*}\mu_i}{n},$$

де M_i та μ_i — значення і-го та j-го показникників та їх ваговий коефіцієнт.

Кількість показників та їх вагових коефіцієнтів залежить від позиції або оцінки, на яку претендує студент.

Наприклад, для позиції програмного напрямку більш впливовими будуть показники дисциплін програмного напрямку та відповідні сертифікати та інші досягнення. Крім того доцільно враховувати напрям тематики дипломного проектування та отримані оцінкі за відподні роботи та проекти.

В основі такої коплексної системи буде лежати тестова система, яка буде пропонувати проходження тестів з різних напрямків. Показником ефективності для кожного тестування буде дискретне значення, яке буде описувати оцінку.

Від кожного типу тестування буде визначатись інтегрована оцінка [11]. Діапазон представлений у таблиці 2.1

Таблиця 2.1 — Інтегрована оцінка

Інтегрована оцінка ефективності E_0	Визначення ефективності
1,0-0,7	високе
0,7-0,5	добре
0,5-0,4	задовільне
0,4-0,2	низьке
0,1-0,2	незадовільне

Також у ході проведення занять, шляхом опитування потрібно також визначити і інші фактори, що впливають на ефективність та досягнення поставленої цілі. Ці фактори наведені в таблиці 2.2.

Таблиця 2.2 — Фактори ефективності

№	Фактори
1	Оцінка рівня зацікавленості заняття
2	Оцінка рівень актуальності заняття
3	Оцінка рівня актуальності занняття
4	Оцінка рівня практичної користі заняття
5	Оцінка загального рівня заняття
6	Чи сподобалось заняття студенту
7	Чи сподобалась методика проведення заняття

Сума цих всіх показників і буде визначати для студента рівень успішності засвоєного матеріалу, а для викладача рівень успішності проведеного заняття.

Структура системи буде досить проста. У кожного студента буде особистий кабінет, в якому буде зберігати уся інформація про досягення. Викладачу буде дотсупний весь список студентів. Блок схема роботи викладача з системою наведена у додатку Б. Блок схема роботи студента з системою наведена у додатку В.

2.3 Вибір технологій для додатків

Так як в основому вся навчальна робота проходить використовуючи ноутбук, тому було обрано робити інформаційно-аналітичну систему для десктопу у вигляді веб додатка в браузері.

Будь-яка веб-програма створюється за допомогою декількох технологій. Стеком називається поєднання цих технологій. SPA — це парадигма для веб-додатків, яка уникає оновлення сторінки для відображення нового контенту.

Замість цього додаток використовує легкі запити на сервер для отримання деяких даних або фрагментів. Потім оновлює веб-сторінку. Результат виглядає досить витонченим у порівнянні із старим способом, коли потрібно було повністю перезавантажити сторінку. Це спричинило зростання фронтенд фреймворків, оскільки значна частина роботи була виконана на стороні клієнта [12].

Отож, стек поєднує в собі технологій, що використовуються для створення веб-програми. Будь-який веб-додаток буде створено з використанням кількох технологій (фреймворки, бібліотеки, бази даних тощо).

2.3.1 Вибір технології на стороні клієнта

У найпростішому випадку на стороні клієнта використовується просто HTML розмітка в поєднанні з мовою програмування JavaScript. Проте станом на зараз існує великий вибір бібліотек та фреймоворків, які використовуються на стороні клієнта.

Фреймворки JS — це бібліотеки програмування JavaScript, в яких є попередньо написаний код для використання у стандартних функціях та завданнях програмування. Це основа для створення веб-сайтів або веб-застосунків навколо.

Почнемо з того, навіщо потрібні фреймворки JavaScript. Кодування цілком можливе без їх використання, але правильно підібране середовище може значно полегшити роботу. Більш того, вони безкоштовні і з відкритим вихідним кодом, тому немає ризику [13].

Насамперед це підвищить продуктивність. Потрібно розглядати це як свого роду обхідний шлях: доведеться писати менше за код вручну, тому що вже є заздалегідь написані і готові до використання функції та шаблони. Деякі компоненти веб-сайту не повинні бути виготовлені за індивідуальним замовленням, тому можна створювати та розширювати заздалегідь створені компоненти.

Фреймворки більш адаптовані для дизайну веб-сайтів, і більшість розробників сайтів віддають перевагу їм. Отож, розглянемо найкращі JavaScript фреймворки.

React в даний час лідером в галузі інфраструктури JavaScript UI є React. Спочатку розробники Facebook почали працювати над цим, щоб спростити свою роботу. Програма під назвою Facebook Ads зростала дуже швидко, що означало складне управління та підтримку. В результаті команда почала створювати структуру, яка допоможе їм ефективно. У них був ранній прототип до 2011 року, а через два роки, структура була з відкритим вихідним кодом і доступна для громадськості [14]. В даний час його використовують багато бізнес-гіганти: AirBNB, PayPal, Netflix і т.д. На рисунку 2.2 показана інформація про бібліотеку React.

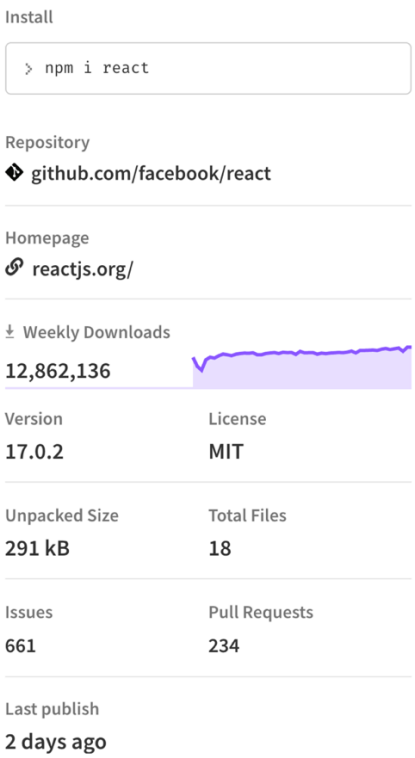


Рисунок 2.2 — Інформація про бібліотеку React

React базується на компоненті багаторазового використання. Простіше кажучи, це блоки коду, які можна класифікувати як класи чи функції. Кожен компонент представляє певну частину сторінки, таку як логотип, кнопку або

поле введення. Параметри, що використовуються ними, називаються реквізитами, що означає властивості. Говорячи про синтакс, більшість розробників сходяться на думці, що React легко освоїти, коли рохзробник вже знаєте JavaScript.

React використовує JSX, синтаксис XML, який поєднує JavaScript і HTML. Це не шаблон JavaScript; це повний JavaScript.

Angular — одне з найпотужніших середовищ JavaScript. Google використовує цю платформу для розробки односторінкової програми (SPA). Це середовище розробки відоме насамперед тому, що воно надає розробникам найкращі умови для об'єднання JavaScript з HTML та CSS. Понад півмільйона сайтів, таких як google.com, youtube.com тощо, використовують Angular. На рисунку 2.3 наведена інформація про фреймворк.

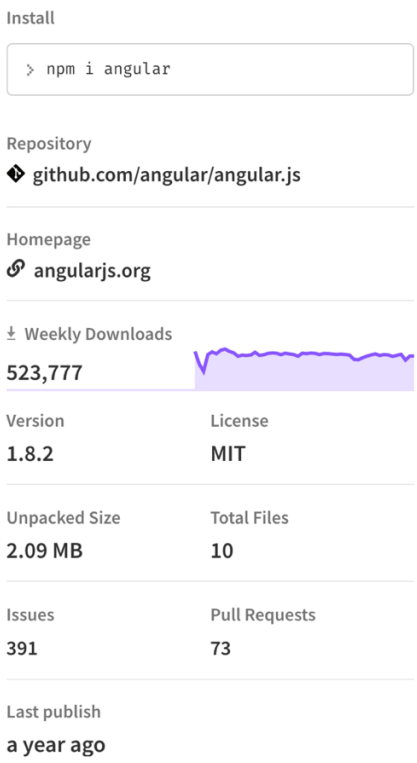


Рисунок 2.3 — Інформація про фреймворк Agular

Це також переважне середовище інтерфейсу JavaScript для розробників додатків Google. Angular має компонентну структуру, як і React. розробник може маніпулювати, вкладати та використовувати їх за необхідності.

Також потрібно буде використовувати TypeScript, щоб написати програму в Angular. Це розширений набір JavaScript, який використовує той же синтаксис, але також підтримує статичну типізацію та класи. У TypeScript можна отримати модифікатори доступу, перерахування, узагальнення, гібридні типи та багато іншого [15].

Vue — це JavaScript-фреймворк з відкритим вихідним кодом для створення креативного інтерфейсу. Інтеграція з Vue у проектах, які використовують інші бібліотеки JavaScript, спрощена, оскільки вона розроблена для адаптації.

Більше 36 000 веб-сайтів зараз використовують Vue. Такі компанії, як Stackoverflow, PlayStation і т.д., покладаються на Vue для своїх сайтів інтерфейсу користувача. На рисунку 2.4 наведена інформація про фреймворк.

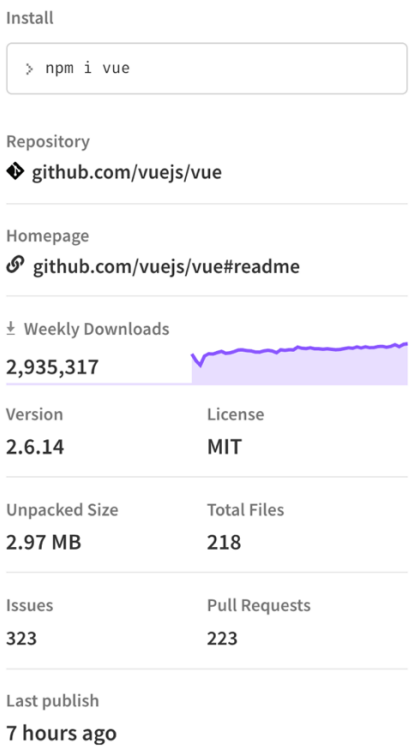


Рисунок 2.4 — Інформація про фреймворк Vue.js

Vue.js також досить простий у освоєнні: все, що потрібно, це JavaScript і HTML. Іншою сильною стороною Vue.js є його інтерфейс командного рядка (CLI). Це базовий інструмент, який прискорює розробку, пропонуючи безліч плагінів, пресетів, миттєвого прототипування та інтерактивного інструменту

розробки проектів. Деякі з його функцій включають компоненти, шаблони, переходи та двостороннє зв'язування даних, а також фокус реактивності. Реактивність виникає при зміні або оновленні будь-якого з об'єктів JavaScript у Vue. Vue.js використовує те, що називається Shadow DOM, що робить рендеринг сторінки швидким [16].

JQuery, мабуть, найпопулярніша бібліотека JavaScript з такою кількістю функцій для сучасної розробки. JQuery — це швидка та лаконічна бібліотека JavaScript, створена Джоном Резігом у 2006 році. Це кроссплатформенна бібліотека JavaScript, призначена для спрощення HTML-скриптингу на стороні клієнта.

Понад 19 мільйонів веб-сайтів зараз використовують jQuery. Такі компанії, як WordPress, Facebook, Google, IBM та багато інших, покладаються на jQuery для забезпечення свого роду перегляду веб-сторінок. На рисунку 2.5 показано інформація про бібліотеку.

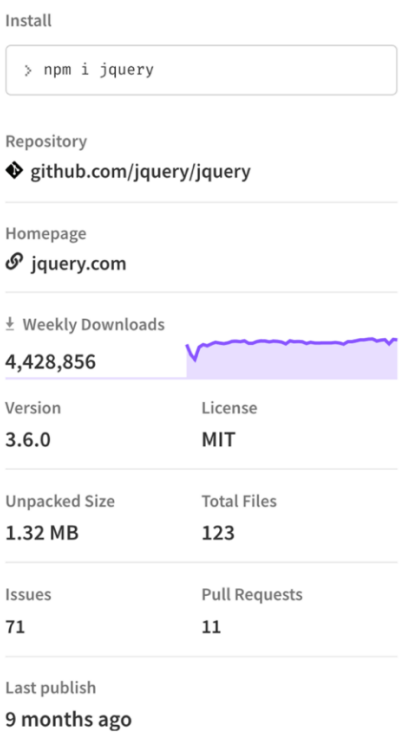


Рисунок 2.5 — Інформація про бібліотеку jQuery

Можна використовувати API jQuery для обробки, анімації та маніпулювання подією в HTML-документі, також відомому як DOM. Крім

того, jQuery використовується з будівельними інструментами Angular та React App. Одним словом, одна з найважливіших бібліотек JavaScript для веб-розробки [17].

BackboneJS — це одна з найпопулярніших платформ JavaScript. Його можна використовувати для створення односторінкової програми. Розробка цього фреймворку передбачає, що це функції із сервера повинні проходити через API, що допоможе досягти складної функціональності з допомогою написання меншої кількості коду.

BackboneJS — це легка бібліотека JavaScript, яка дозволяє розробляти та структурувати клієнтські програми, що працюють у веб-браузері.

На відміну від інших середовищ, Backbone доручає розробнику вибрати правильний інструмент, який найкраще підходить для цього проекту. Більше півмільйона сайтів зараз використовують Backbone, включаючи tumblr.com, espn.com, soundcloud.com та багато інших [18]. На рисунку 2.6 інформація про бібліотеку.

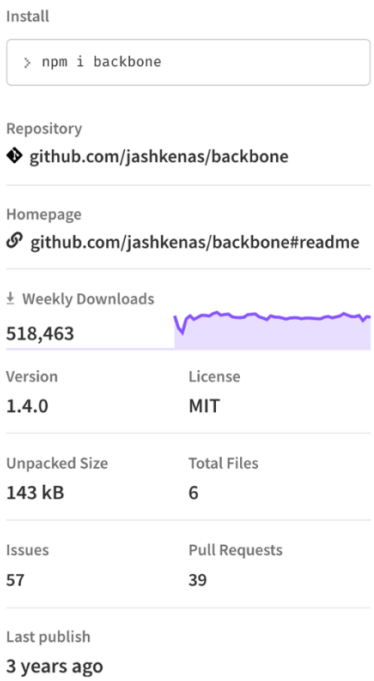


Рисунок 2.6 — Інформація про бібліотеку BackboneJS

Ember.js це JavaScript фреймворк з відкритим вихідним кодом, який був спочатку випущений Єгудою Кацем в 2011 році. Спочатку він називався

SproutCore 2.0, перш ніж став називатися Ember.js. Робота над Ember Framework почалася в 2011 році, а версія 1.0 була випущена через два роки. На рисунку 2.7 наведена інформація про фреймоворк.

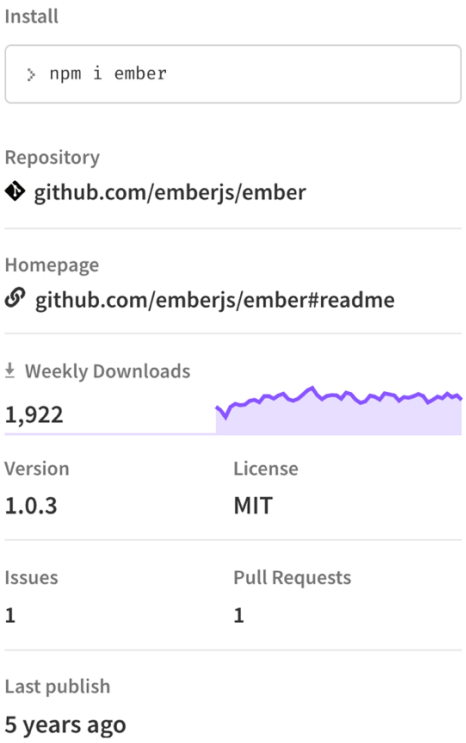


Рисунок 2.7 — Інформація про фреймворк Ember.js

Він також добре масштабується і може використовуватися для великих проектів. Apple Music, яка має понад 60 мільйонів передплатників у всьому світі, була побудована на Ember. Щоб краще побачити, що можна зробити за допомогою цього фреймворка, можна використати інструмент під назвою Ember Inspector, який дозволяє ближче познайомитися з проектами Ember в Інтернеті.

Ember має відмінний інструмент для складання, запозичений з багатьох інших середовищ SPA, званий Ember CLI. Цей інструмент збирання має все необхідне для початку роботи. Там вбудовано роутер для навігації. Також, якщо потрібно пройти тестування, то там також вбудований такий інструмент. Ember.js дуже продуманий, гнучкий [19].

Polymer — це бібліотека JavaScript з відкритим вихідним кодом, підтримувана Google для створення веб-застосунків з використанням веб-

компонентів. Він також підтримує як односторонню, так і двосторонню прив'язку даних, створюючи більш широку сферу застосування.

Коли порівнювати Angular з Polymer, оскільки обидва вони створені Google, Angular є комплексним середовищем для розробки веб-додатків, тоді як Polymer — це лише бібліотека для розробки веб-компонентів. Це була перша бібліотека, яка дозволяла створювати інтерактивні програми за допомогою веб-компонентів. В даний час більше 3000 веб-сайтів використовують Polymer, наприклад virustotal.com, rogers.com, zeplin.io і так далі [20]. На рисунку 2.8 зображена інформація про бібліотеку.

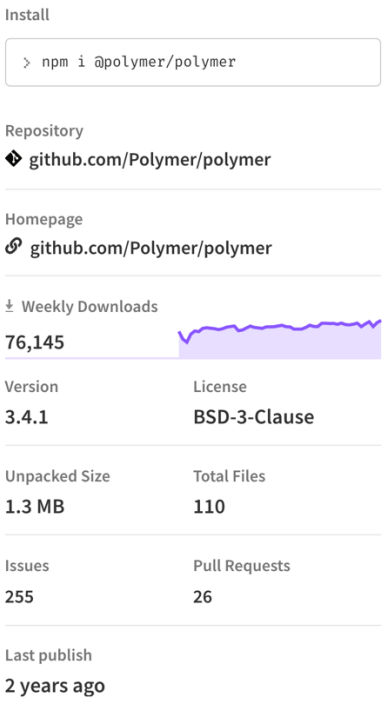


Рисунок 2.8 — Інформація про бібліотеку Polymer

2.3.2 Вибір технологій на стороні сервера

Так само, коли справа доходить до серверної веб-розробки насамперед потрібна мова програмування серверної частини, щоб веб-сайт працював разом з різними іншими інструментами та технологіями, такими як бази даних, фреймворки, веб-сервери і т.д.

Але оскільки є сотні мов програмування, необхідно вибрати мову програмування, враховуючи різні параметри, такі як вимоги проекту, його

крива навчання, продуктивність, надійність тощо. Крім того, також потрібно враховувати попит та популярність конкретної мови програмування у світі технологій.

Щоразу, коли йдеться про веб-розробку швидше за все, у 9 із 10 випадків йдеться про назву JavaScript. Згідно з щорічними звітами різних популярних платформ, таких як Stack Overflow та Octoverse, JavaScript є однією з найбільш кращих та провідних мов програмування у світі технологій. Однією з основних причин цього є те, що конкретна мова може використовуватися як для інтерфейсної веб-розробки, так і для внутрішньої веб-розробки. Дивлячись на кілька минулих тенденцій та статистику, можна сказати, що популярність Node.js якимось чином збільшила використання JavaScript як внутрішній для веб-розробки [21].

Тим часом, мова надає вам кілька чудових функцій для внутрішньої розробки, таких як полегшена мова сценаріїв, динамічний набір тексту, що інтерпретується, підтримка об'єктно-орієнтованого програмування, перевірка на стороні клієнта. Фреймворки JavaScript для серверної веб-розробки: Next.js, Express, MeteorJS і т.д. Популярні сайти, що використовують JavaScript: Facebook, Google, eBay і т.д.

Хоча Python досить відомий серед людей своєю сумісністю з передовими технологіями, такими як машинне навчання, Інтернет речей (IoT), Data Science і т. д. Дозвольте нам сказати вам, що ця збагачувальна мова програмування широко використовується і дуже підходить для серверної веб-розробки. також. Навіть один із провідних IT-гігантів в даний час Google значною мірою покладається на Python, і це одна з трьох основних мов, що використовуються Google (дві інші — Java та C++). Однією з основних переваг використання Python для веб-розробки є величезна колекція стандартних бібліотек, які роблять роботу розробників порівняно простою та ефективною [22]. Додаткові видатні та унікальні особливості Python, такі як покращена читаність коду. більш проста інтеграція з іншими мовами, підтримка програмування GUI, переносимість. Фреймворки Python для серверної веб-

розробки: Django, Flask, Pyramid і т.д. Популярні веб-сайти, що використовують Python: Spotify, Pinterest, Instacart і т.д.

PHP (або, можна сказати, препроцесор гіпертексту) — ветеран у світі веб-розробки. Ця серверна мова сценаріїв з відкритим вихідним кодом створена в 1994 році і спеціально використовується для веб-розробки. Оскільки це мова, що інтерпретується вона також не вимагає компілятора, а також може працювати практично у всіх основних операційних системах, таких як Windows, Linux, macOS, Unix і т. д.

Говорячи про розширюючі функції PHP, таких дуже багато. простота в освоєнні, кросплатформова сумісність, функції ООП, підтримка різних стандартних баз даних, таких як MySQL, SQLite і т. д., величезна підтримка спільноти та багато інших. В іншому PHP дуже безпечний як мова сценаріїв на стороні сервера, оскільки PHP є безліч хеш-функцій для шифрування даних користувача. Зокрема, PHP фреймворки для серверної веб-розробки: Laravel, CodeIgniter, Symfony і т.д. Популярні веб-сайти, які використовують PHP: WordPress, MailChimp, Flickr і т.д. [23].

Java — це ще один приклад мови програмування для серверної веб-розробки. Об'єктно-орієнтована мова програмування широко використовується для розробки веб-додатків корпоративного рівня поряд з розробкою додатків для Android, настільних додатків, наукових додатків тощо. Основна перевага використання Java полягає в тому, що він працює за принципом Write Once Run Anywhere, т. е. , скомпільований код Java може бути виконаний на будь-якій платформі, що підтримує Java, без необхідності повторної компіляції.

Говорячи конкретніше, код Java спочатку компілюється в байтовий код, який залежить від машини, та був цей байтовий код виконується на JVM незалежно від базової архітектури. Крім того, Java підтримує багатопоточність, яка дозволяє одночасне виконання двох або більше потоків для максимального використання ЦП. Фреймворки Java для серверної веб-

розробки: Spring, Struts, Grails. Популярні веб-сайти, що використовують Java: LinkedIn, IRCTC, Yahoo і т.д. [24].

Ruby — це інтерпретована мова програмування загального призначення, яка підтримує різні парадигми програмування, такі як процедурне, функціональне та об'єктно-орієнтоване програмування. Ця мова широко використовується для веб-розробки по всьому світу і дуже рекомендується новачкам для початку роботи з серверною веб-розробкою, так як він порівняно простіше в освоєнні. Як і Python, Ruby також фокусується на підвищенні продуктивності розробників, що прискорює процес веб-розробки. Конкретна мова підтримує майже всі основні платформи, такі як Windows, Mac та Linux, і дозвольте нам також сказати вам, що Ruby сильно заснований на багатьох інших мовах програмування, таких як Perl, Lisp, Eiffel, Ada і т.д. Duck набір тексту, автоматичний збір сміття, велика стандартна бібліотека, настроювана поведінка відправки, гнучкість. Ruby Frameworks для серверної веб-розробки: Ruby on Rails, Sinatra, Grape і т.д. Популярні сайти, що використовують Ruby: Airbnb, Shopify, Slideshare [25].

6. Golang — це була одна з 5 найулюбленіших мов програмування розробниками у всьому світі. Go — це статично типізована мова програмування, розроблена в Google і має синтаксис, дуже схожий на мову C. Мова дозволяє розробникам більш ефективно створювати масштабовані та безпечні веб-програми. Однією з основних переваг використання Go є те, що забезпечує відмінну підтримку багатопоточності, а також має функцію складання сміття для автоматичного управління пам'яттю.

Деякі з інших значних особливостей мови Go — це простий у вивченні, код, що підтримується, Google, скомпільована мова, управління пакетами, потужна стандартна бібліотека, підтримка паралелізму, висока продуктивність і багато іншого. Go Framework для серверної веб-розробки: beego, echo, revel тощо. Популярні веб-сайти, що використовують Go: Dropbox, SoundCloud, Dailymotion [26].

C# — одна з тих небагатьох мов, яка протягом останніх кількох років постійно входить до п'ятірки найкращих мов програмування за різними стандартними індексами. Однак вам необхідно знати, що ця універсальна мова спочатку була розроблена Microsoft в першу чергу для середовища. Поряд із серверною веб-розробкою, тепер C# широко використовується в багатьох областях, таких як розробка програм Windows, розробка ігор та багато інших. Крім того, C# пропонує багатий набір бібліотек, які допомагають розробникам прискорити та підвищити ефективність процесу розробки. Отже, C# Framework для серверної веб-розробки .NET Популярні веб-сайти, що використовують C#: GoDaddy, Marketwatch, Stack Overflow [27].

У додатку Г наведена статистика використання різних мов програмування, де детальніше її можна розглянути. У додатку Д у вигляді списка наведена фінальні рейтинги мов програмування.

2.3.3 Вибір бази даних

База даних — це інструмент для збору та в організатора відомостей. У базах даних можуть зберігатися відомості про товари, товари, замовлення та інші дані. Багато баз даних починаються зі списку в word-processing program або spreadsheet.

У міру зростання списку даних з'являються надлишкові і невідповідності. Дані стає важко зрозуміти у формі списку, і існує обмежений спосіб пошуку або виведення множини даних для перевірки. Коли ці проблеми почнуть з'являтися, краще перенести дані до бази даних, створеної системою управління базами даних (СУБД), такої як Access.

Почнемо з трьох типів БД, які все ще можуть зустрічатися в спеціалізованих середовищах, але переважно замінені надійними та продуктивними альтернативами.

Перший та найпростіший спосіб зберігання даних — текстові файли, рисунок 2.9. Метод застосовується сьогодні для роботи з невеликими обсягами інформації [28]. Для розділення полів використовується спеціальний символ:

кома або крапка з комою в csv-файлах датасетів, двокрапка або пробіл в *nix-подібних системах:

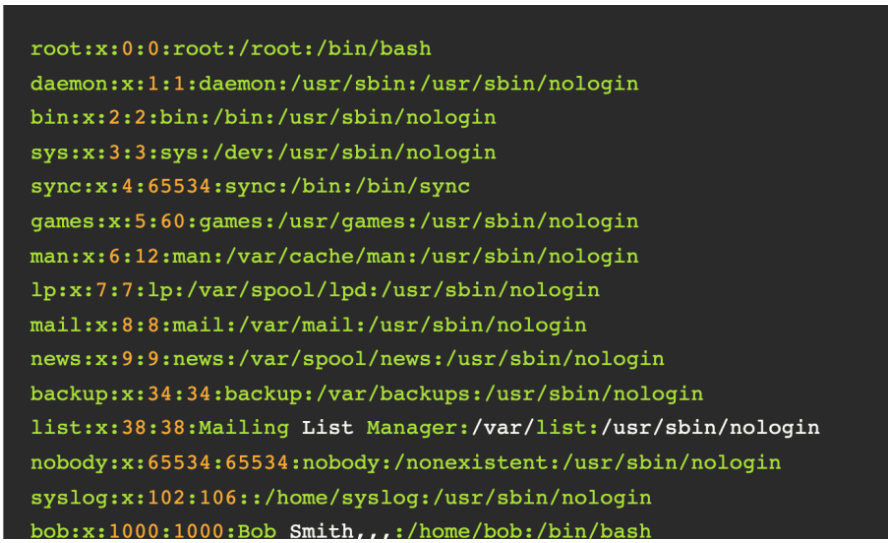


Рисунок 2.9 — Проста структура зберігання даних

Наслідки:

— обмежений тип і рівень складності інформації, що зберігається;

— важко встановити зв'язок між компонентами даних;

— відсутність функцій паралелізму;

— практичні лише для систем з невеликими вимогами до читання та запису;

— використовуються для зберігання конфігураційних даних;

— немає потреби у сторонньому програмному забезпеченні.

Приклади:

— /etc/passwd та /etc/fstab у *nix-системах

— csv-файли

Реляційні бази даних — найстаріший тип широко використовуваних БД загального призначення. Дані та зв'язки між даними організовані за допомогою таблиць показано на рисунку 2.10. Кожен стовпець у таблиці має ім'я та тип. Кожен рядок представляє окремий запис або елемент даних у таблиці, що містить значення для кожного зі стовпців [29].

Наслідки:

— поле в таблиці, зване зовнішнім ключем, може містити посилання на стовпці інших таблицях, що дозволяє їх з'єднувати;

— високоорганізована структура і гнучкість робить реляційні БД сильними і адаптованими до різних типів даних;

— для доступу до даних використовується мова структурованих запитів (SQL);

— надійний вибір для багатьох програм.

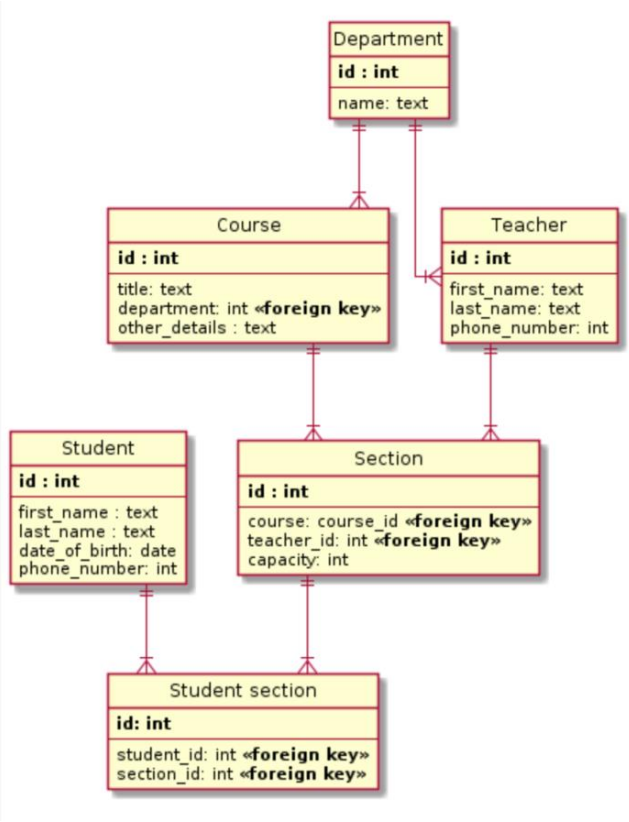


Рисунок 2.10 — Схема реляціної бази даних

Приклади:

— MySQL

— MariaDB

— PostgreSQL

— SQLite

NoSQL бази даних — група типів БД, які пропонують підходи, відмінні від стандартного реляційного шаблону. Говорячи NoSQL, мають на увазі або

"не-SQL", або "не тільки SQL", щоб уточнити, що іноді допускається SQL-подібний запит.

У базах даних «ключ-значення» для зберігання інформації ви надаєте ключ та об'єкт даних, який потрібно зберегти. Наприклад, об'єкт JSON, зображення або текст [30]. Щоб запросити дані, відправляєте ключ та отримуєте blob-об'єкт, рисунок 2.11.

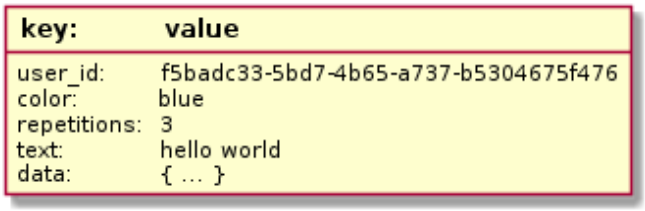


Рисунок 2.11 — Зберігання диних в нереляційні базі даних

Наслідки:

— сховища забезпечують швидкий та маловитратний доступ;

— часто зберігають дані конфігурацій та інформацію про стан даних, представлених словниками або хеш;

— немає жорсткої схеми відносини між даними, у таких БД часто зберігають одночасно різні типи даних;

— розробник відповідає визначення схеми іменування ключів і те, щоб значення мало відповідний тип/формат.

Приклади:

— Redis

— memcached

— etcd

Серед NoSQL баз даних і підтипи. Їх розглянуто далі. Документні бази даних (також документоорієнтовані БД або сховища документів), їх зображено на рисунку 2.12, спільно використовують базову семантику доступу та пошуку сховищ ключів та значень.

Такі бази даних також використовують ключ для унікальної ідентифікації даних. Різниця між сховищами «ключ-значення» та

документними БД полягає в тому, що замість зберігання blob-об'єктів, документоорієнтовані бази зберігають дані у структурованих форматах JSON, BSON або XML [31].

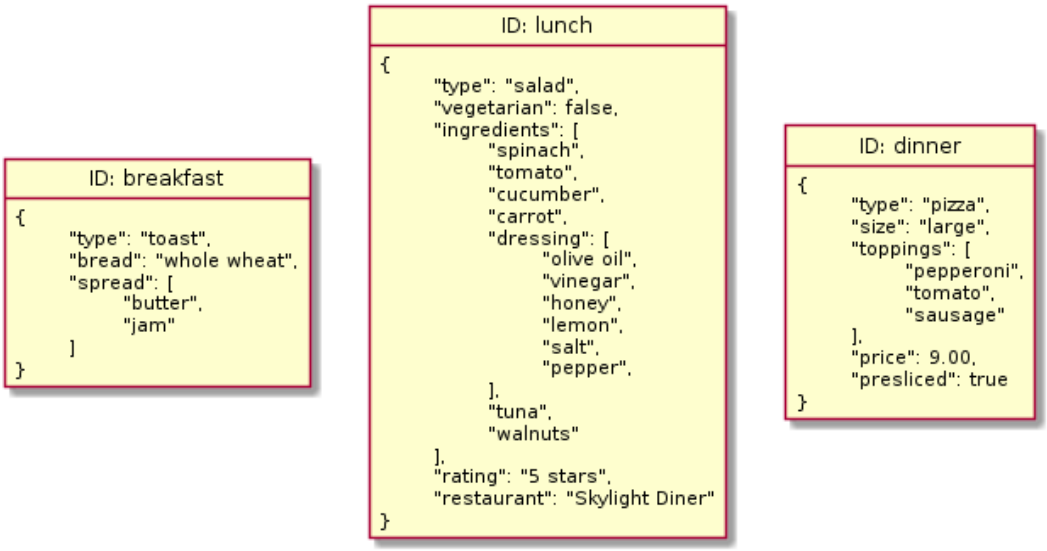


Рисунок 2.12 — Документна база даних

Наслідки:

— база даних не визначає певний формат або схему;

— кожен документ може мати внутрішню структуру;

— документні БД є добрим вибором для швидкої розробки;

— будь-якої миті можна змінювати властивості даних, не змінюючи структуру чи самі дані.

Приклади:

— MongoDB

— RethinkDB

Графова база даних зображена на рисунку 2.13. Замість зіставлення зв'язків з таблицями та зовнішніми ключами, графові бази даних встановлюють зв'язки, використовуючи вузли, ребра та властивості.

Графові бази представляють дані як окремих вузлів, які можуть мати будь-яку кількість пов'язаних із нею властивостей.

Наслідки:

— виглядають аналогічно мережевим;

— фокусуються на зв'язках між елементами;

— явно відображає зв'язок між типами даних;

— не вимагають покрокового обходу переміщення між елементами;

— немає обмежень у типах представлених зв'язків.

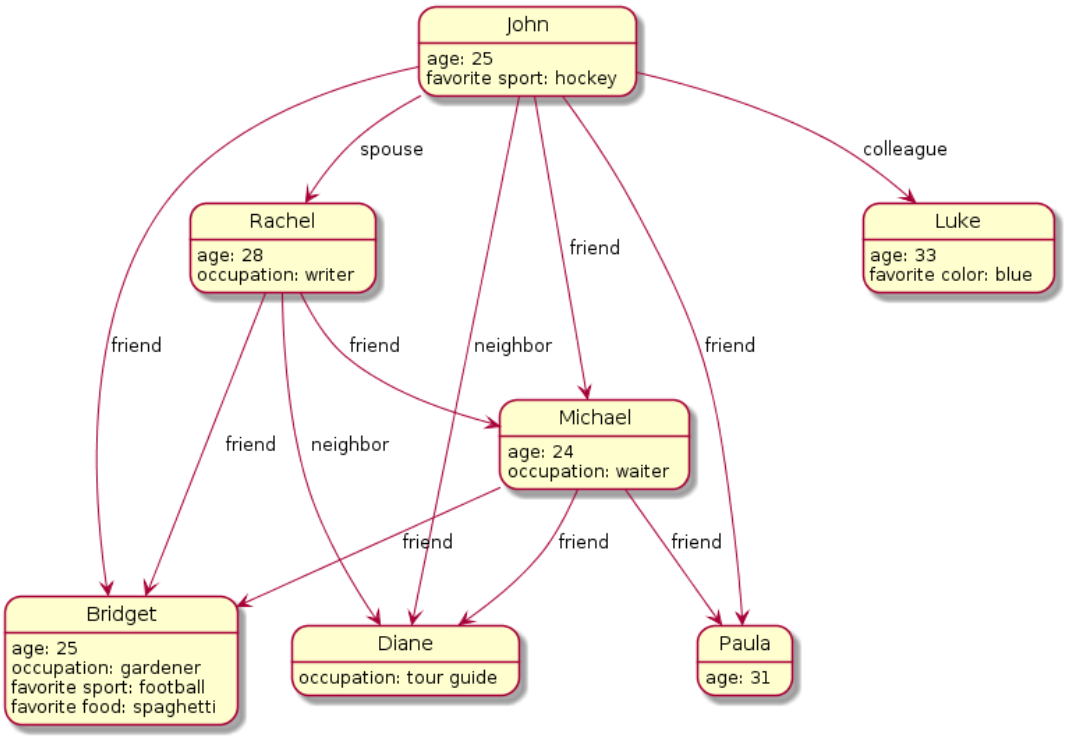


Рисунок 2.13 — Графова база даних

Приклади:

— Neo4j

— JanusGraph

— Dgraph

Колонкові бази даних (також нереляційні колонкові сховища або бази даних із широкими стовпцями) належать до сімейства NoSQL БД, але зовні схожий на реляційні БД, вони зображені на рисунку 2.14. Як і реляційні, колонкові БД зберігають дані, використовуючи рядки та стовпці, але з іншим зв'язком між елементами.

У реляційних БД усі рядки повинні відповідати фіксованій схемі. Схема визначає, які стовпці будуть у таблиці, типи даних та інші критерії. У колонкових базах замість таблиць є структури колонкові сімейства. Сімейства

містять рядки, кожен із яких визначає власний формат. Рядок складається з унікального ідентифікатора, який використовується для пошуку, за яким слідують набори імен та значень стовпців [31].

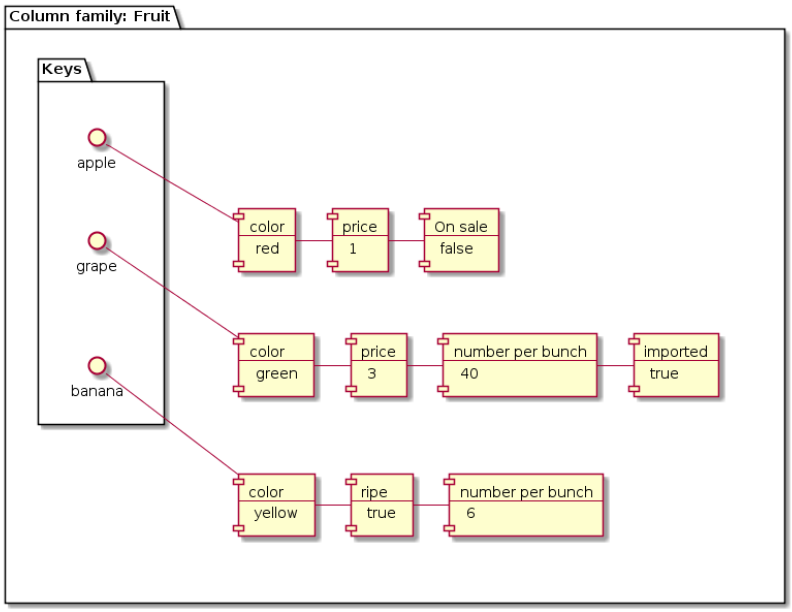


Рисунок 2.14 — Колонкова база даних

Наслідки:

— БД зручні при роботі з програмами, що вимагають високої продуктивності;

— дані та метадані записи доступні по одному ідентифікатору;

— гарантовано розміщення всіх даних із рядка в одному кластері, що спрощує сегментацію та масштабування даних.

Приклади:

— Cassandra

— HBase

Підсумовуючи, було обрано мову програмування Javascript, бібліотеку React та нереляціну базу даних MongoDB.

Стек MERN — це JavaScript стек, розроблений для спрощення процесу розробки. MERN включає чотири компоненти з відкритим вихідним кодом:

MongoDB, Express, React і Node.js. Ці компоненти забезпечують комплексне середовище для розробників.

Основною перевагою для розробників, які використовують стек MERN, є те, що кожен рядок коду написаний на JavaScript. Це мова програмування, яка використовується скрізь як для коду на стороні клієнта, так і для коду на стороні сервера. З однією мовою між рівнями немає потреби у перемиканні контексту.

Для технічного стека з декількома мовами програмування розробники мають з'ясувати, як об'єднати. За допомогою стека JavaScript розробники повинні мати тільки JavaScript і JSON.

В цілому, використання стеку MERN дозволяє розробникам створювати високоефективні веб-програми. Структура MERN додатка показана на рисунку 2.15.

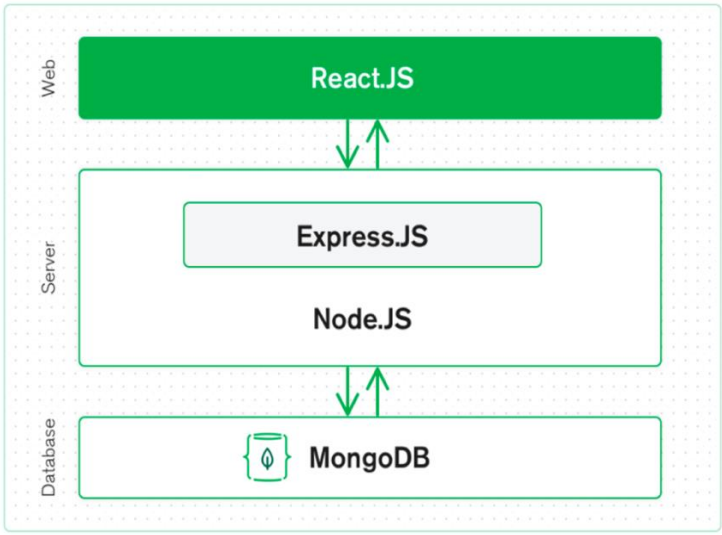


Рисунок 2.15 — Структура MERN додатка

3. РОЗРОБКА СИСТЕМИ

У цьому розділі буде описано кроки програманої реалізації системи.

3.1 Огляд середовища розробки

Для роботи існує багато середовищ розробки. Вони бувають як платними так і безкоштовними. З огляду на популярність і простоту користування було обрано середовище WebStorm.

WebStorm - це інтегроване середовище для розробки на JavaScript та пов'язаних з ним технологіях. Як і інші IDE JetBrains, WebStorm дозволяє автоматизувати рутинну роботу та легко справлятися зі складними завданнями, роблячи розробку більш цікавою.

За замовчуванням WebStorm має більше 100 встановлених доповнень, які забезпечують підтримку та зручну розробку, використовуючи різні JS-фреймворки, бібліотеки, різні нововведення CSS/HTML і т.д. Це лише мала частина того, з чим WebStorm здатний працювати із коробки:

Модифікація файлів .css, .html, .js з одночасним переглядом результатів, у деяких джерелах ця функціональність називається «редагування файлів на льоту» або «в реальному часі» або «без перезавантаження сторінки»). Також середовище представляє натупні можливості:

— підтримка HTML5;

— підтримка JSDoc;

— підтримка Node.js;

— підтримка React, JSX;

— можливості Zen Coding та Emmet;

— налагодження коду на JavaScript;

— віддалене розгортання за протоколами FTP, SFTP, на монтованих мережевих дисках тощо з можливістю автоматичної синхронізації;

— інтеграція із системами керування версіями: Subversion, Git, GitHub, Perforce, Mercurial, CVS підтримуються з коробки з можливістю побудови списку змін та відкладених змін;

— інтеграція із системами відстеження помилок.

Кожного року розробники працють, щоб випустити нову версію середовища. З поглядів на такі можливості це середовище найкраще підходить для роботи [32].

3.2.1 Налаштування сервера

Для створення системи було обрано стек технологій MERN. Бібліотека React є головним компонентом у цьому стеку. Використовувати React потрібно для візуалізації представлень (V у MVC), але як пов’язати решту програм разом, залежить від розроблюваного проекту.

Node або Node.js — програмна платформа, заснована на движку V8 (що транслює JavaScript в машинний код), що перетворює JavaScript з вузькоспеціалізованої мови на мову загального призначення. Node.js додає можливість JavaScript взаємодіяти з пристроями введення-виводу через свій API, написаний на C++, підключати інші зовнішні бібліотеки, написані різними мовами, забезпечуючи виклики до них з JavaScript-коду.

Node.js застосовується переважно на сервері, виконуючи роль веб-сервера. В основі Node.js лежить подієво-орієнтоване та асинхронне (або реактивне) програмування з неблокуючим введенням/виводом [33].

У роботі додатково використовується npm — Node Package Manager менеджер пакетів, що входить до складу Node.js. Менеджер npm складається із двох частин:

— CLI (інтерфейс командного рядка) - засіб для розміщення та скачування пакетів;

— онлайн-репозиторії, що містять пакети JS.

Кожен проект у JavaScript — будь то Node.js або веб-додаток може бути скопійований як npm пакет із власним описом та файлом package.json.

package.json можна уявити, як стікери (список пакетів потрібних версій) на npm-коробці (проект). Файл генерується командою npm init під час створення JavaScript/Node.js проекту з наступними метаданими:

— name: назва JS бібліотеки/проекту;

— version: версія проекту;

— description: опис проекту;

— license: ліцензія проекту.

Так як система складається з двох части, а саме з сервера і клієнта, тому потрібно створити папку, в якій створити ще дві папки для клієнта та сервера.

Для створення проекту для сервера, потрібно скористатись командою:

npm init

Ця команда створить фалй package.json, що зображений на рисунку 3.1, для опису проекту та залежностей

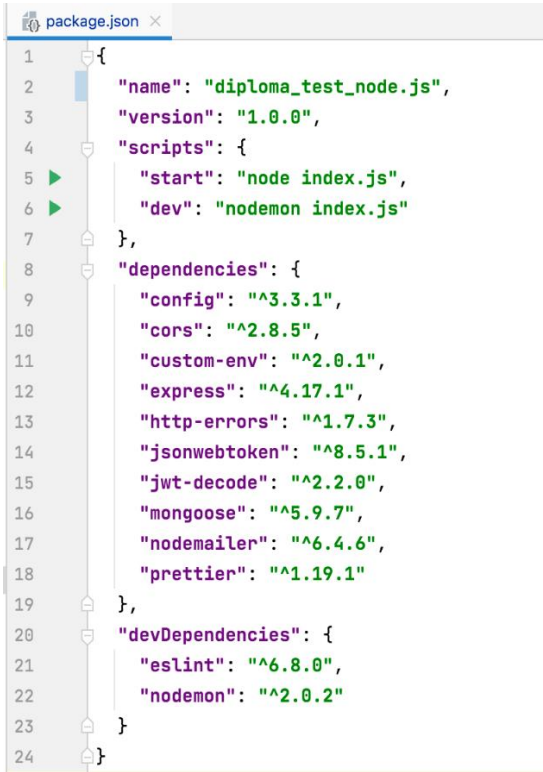


Рисунок 3.1 — Опис файла package.json

У файлі дано ім’я для проекту, у списку залежностей вказано, які додаткові біблеотеки будуть використовуватись в ході роботи.

Далі потрібно ввести команду, яка завантажить усі ці необхідні пакети у проект:

npm install

Далі потрібно сворити точку входу для сервера. Для цього потрібно створити файл index.js, в якому викликати метод start, який запустить сервер. Код функції наведено в лістингу 3.1

Лістинг 3.1 — Метод запуску сервера

```
const mongoose = require('mongoose');
const app = require('./app');
require('custom-env').env();
const start = () => {
  try {
    const DB =
'mongodb+srv://justPassword:justPassword@cluster0.tqfrv.mongodb.net
/tet?retryWrites=true&w=majority';
    Mongoose
      .connect(DB, {
        useNewUrlParser: true,
        useCreateIndex: true,
        useFindAndModify: false,
        useUnifiedTopology: true,
      })
      .then(() => console.log('DB connection successful!'));
    const port = process.env.PORT || 3001;
    app.listen(port, () => {
      console.log('server was started');
    });
  } catch (e) {  console.log(e);  }
};start();
```

У константі DB повинно зберігатись посилання на базу даних. У корені проекта також потрібно створити папки для роутингу, контролерів, моделей та для допоможних функцій.

Загалом вся струкутура сервера показана на рисунку 3.2

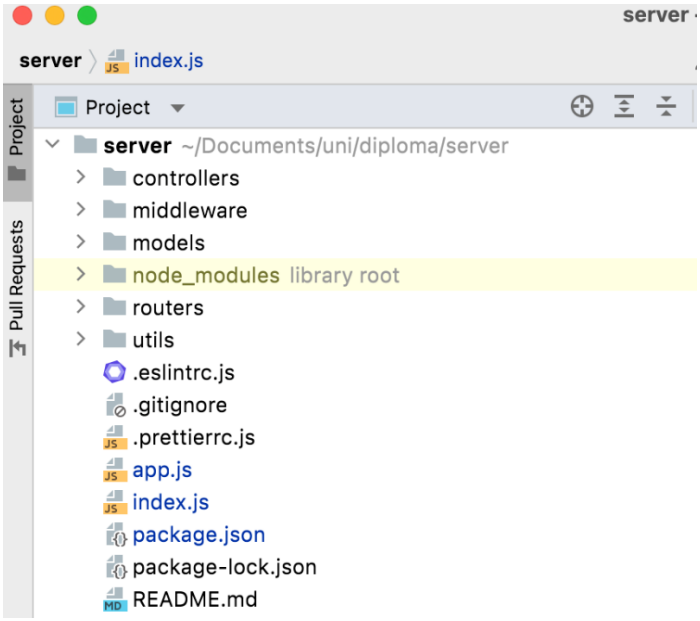


Рисунок 3.2 — Структура серверної частини програми

Файл app.js відповідає за марщрутизацію. Програмний код для маршрутизації файла наведено в лістингу 3.2

Лістинг 3.2 — Програмний код для маршрутизації

```
const express = require('express');
const cors = require('cors');
const authRouter = require('./routers/authRouter');
const sendInviteRouter = require('./routers/sendInviteRouter');
const userRouter = require('./routers/userRouter');
const testRouter = require('./routers/testRouter');
const middleware = require('./middleware/handleError');
const app = express();
app.use(cors());
app.use(express.urlencoded({ extended: true }));
```

```
app.use(express.json());
app.use(authRouter);
app.use('/mail', sendInviteRouter);
app.use('/user', userRouter);
app.use('/tests', testRouter);
module.exports = app;
```

На рисунку 3.3 показано структуру папки routes.

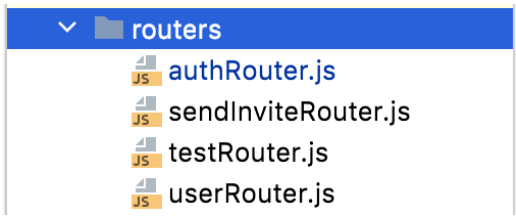


Рисунок 3.3 — Структура папки для маршритизації

Ця папка вміщує файли різного призначення. До прикладу, authRouter.js описує роутинг для авторизаці. Код наведено в лістингу 3.3.

Лістинг 3.3 — Програмний код для маршрутизації при авторизації

```
const router = require('express').Router();
const middleware = require('./../middleware/autoriz');const {
} = require('./../middleware/handleError');
const controllers = require('./../controllers/authController');
router.route('/user/register').post(catchAsyncError(controllers.register));
router.route('/user/login').post(controllers.login);
router.route('/verifyToken').get(controllers.verifyToken);
module.exports = router;
```

У папці models описуються моделі, тобто це є прототипом бази. На рисунку 3.4 показано вміст папки

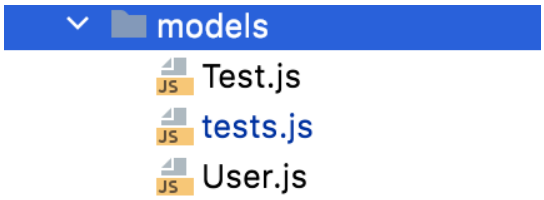


Рисунок 3.4 — Структура папки models

Поля які є в цій моделі, повинні бути і в базі даних. Приклад моделі для користувача наведено в лістингу 3.4

Лістинг 3.4 — Модель користувача

```
const mongoose = require('mongoose');
const userSchema = new mongoose.Schema(
  {
    name: String,
    surname: String,
    email: {
      type: String,
      unique: true,
      required: true,
    },
   group: String,
    comment: String,
    password: String,
    invited: {
      type: Boolean,
      default: false,
    },
    activated: {
      type: Boolean,
      default: false,
    },
```

```
        accessToTest: {
            type: [String],
            required: true,
        },
        results: [{
                results: [String],
                testId: String, }
        ], },
    { timestamps: true, });
const User = mongoose.model('User', userSchema);
module.exports = User;
```

Далі папка controllers описує набір функцій для валідації, викликає інші функції, викликає менеджера для роботи з базо даних, для обрахування якогось результату і виведення його. Для прикладу функція verifyToken перевіряє валідність токена при авторизації. Програмний код наведено в лістингу 3.5

Лістинг 3.5 — Програмний код функції верифікації

```
const verifyToken = async (req, res) => {
 const token = req.headers.authorization.split(' ')[1];
  if (!token) {
    return res.status(400).json({
      error: true,
     message: "Token is required."
    });
  }
  jwt.verify(token, 'secret', async (err, user) => {
        if (err) return res.status(401).json({
      error: true,
     message: "Invalid token."
```

```
    });
    const userData = await User.findOne({email: user.email});
    if (!userData) {
      return res.status(401).json({
        error: true,
        message: "Invalid user."
      });}
    const testData = await Test.find({_id: userData.accessToTest});
    const testsId = userData.results.map(({testId}) => testId);
    const availableTests = testData.filter(({_id}) => !testsId.includes(_id));
    const completedTests = testData.filter(({_id}) => testsId.includes(_id));
    return res.status(200).json({user: userData, test: availableTests,
completedTests, token});
  });};
```

Додаток Express, по суті, є серією викликів функцій проміжної обробки. Функції проміжної обробки (middleware) — це функції, що мають доступ до об'єкта запиту (req), об'єкта відповіді (res) і до наступної функції проміжної обробки в циклі запит-відповідь програми. Наступна функція проміжної обробки, як правило, позначається змінною next.

Функції проміжної обробки можуть виконувати такі завдання:

— виконання будь-якого коду;

— внесення змін до об'єктів запитів та відповідей;

— завершення циклу "запит-відповідь";

— виклик наступної функції проміжної обробки зі стеку;

— папка middleware створена для того, щоб в ній описати.

Тож папка middleware вміщує у собі функції проміжної обробки даних. Стуркутра папки на рисунку 3.5.

Файл autoriz.js перевіряє чи є в користувача токен чи потребує користувач нового токена для авторизації, а файл handleError.js є допоміжною

функцією для пошуку помилок. Весь код серверної частини міститься в додатку Е.

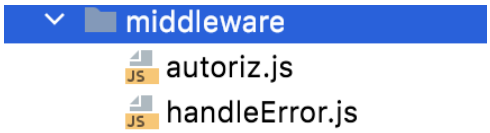


Рисунок 3.5 — Структура папки middleware

3.2.2 Налаштування клієнта

Так як клієнтська частина буде виконуватись за допомогою бібліотеки React, цю біліотеку потрібно встановити у проект. Для цього потрібно ввести команду

npm install create-react-app client

Структура проекту для клієнта зображена на рисунку 3.6

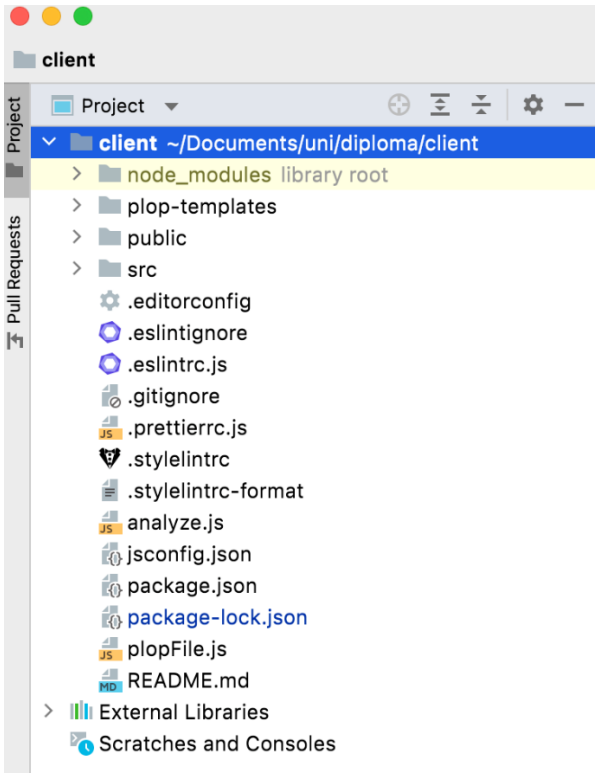


Рисунок 3.6 — Структура програми для клієнтської частини

Початковою точкою входу в проект є файл index.js. Програмний код наведено у лістингу 3.6

Лістинг 3.6 — Програмний код файла index.js.

```
import React from 'react';
import ReactDOM from 'react-dom';
import { BrowserRouter } from 'react-router-dom';
import App from './App';
import './index.css';
ReactDOM.render(
 <BrowserRouter>
  <div className="back">
   <App />
  </div>
 </BrowserRouter>,
 document.getElementById('root'),);
```

В ньому використовується функція render. ReactDOM.render() керує вмістом вузла контейнера, що передається вами. Будь-які існуючі елементи DOM всередині замінюються під час першого виклику. Більш пізні виклики використовують алгоритм відстеження змін React DOM для ефективного оновлення.

ReactDOM.render() не змінює вузол контейнера (змінює лише дочірні елементи контейнера). Якщо потрібно, можна вставити компонент до існуючого вузол DOM без перезапису існуючих дочірніх елементів.

ReactDOM.render() зараз повертає посилання на кореневий екземпляр ReactComponent.

React-елементи — це складові блоки React-додатків. Їх можна переплутати з більш відомою концепцією «компонентів», але на відміну від компонента елемент описує те, що користувач хочете побачити на екрані. React-елементи іммутабельні.

React-компоненти — це маленькі, повторно використовувані частини коду, які повертають React-елементи для відображення на сторінці.

Найпростіший React-компонент — це проста функція JavaScript, яка повертає елементи React.

В лістингу 3.7 наведено приклад компонента кнопки

Лістинг 3.7 — Програмний код для кнопки

```
import React from 'react';
import { string, func } from 'prop-types';
import './Button.scss';
const Button = ({ children, onClick, className, size }) => {
  return (
    <button type="submit" className={`${className} ${size}`}
onClick={onClick}>
      {children}
    </button>  );};
Button.propTypes = {
  children: string.isRequired,
  className: string.isRequired,
  onClick: func,
  size: string,
};
Button.defaultProps = {
  onClick: func,
  size: '',};
export default Button;
```

У будь-якій реальній програмі потрібні маршрути, і програма React не виняток. Користувач повинен бачити, де він знаходиться в програмі будь-який момент часу. А бачить він своє місце розташування в адресному рядку браузера. Отже, програма повинна вміти зіставляти певну URL-адресу з відповідною сторінкою.

Для цього у React має стороню бібліотеку для маршрутизації. Встановити її можна ввести у командний рядок наступне:

npm install react-router-dom

Ця бібліотека є досить простою у використанні і надає такі можливості:

— навігація по кліку (компонент <Link>);

— перенаправлення (компонент <Redirect>);

— маршрутизація (компонент Route);

— історія (властивість history).

Всі маршрути зберігаються у папці router, що зображено на рисунку 3.7

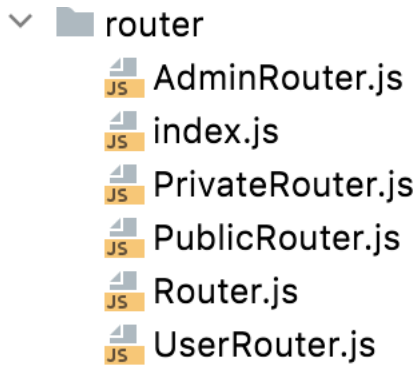


Рисунок 3.7 — Структура папки router

Передбачено, що будуть приватні маршрути та публічні. Публічні маршрути є доступними всюди. Приватні маршрути перебачені лише для авторизованих користувачів. Приклад приватного маршруту наведено в лістингу 3.8 У функції здійснюється перевірка чи має користувач токен чи ні.

Лістинг 3.8 — Програмний код приватного маршруту

```
import React from 'react';
import { Route, Redirect } from 'react-router-dom';
import { getToken } from '../utils/common';
function PrivateRouter({ component: Component, ...rest }) {
  return (
    <Route
      {...rest}
```

```
    render={(props) => getToken() ? <Component {...props} /> : <Redirect
to={{ pathname: '/login', state: { from: props.location } }} />}    />
  )}
export default PrivateRouter;
```

Контекст забезпечує спосіб передачі через дерево компонентів без необхідності передавати властивості вручну кожному рівні.

У типовій програмі React дані передаються зверху вниз (від батька до нащадка) через властивості props. Однак це може бути громіздким для певних типів властивостей (тема UI; уподобання, пов'язані з локалі), які потрібні для багатьох компонентів у додатку. Контекст надає спосіб спільного використання таких значень між компонентами без необхідності передавати властивість через кожен рівень дерева. На рисунку 3.8 зображено структуру папки для різних контекстів

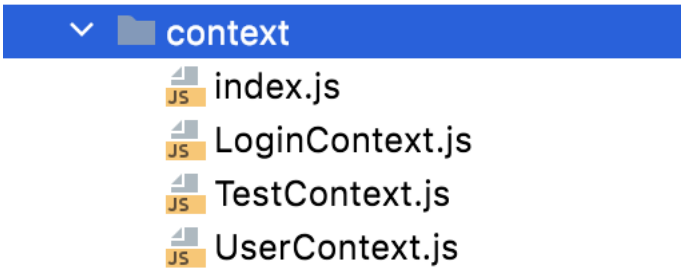


Рисунок 3.8 — Структура папки context

Приклад коду для контексту користувача наведено в лістингу 3.9.

Лістинг 3.9 — Програмний код контексту користувача

```
import React, { useContext } from 'react';
export const UserContext = React.createContext({
  user: {},
  setUser: () => {},
  userId: '',
  setUserId: () => '',
  filterUser: 'all',
```

```
  setFilterUser: () => '',
});
export const useUserContext = () => {
  const { user, setUser, userId, setUserId, filterUser, setFilterUser } =
useContext(UserContext);
  return { user, setUser, userId, setUserId, filterUser, setFilterUser };};
export default UserContext;
```

Весь код клієнтської частини наведено в додатку Ж.

3.2.3 Підключення до бази даних

Для підключення бази даних потрібно на офіційному сайті Mongoo https://cloud.mongodb.com/ створити проект, в якому створити базу даних. На рисунку 3.9 показано екран з базами даних. На рисунку 3.10 зображено структуру бази для тестів.

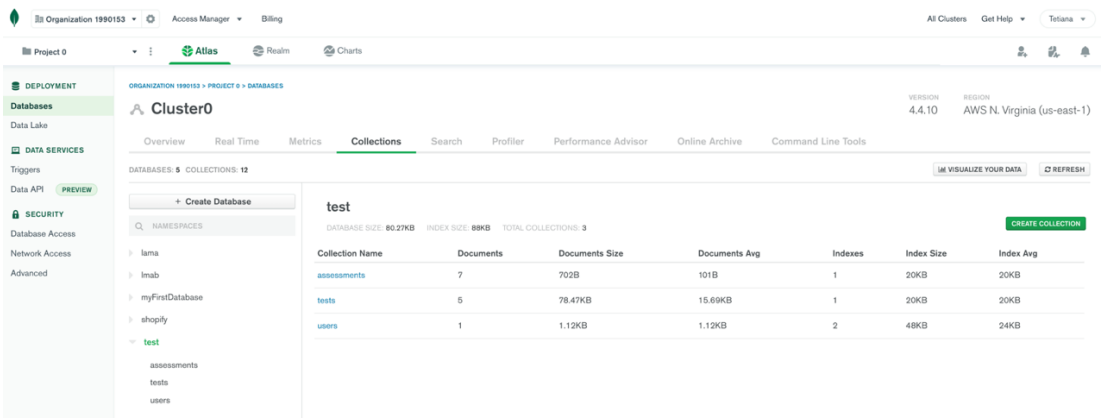


Рисунок 3.9 — Початковий екран MongooDB з базами даних

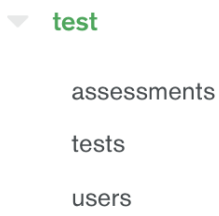


Рисунок 3.10 — структура бази даних для тестів

Структуру полів для користувача показано на рисунку 3.11.

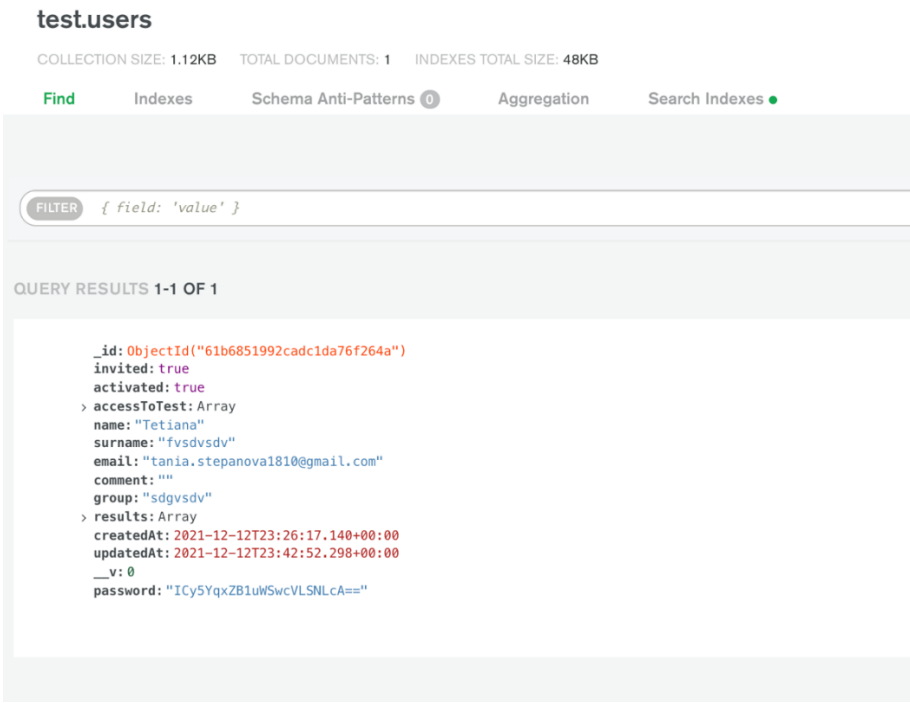


Рисунок 3.11 — Структура полів для користувача

У полі tests зберігаються всі питання і їх варінти відповідей. Об’єкст з полями доступу показано на рисунку 3.12.

Рисунок 3.12 — Структура тесту

Сторінка викладача має вигляд, який зображено на рисунку 3.13

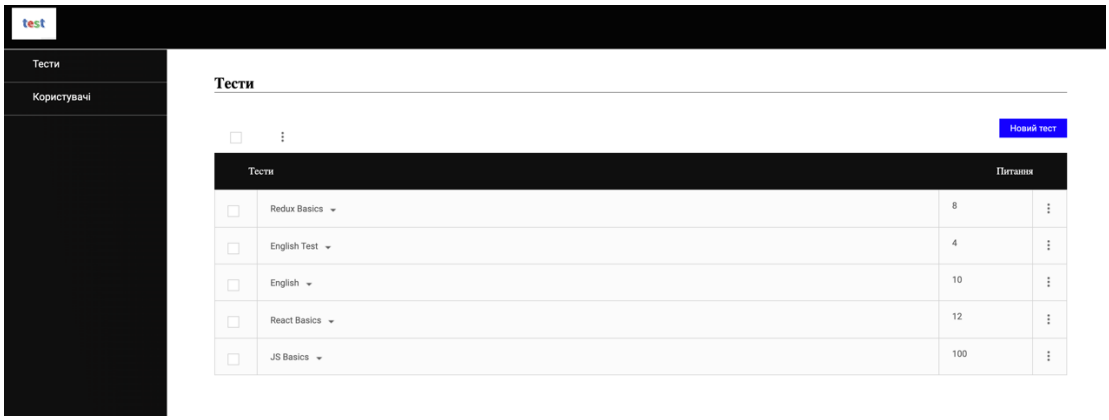


Рисунок 3.13 — Вигляд сторінки викладача

На цій сторінці викладач має змогу проглядати кількість користувачів, а також кількість тестів, зображено на рисунку 3.14.

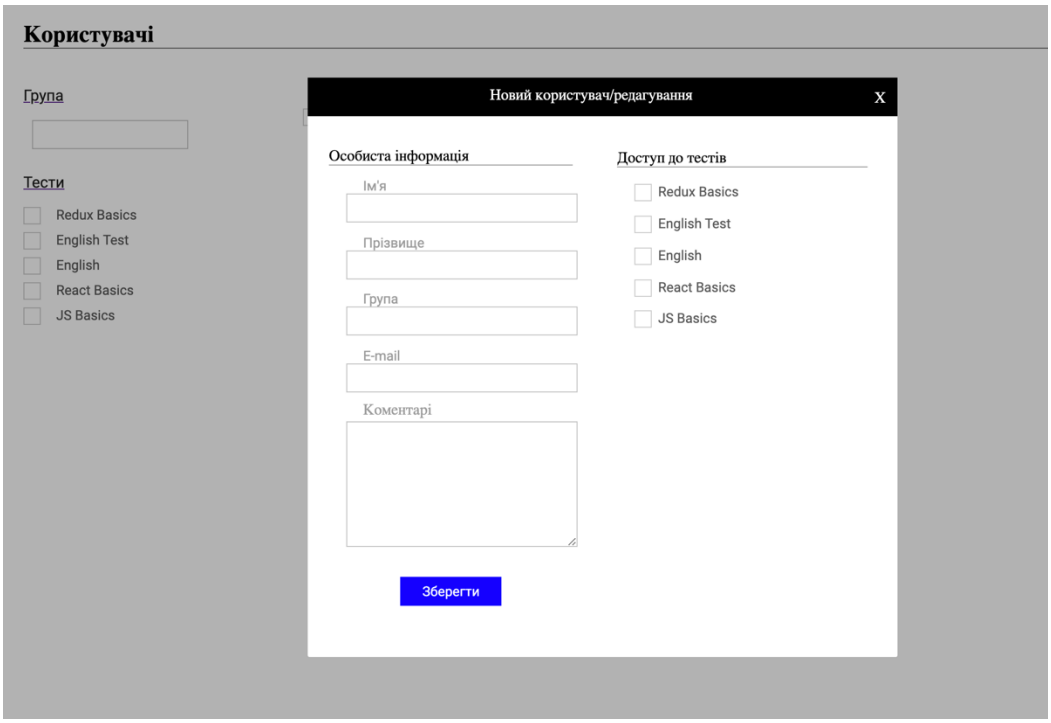


Рисунок 3.14 — Сторінка для додавання та редагування користувачів

Викладач може створювати список студентів, надавати доступ до тестів, а також надсилати листи з запрошенням для реєстрації в системі. На рисунку 3.15 зображено основні функції для маніпуляції.

Для відправлення запрошення про проходження тестів, викладачу потрібно згенерувати посилання і відправити його, це зображено на рисунку 3.16.

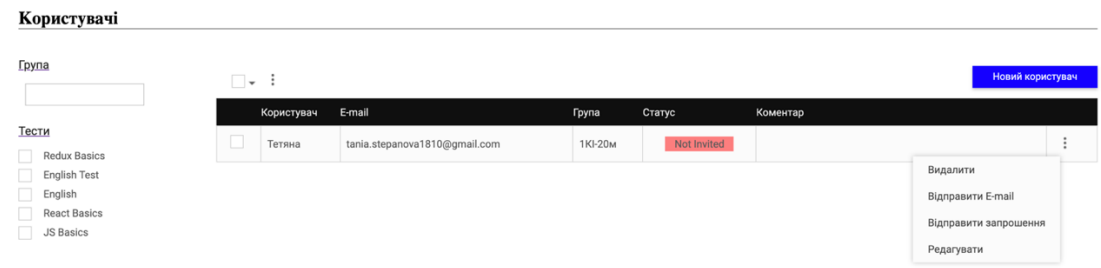


Рисунок 3.15 — Функції маніпуляції з користувачами

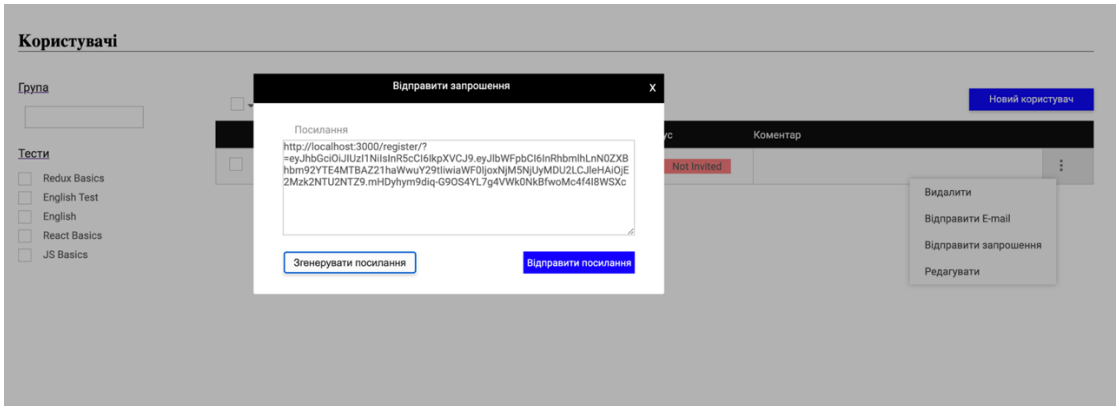


Рисунок 3.16 — Вікно з формою генерації та відпрвлення запрошення

Після цього на особисту пошту студента прийде запрошення, йому потрібно перейти по ньому і авторизуватись. На особистій сторінці студенту доступні тести, які дозволив викладач. На рисунку 3.17 показано сторінку студента з усіма доступними йому тестами.

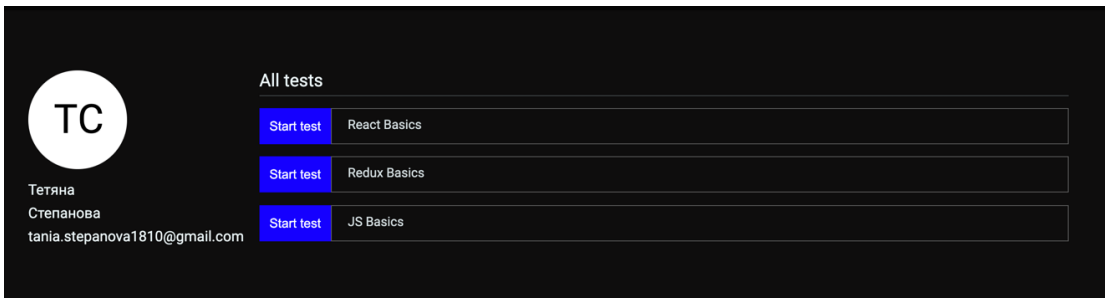


Рисунок 3.17 — Сторінка студента з усіма доступними тестами

По закінченні тестування, студент натискає кнопку Submit та відправляє результат на перевірку. Усі правильні та неправильні відповіді студент має

змогу бачити одразу у своєму кабінеті. На рисунок 3.18 показано сторінку з усіма тестами, а також результати по тестах.

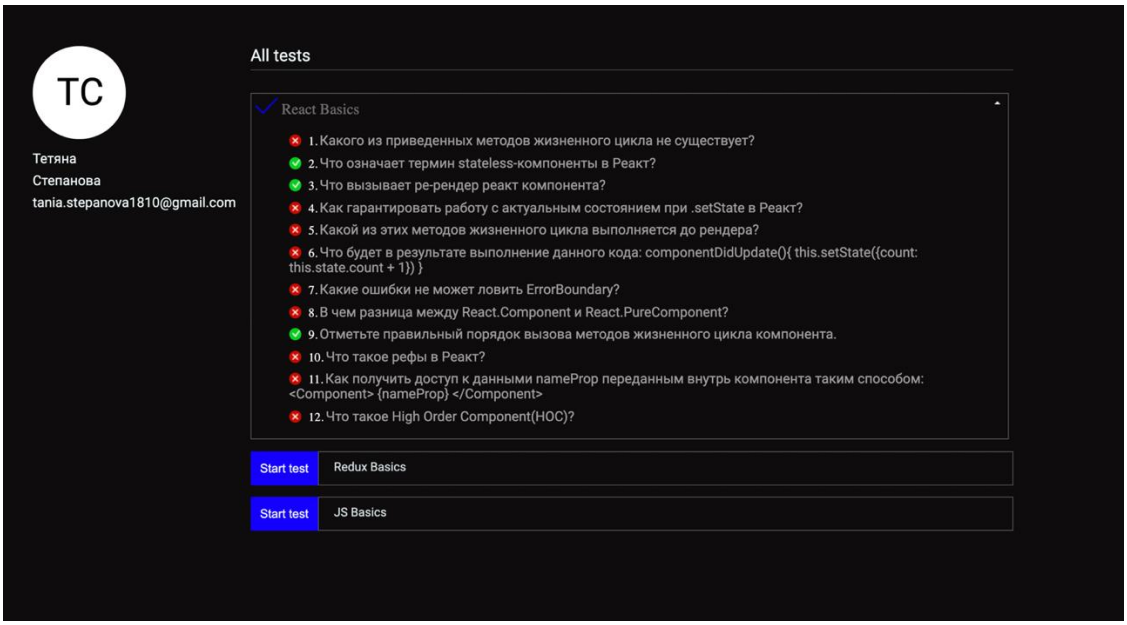


Рисунок 3.18 — Сторінка студента з усіма доступними тестами та їх результи

4 ЕКОНОМІЧНА ЧАСТИНА

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

Магістерська кваліфікаційна робота на тему «Інформаційно-аналітична система оцінювання комплексного рівня підготовки фахівців навчальним закладом» відноситься до науково-технічних робіт, які орієнтовані на виведення на ринок (або рішення про виведення науково-технічної розробки на ринок може бути прийнято у процесі проведення самої роботи), тобто коли відбувається так звана комерціалізація науково-технічної розробки.

Цей напрямок є пріоритетним, оскільки результатами розробки можуть користуватися інші споживачі, отримуючи при цьому певний економічний ефект. Але для цього потрібно знайти потенційного інвестора, який би взявся за реалізацію цього проекту і переконати його в економічній доцільності такого кроку.

Для наведеного випадку нами мають бути виконані такі етапи робіт:

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

— розраховано витрати на здійснення науково-технічної розробки;

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

4.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Інформаційно-аналітична система оцінювання комплексного рівня

підготовки фахівців навчальним закладом» є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в таблиці 4.1 [34].

Таблиця 4.1 — Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

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

Закінчення таблиці 4.1.

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

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці 4.2.

За результатами розрахунків, наведених в таблиці 4.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 4.3.

Таблиця 4.2 — Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	4	5	4
2. Ринкові переваги (наявність аналогів)	3	2	3
3. Ринкові переваги (ціна продукту)	3	4	3
4. Ринкові переваги (технічні властивості)	4	4	4
5. Ринкові переваги (експлуатаційні витрати)	0	0	0
6. Ринкові перспективи (розмір ринку)	3	4	4
7. Ринкові перспективи (конкуренція)	3	3	3
8. Практична здійсненність (наявність фахівців)	4	4	4
9. Практична здійсненність (наявність фінансів)	3	3	2
10. Практична здійсненність	4	4	4
11. Практична здійсненність (термін реалізації)	4	3	4
12. Практична здійсненність (розробка документів)	3	3	3
Сума балів	38	39	38
Середньоарифметична сума балів $СБ_с$	38,3		

Таблиця 4.3 — Науково-технічні рівні та комерційні потенціали розробки

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

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Інформаційно-аналітична система оцінювання комплексного рівня підготовки фахівців навчальним закладом» становить 38,3

бала, що, відповідно до таблиці 4.3, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

4.2 Визначення рівня конкурентоспроможності розробки

В процесі визначення економічної ефективності науково-технічної розробки також доцільно провести прогноз рівня її конкурентоспроможності за сукупністю параметрів, що підлягають оцінюванню.

Одиничний параметричний індекс розраховуємо за формулою:

$$q_i = \frac{P_i}{P_{базі}},$$

де q_i — одиничний параметричний індекс, розрахований за і-м параметром;

P_i — значення і-го параметра виробу;

$P_{базі}$ — аналогічний параметр базового виробу-аналога, з яким проводиться порівняння.

Загальні технічні та економічні характеристики розробки представлено в таблиці 4.4.

Таблиця 4.4 — Основні техніко-економічні показники аналога та розробки, що проектується

Показники (параметри)	Одиниця вимірю-вання	Аналог	Проектований пристрій	Відношення параметрів нової розробки до аналога	Питома вага показника
Кількість аналітичних модулів	шт	4	8	2	0,25
Кількість вибіркових показників аналізу	шт	9	12	1,33	0,12
Швидкість обробки аналітичних даних	мс	4	6	0,67	0,18

Закінчення таблиці 4.4.

Максимальний об'єм аналізованої бази даних	Мв	320	450	1,14	0,15
Доступність інтерфейсу	бал	5	7	1,4	0,3
Експлуатаційні витрати	грн	460	460	1	0,45
Вартість доступу	грн	520	480	0,92	0,55

Нормативні параметри оцінюємо показником, який отримує одне з двох значень: 1 — пристрій відповідає нормам і стандартам; 0 — не відповідає.

Груповий показник конкурентоспроможності за нормативними параметрами розраховуємо як добуток частинних показників за кожним параметром за формулою:

$$I_{НП} = \prod_{i=1}^{n} q_i \ ,$$

де $I_{нп}$ — загальний показник конкурентоспроможності за нормативними параметрами;

q_i — одиничний (частинний) показник за і-м нормативним параметром;

n — кількість нормативних параметрів, які підлягають оцінюванню.

За нормативними параметрами розроблюваний пристрій відповідає вимогам ДСТУ, тому $I_{нп} = 1$.

Значення групового параметричного індексу за технічними параметрами визначаємо з урахуванням вагомості (частки) кожного параметра:

$$I_{ТП} = \sum_{i=1}^{n} q_i \cdot \alpha_i \ ,$$

де $I_{тп}$ — груповий параметричний індекс за технічними показниками (порівняно з виробом-аналогом);

q_i — одиничний параметричний показник і-го параметра;

α_i — вагомість і-го параметричного показника, $\sum_{i=1}^{n}\alpha_i = 1$;

n — кількість технічних параметрів, за якими оцінюється конкурентоспроможність.

Проведемо аналіз параметрів згідно даних таблиці 4.4.

I_{mn} = 2·0,25+1,33·0,12+0,67·0,18+1,14·0,15+1,4·0,3 = 1,37.

Груповий параметричний індекс за економічними параметрами розраховуємо за формулою [34]:

$$I_{ЕП} = \sum_{i=1}^{m} q_i \cdot \beta_i ,$$

де $I_{ЕП}$ — груповий параметричний індекс за економічними показниками;

q_i — економічний параметр і-го виду;

β_i — частка і-го економічного параметра, $\sum_{i=1}^{m}\beta_i = 1$;

m — кількість економічних параметрів, за якими здійснюється оцінювання.

Проведемо аналіз параметрів згідно даних таблиці.

$I_{ЕП}$ = 1 · 0,45 + 0,92 · 0,55 = 0,96.

На основі групових параметричних індексів за нормативними, технічними та економічними показниками розрахуємо інтегральний показник конкурентоспроможності за формулою:

$$K_{ІНТ} = I_{НП} \cdot \frac{I_{ТП}}{I_{ЕП}},$$

$K_{ІНТ}$ = 1 · 1,37 / 0,96 = 1,43.

Інтегральний показник конкурентоспроможності $К_{ІНТ} > 1$, отже розробка переважає відомі аналоги за своїми техніко-економічними показниками.

4.3 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Інформаційно-аналітична система оцінювання комплексного рівня підготовки фахівців навчальним закладом», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

4.3.1 Витрати на оплату праці

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

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

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

де k — кількість посад дослідників залучених до процесу досліджень;

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

t_i — число днів роботи конкретного дослідника, дн.;

T_p — середнє число робочих днів в місяці, T_p=21 дні.

$З_о$ = 12105,00 · 42 / 21 = 24210,00 грн.

Проведені розрахунки зведемо до таблиці 4.5.

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР на тему «Інформаційно-аналітична система

оцінювання комплексного рівня підготовки фахівців навчальним закладом» розраховуємо за формулою:

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

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

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

Таблиця 4.5 — Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	12105,00	576,43	42	24210,00
Інженер-розробник інформаційних систем	11320,00	539,05	21	11320,00
Інженер-розробник програмного забезпечення	11320,00	539,05	15	8085,71
Консультант (викладач ЗВО, д.т.н., професор)	12500,00	595,24	10	5952,38
Технік	7240,00	344,76	21	7240,00
Всього				56808,10

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

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

де M_M — розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо Мм=2379,00 грн;

K_i — коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду;

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

$Т_р$ — середнє число робочих днів в місяці, приблизно $Т_р$ = 21 дн;

$t_{зм}$ — тривалість зміни, год.

$С_1$ = 2379,00 · 1,10 · 1,65 / (21 · 8) = 25,70 грн.

$З_{р1}$ = 25,70 · 8,20 = 210,75 грн.

Таблиця 4.6 — Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Установка обладнання для моделювання та проектування системи	8,20	2	1,10	25,70	210,75
Підготовка робочого місця розробника	4,70	3	1,35	31,54	148,25
Підготовка серверного обладнання	3,20	4	1,50	35,05	112,15
Інсталяція ПЗ для моделювання та розробки інформаційної системи	6,22	4	1,50	35,05	218,00
Компіляція програмних блоків	8,00	5	1,70	39,72	317,77
Налагодження програмних блоків	4,60	4	1,50	35,05	161,22
Тестування системи	10,00	2	1,10	25,70	257,02
Всього					1425,16

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

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

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

$З_{дод}$ = (56808,10 + 1425,16) · 12 / 100% = 6987,99 грн.

4.3.2 Відрахування на соціальні заходи

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

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

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

$Зн$ = (56808,10 + 1425,16 + 6987,99) · 22 / 100% = 14348,67 грн.

4.3.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за темою «Інформаційно-аналітична система оцінювання комплексного рівня підготовки фахівців навчальним закладом».

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

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

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

n — кількість видів матеріалів;

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

$К_j$ — коефіцієнт транспортних витрат, ($К_j$ = 1,1 … 1,15);

B_j — маса відходів j-го найменування, кг;

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

M_1 = 3,00 · 108,00 · 1,1 - 0,000 · 0,00 = 356,40 грн.

Проведені розрахунки зведемо до таблиці 4.7.

Таблиця 4.7 — Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Офісний папір Cristal A4 500	108,00	3,00	0,000	0,00	356,40
Папір для записів Cristal LightPapers 65 A5	55,00	5,00	0,000	0,00	302,50
Органайзер офісний Cristal	175,00	2,00	0,000	0,00	385,00
Набір офісний Cristal Base	203,00	3,00	0,000	0,00	669,90
Картридж для принтера Canon LBP5000	960,00	1,00	0,000	0,00	1056,00
Диск оптичний Vybir CD-R	13,25	3,00	0,000	0,00	43,73
Flesh-пам'ять Kingston 32 GB	234,00	1,00	0,000	0,00	257,40
Всього					3070,93

4.3.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі ($К_в$), які використовують при проведенні НДР на тему «Інформаційно-аналітична система оцінювання комплексного рівня підготовки фахівців навчальним закладом», розраховуємо, згідно з їхньою номенклатурою, за формулою:

$$К_в = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j ,$$

де $Н_j$ — кількість комплектуючих j-го виду, шт.;

$Ц_j$ — покупна ціна комплектуючих j-го виду, грн;

$К_j$ — коефіцієнт транспортних витрат, ($К_j$ = 1,1 … 1,15).

$К_в$ = 1 · 2940,00 · 1,1 = 3234,00 грн.

Проведені розрахунки зведемо до таблиці 4.8.

Таблиця 4.8 — Витрати на комплектуючі

Найменування комплектуючих	Кількість, шт.	Ціна за штуку, грн	Сума, грн
RouterBOARD TP-Link230	1	2940,00	3234,00
База даних (тестувальна) БД64-А10	1	750,00	825,00
Всього			4059,00

4.3.5 Спецустаткування для наукових (експериментальних) робіт

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

Балансову вартість спецустаткування розраховуємо за формулою:

$$B_{спец} = \sum_{i=1}^{k} Ц_i \cdot C_{пр.i} \cdot K_i \ ,$$

де $Ц_i$ — ціна придбання одиниці спецустаткування даного виду, марки, грн;

$C_{пр.i}$ — кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

K_i — коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, ($К_і$ = 1,10…1,12);

k — кількість найменувань устаткування.

$B_{спец}$ = 27860,00 · 1 · 1,1 = 30646,00 грн.

Отримані результати зведемо до таблиці 4.9.

Таблиця 4.9 — Витрати на придбання спецустаткування кожного виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Серверне обладнання на основі EOM DELUX F43-A71BC	1	27860,00	30646,00
Всього			30646,00

4.3.6 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

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

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

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

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

k — кількість найменувань програмних засобів.

$B_{прг}$ = 4760,00 · 1 · 1,1 = 5236,00 грн.

Отримані результати зведемо до таблиці 4.10.

Таблиця 4.10 — Витрати на купівлю програмних засобів кожного виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Емулятор серверу для моделювання поведінки інформаційної системи	1	4760,00	5236,00
Всього			5236,00

4.3.7 Амортизація обладнання, програмних засобів та приміщень

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

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12},$$

де $Ц_б$ — балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ — термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ — строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл}$ = (20460,00 · 2) / (2 · 12) = 1705,00 грн.

Проведені розрахунки зведемо до таблиці 4.11.

Таблиця 4.11 — Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Персональний комп'ютер розробника ПЗ	20460,00	2	2	1705,00
Персональний комп'ютер розробника інформаційних систем	20460,00	2	2	1705,00
Робоче місце інженера-програміста	7000,00	5	2	233,33
Робоче місце аналітика	7000,00	5	2	233,33
Пристрої передачі даних	6800,00	4	2	283,33

Закінчення таблиці 4.11

Оргтехніка	8100,00	4	2	337,50
Приміщення лабораторії розробки ІС	320000,00	25	2	2133,33
ОС Windows 10	5700,00	2	2	475,00
Прикладний пакет Microsoft Office 2016	5100,00	2	2	425,00
Всього				7530,83

4.3.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію ($В_е$) розраховуємо за формулою:

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

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

t_i — тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ — вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_e$ = 4,50 грн;

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

η_i — коефіцієнт корисної дії обладнання, η_i<1.

$В_е$ = 0,45 · 320,0 · 4,50 · 0,95 / 0,97 = 648,00 грн.

Проведені розрахунки зведемо до таблиці 4.12.

Таблиця 4.12 — Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Персональний комп'ютер розробника ПЗ	0,45	320,0	648,00
Персональний комп'ютер розробника інформаційних систем	0,25	200,0	225,00
Робоче місце інженера-програміста	0,12	200,0	108,00
Робоче місце аналітика	0,12	200,0	108,00

Закінчення таблиці 4.12

Пристрої передачі даних	0,05	100,0	22,50
Оргтехніка	0,70	15,0	47,25
Серверне обладнання на основі EOM DELUX F43-A71BC	0,25	200,0	225,00
Всього			1383,75

4.3.9 Службові відрядження

До статті «Службові відрядження» дослідної роботи на тему «Інформаційно-аналітична система оцінювання комплексного рівня підготовки фахівців навчальним закладом» належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів тощо. Витрати за статтею «Службові відрядження» відсутні.

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

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

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

де $H_{сп}$ — норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 30%.

$В_{сп}$ = (56808,10 + 1425,16) · 30 / 100% = 17469,98 грн.

4.3.11 Інші витрати

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

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

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

де $Н_{ів}$ — норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 55%.

$I_в$ = (56808,10 + 1425,16) · 55 / 100% = 32028,29 грн.

4.3.12 Накладні (загальновиробничі) витрати

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

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

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

де $Н_{нзв}$ — норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 105%.

$В_{нзв}$ = (56808,10 + 1425,16) · 105 / 100% = 61144,92 грн.

Витрати на проведення науково-дослідної роботи на тему «Інформаційно-аналітична система оцінювання комплексного рівня підготовки фахівців навчальним закладом» розраховуємо як суму всіх попередніх статей витрат за формулою:

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

$B_{заг}$ = 56808,10 +1425,16 +6987,99 +14348,67386 +3070,93 +4059,00 +30646,00 + 5236,00 + 7530,83 +1383,75 +0,00 +17469,98 +32028,29 +61144,92 = 242139,61 грн.

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

$$ЗВ = \frac{B_{заг}}{\eta},$$

де η — коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η =0,95.

ЗВ = 242139,61 / 0,95 = 254883,80 грн.

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

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

Результати дослідження проведені за темою «Інформаційно-аналітична система оцінювання комплексного рівня підготовки фахівців навчальним закладом» передбачають комерціалізацію протягом 4-х років реалізації на ринку.

В цьому випадку основу майбутнього економічного ефекту будуть формувати:

ΔN — збільшення кількості споживачів яким надається відповідна інформаційна послуга у періоди часу, що аналізуються;

Показник	1-й рік	2-й рік	3-й рік	4-й рік
Збільшення кількості споживачів, осіб	2000	4000	3000	2000

N — кількість споживачів яким надавалась відповідна інформаційна послуга у році до впровадження результатів нової науково-технічної розробки, приймемо 14000 осіб;

$Ц_б$ — вартість послуги у році до впровадження інформаційної системи, приймемо 400,00 грн;

$\pm\Delta Ц_о$ — зміна вартості послуги від впровадження результатів, приймемо 80,00 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 4-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [35]:

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

де λ — коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2021 році ставка податку на додану вартість складає 20%, а коефіцієнт λ=0,8333;

ρ — коефіцієнт, який враховує рентабельність інноваційного продукту). Приймемо ρ=40%;

ϑ — ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2021 році ϑ=18%;

Збільшення чистого прибутку 1-го року:

$\Delta П_1$ = (80,00·14000,00+480,00·2000)·0,83·0,4·(1-0,18/100%) = 566259,20 грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2$ =(80,00·14000,00+480,00·6000)·0,83·0,4·(1-0,18/100%) = 1088960,00 грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3$ =(80,00·14000,00+480,00·9000)·0,83·0,4·(1-0,18/100%) =

1480985,60 грн.

Збільшення чистого прибутку 4-го року:

$\Delta П_4$ =(80,00·14000,00+480,00·11000)·0,83·0,4·(1-0,18/100%) =

1742336,00 грн.

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

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

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

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

τ — ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,14;

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

$ПП$ =566259,20/(1+0,14)^1+1088960,00/(1+0,14)^2+1480985,60/(1+0,14)^3+

+1742336,00/(1+0,14)^4=496718,60+837919,36+999623,10+1031602,78=

3365863,83 грн.

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

$$PV = k_{інв} \cdot ЗВ,$$

де $k_{інв}$ — коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=1,5;

ЗВ — загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 254883,80 грн.

$PV = k_{інв} \cdot ЗВ$ = 1,5 · 254883,80 = 382325,70 грн.

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

$$E_{абс} = ПП - PV,$$

де *ПП* — приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 3365863,83 грн;

PV — теперішня вартість початкових інвестицій, 382325,70 грн.

$E_{абс} = ПП - PV$ = 3365863,83 - 382325,70 = 2983538,13 грн.

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

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

де $E_{абс}$ — абсолютний економічний ефект вкладених інвестицій, 2983538,13 грн;

PV — теперішня вартість початкових інвестицій, 382325,70 грн;

$T_ж$ — життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 4 роки.

$E_в = \sqrt[Тж]{1 + \frac{E_{абс}}{PV}} - 1$ = (1+2983538,13/382325,70)$^{1/4}$= 0,72.

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

$$\tau_{мін} = d + f,$$

де d — середньозважена ставка за депозитними операціями в комерційних банках; в 2021 році в Україні d =0,11;

f — показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,15.

$\tau_{мін}$ = 0,11+0,15 = 0,26 < 0,72 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Інформаційно-аналітична система оцінювання комплексного рівня підготовки фахівців навчальним закладом» доцільно.

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

$$T_{ок} = \frac{1}{E_в},$$

де $E_в$ — внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 0,72 = 1,38 р.

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

ВИСНОВКИ

У магістерській кваліфікаційній роботі було розроблено інформаційно-аналітичну систему оцінювання комплексного рівня підготовки фахівців навчальним закладом.

У першому розділі було проведено аналіз існуючих систем оцінювання та тестування. Виявлено переваги і недоліки в існуючих системах. Основні переваги було враховано у системі.

У другому розділі було розглянуто та проаналізовано методику для реалізації комплексного оцінювання знань студентів. За основу взялись дисципліни зі спеціальності 123 — Комп'ютерна інженерія. Також розроблено струкрутури для викладача та студента Також у цьому розділі було розглянуто, проаналізовано та обрано технології для розробки клієнтської частини додатку та серверної частини. Проведено огляд і аналіз усіх типів баз даних, обрано найоптимальнішу із них.

У третьому розділі проводився власне огляд на розробку частини інформаційно-аналітичної системи. Обрано зручне середовище розробки. Створено базу даних з тестами та студентами. Створено проект для серверної частини системи, а також проект для клієнтської частини програми. Розроблено мінімальний інтерфейс користувача, який дає змогу легко переміщуватись та бачити контент у зручному для ока вигляді.

У четвертому розділі були проведені дослідження та розрахунки доцільності розробки та використання розробленої системи. Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Інформаційно-аналітична система оцінювання комплексного рівня підготовки фахівців навчальним закладом» становить 38,3 бала, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

При оцінюванні рівня конкурентоспроможності, згідно узагальненого коефіцієнту конкурентоспроможності розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 1,43 рази.

Також термін окупності становить 1,38 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Інформаційно-аналітична система оцінювання комплексного рівня підготовки фахівців навчальним закладом».

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Підгорна О. Що таке формувальне оцінювання, чому воно потрібне учням і які основні виклики [Електронний ресурс] / О. Підгорна, Т. Береговська. — 2021. — Режим доступу до ресурсу: https://nus.org.ua/view/shho-take-formuvalne-otsinyuvannya-chomu-vono-potribne-uchnyam-i-yaki-osnovni-vyklyky/

2. Симонова И. Оценка персонала: объективные и субъективные факторы [Електронний ресурс] / И. Симонова, И. Еремина, Л. Дудаева. — 2021. — Режим доступу до ресурсу:https://hrliga.com/index.php

3. Индивидуально-ориентированное тестирование [Електронний ресурс] — Режим доступу до ресурсу: https://studfile.net/preview/2041053

4. Моніторинг // Юридична енциклопедія: [у 6 т.] / ред. кол. Ю. С. Шемшученко — К. : Українська енциклопедія ім. М. П. Бажана, 2001. — Т. 3 : К — М. — 792 с. — ISBN 966-7492-03-6

5. Метод письмового контролю [Електронний ресурс] — Режим доступу до ресурсу: https://pidru4niki.com/1326030734990/pedagogika

6. Шкуропия А. Конструктор тестов easyQuizzy [Електронний ресурс] / А. Шкуропия, И. Полсакова — Режим доступу до ресурсу: http://easyquizzy.ru/.

7. MyTestXPro [Електронний ресурс] — Режим доступу до ресурсу: http://mytest.klyaksa.net/wiki

8. Матусець Т. MyTest [Електронний ресурс] / Тетяна Матусець — Режим доступу до ресурсу: http://matematica.inf.ua/files/program/program_all

9. INDIGO MENTAL TRAINING CLUB [Електронний ресурс] — Режим доступу до ресурсу: https://indigomental.com/company

10. Система тестирования Let's test [Електронний ресурс] — Режим доступу до ресурсу: https://letstest.ru/

11. Баканов М.И., Шеремет А.Д. Теория экономи-ческого анализа: Учебник. М.: Финансы и статистика, 1997. 416 с.

12. Що таке стек в програмуванні [Електронний ресурс] — Режим доступу до ресурсу: https://genomukr.ru/komp-juteri/21264-shho-take-stek-v-programuvanni.html

13. Кривоченко М. 10 JavaScript-фреймворков [Електронний ресурс] / Мария Кривоченко — Режим доступу до ресурсу: https://tproger.ru/articles/10-javascript-frejmvorkov-kotorye-stoit-vyuchit-v-2021-godu/

14. Facebook Inc. React [Електронний ресурс] / Facebook Inc. — Режим доступу до ресурсу: https://ru.reactjs.org/

15. The modern web developer's platform — Angular [Електронний ресурс] — Режим доступу до ресурсу: https://angular.io/

16. Vue.js [Електронний ресурс] — Режим доступу до ресурсу: https://vuejs.org/

17. jQuery [Електронний ресурс] — Режим доступу до ресурсу: https://jquery.com/

18. backbonejs [Електронний ресурс] — Режим доступу до ресурсу: https://backbonejs.org/

19. A framework for ambitious web developers [Електронний ресурс] — Режим доступу до ресурсу: https://emberjs.com/

20. Polymer library [Електронний ресурс] — Режим доступу до ресурсу: https://polymer-library.polymer-project.org/3.0/docs/devguide/feature-overview

21. JavaScript [Електронний ресурс] — Режим доступу до ресурсу: https://developer.mozilla.org/ru/docs/Web/JavaScript

22. Python [Електронний ресурс] — Режим доступу до ресурсу: https://www.python.org/

23. The PHP Group. PHP [Електронний ресурс] / The PHP Group — Режим доступу до ресурсу: https://www.php.net/manual/ru/intro-whatis.php

24. ORACLE. JAVA [Електронний ресурс] / ORACLE — Режим доступу до ресурсу: https://www.java.com/ru/

25. Ruby [Електронний ресурс]. — 2021. — Режим доступу до ресурсу: https://www.ruby-lang.org/ru/

26. Google. Build fast, reliable, and efficient software at scale [Електронний ресурс] / Google — Режим доступу до ресурсу: https://go.dev/

27. Язык C# и платформа .NET [Електронний ресурс]. — 2021. — Режим доступу до ресурсу: https://metanit.com/sharp/tutorial/1.1.php

28. Мирошниченко Е. А. К формальному определению понятия «база данных» // Пробл. информатики. 2011. № 2. С. 83-87

29. БАЗИ ДАНИХ [Електронний ресурс] — Режим доступу до ресурсу: https://ua5.org/database/189-reljacjjna-baza-danikh.html

30. NOSQL meetup Tickets, Thu, Jun 11, 2009 at 10:00 AM. Eventbrite.com. Архів оригіналу за 2017-08-03. Процитовано 2017-03-06

31. Определение документной базы данных [Електронний ресурс] — Режим доступу до ресурсу: https://aws.amazon.com/ru/nosql/document/

32 .Fowler, Adam (February 24, 2015). NoSQL for Dummies. John Wiley & Sons. с. 298–. ISBN 978-1-118-90574-6

33. WebStorm [Електронний ресурс] — Режим доступу до ресурсу: https://www.jetbrains.com/ru-ru/webstorm/

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

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

ДОДАТОК А

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
д.т.н., проф. О. Д. Азаров
"__" _____ 2021 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на виконання магістерської кваліфікаційної роботи на тему:
«Інформаційно-аналітична система оцінювання комплексного рівня підготовки фахівців навчальним закладом»
08-23.МКР.012.00.000 ПЗ

Науковий керівник к.т.н., проф. каф. ОТ
_____ Захарченко С. М.
Студентка групи 1КІ-20м
_____ Степанова Т. М.

Вінниця 2021

1 Підстава для використання МКР

а) актуальність розробки полягає у створенні та вдосконаленні системи оцінювання знань.

б) наказ про затвердження теми кваліфікаційної роботи.

2 Мета і призначення МКР

а) мета роботи — створення інформаційно-аналітичної системи комплексного оцінювання рівня підготовки фіхівців навчальним закладом;

б) призначення розробки — використання системи у навчальних цілях.

3 Джерела розробки МКР

Інтегроване середовище розробки WebStorm та стек сучасних JavaScript технологій MERN.

4 Технічні вимоги до виконання МКР

— наявність ком'ютера з виходом у мережу Internet;

— наявність програми, яка демонструє функції для оцінювання та тестування студента.

5 Етапи МКР та очікувані результати

5.1 Робота виконується за шість етапів, таблиця А.1.

Таблиця А.1 — Етапи виконання роботи

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Огляд і аналіз існуючих систем оцінювання	01.09.2021	01.10.2021	Аналітичний огляд літературних джерел, задачі досліджень
2	Вибір методики реалізації комплексного оцінювання	01.10.2021	10.10.2021	Методика, 2 розділ
3	Програмна реалізація частини системи	11.10.2021	24.10.2021	3 розділ

Закінчення таблиці А.1

4	Підготовка економічної частини	25.10.2021	7.11.2021	4 розділ
5	Опублікування результатів досліджень	18.11.2021	3.12.2021	стаття
6	Оформлення пояснювальної записки, графічного матеріалу і/або презентації	4.12.2021	22.12.2021	пояснювальна записка, графічний матеріал і/або презентація

6 Матеріали, що подаються до захисту МКР

Пояснювальна записка МКР, графічні і ілюстративні матеріали, протокол попереднього захисту МКР на кафедрі, відзив наукового керівника, відзив опонента, протоколи проходження перевірки на плагіат, анотації до МКР українською та іноземною мовами, нормоконтроль про відповідність оформлення МКР діючим вимогам.

7 Порядок контролю виконання та захисту МКР

Виконання етапів графічної та розрахункової документації МКР контролюється науковим керівником згідно зі встановленими термінами. Захист МКР відбувається на засіданні Державної екзаменаційної комісії, затвердженою наказом ректора.

8 Вимоги до оформлення МКР

Оформлення магістерської кваліфікаційної роботи відбувається згідно «Положень про кваліфікаційні роботи» та ДСТУ 3008-2015.

Технічне завдання отримав _____ Степанова Т. М.

ДОДАТОК Б

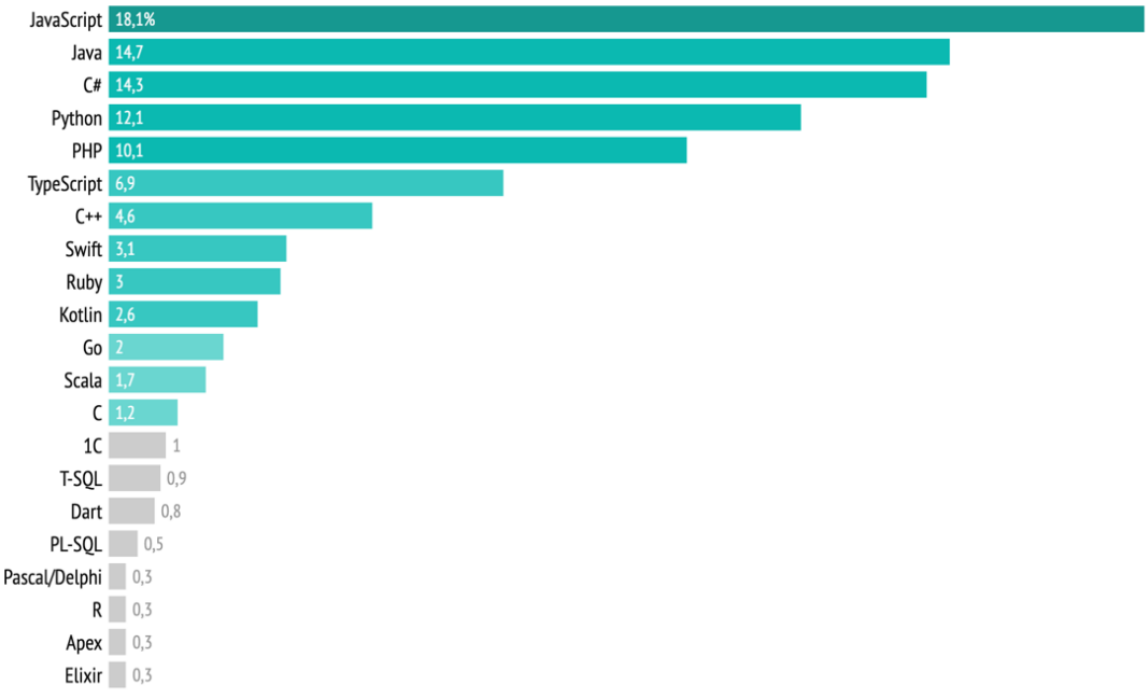
Блок схема роботи викладача з системою

ДОДАТОК В

Блок схема роботи студента з системою

ДОДАТОК Г

Статистика використання різних мов програмування



ДОДАТОК Д

Інформація з детальним оглядом мов програмування

№	Мова	Частка ринку	Зміни	Основна	Додаткова	Свої проєкти	Індекс вподобання
1	JavaScript	18		1201	3168	1537	0.55
2	Java	14.7		978	737	771	0.75
3	C#	14.3		949	479	615	0.83
4	Python	12.1	-1	804	1141	783	0.72
5	PHP	10.1		672	490	469	0.7
6	TypeScript	6.92	2.5	459	1421	593	0.8
7	C++	4.6	-1.3	305	392	373	0.63
8	Swift	3.1		204	155	153	0.85
9	Ruby	3.0	0.6	198	145	154	0.72
10	Kotlin	2.6		174	236	175	0.88
11	Go	2.0		132	313	210	0.87
12	Scala	1.75	0.65	116	133	75	0.80
13	C	1.1		79	292	175	0.37
14	1C	1	-0.7	63	37	38	0.47
15	T-SQL			60	793	65	0.28
16	Dart			55	120	69	0.72
17	PL-SQL			34	355	26	0.32
18	Pascal/Delphi			23	42	27	0.56
19	R			22	84	33	0.5
20	Apex			18	6	4	

ДОДАТОК Е

Код серверної частини

Файл index.js

```
const mongoose =
require('mongoose');
                        const app = require('./app');
                        require('custom-env').env();
                        const start = () => {
                         try {
                          const DB = process.env.DB;
                          mongoose
                           .connect(DB, {
                            useNewUrlParser: true,
                            useCreateIndex: true,
                            useFindAndModify: false,
                            useUnifiedTopology: true,
                           })
                           .then(() => console.log('DB connection successful!'));
                          const port = process.env.PORT || 3001;
                          app.listen(port, () => {
                           console.log('server was started');
                          });
                         } catch (e) {
                          console.log(e);
                         }
                        };
                        start();
```

Файл app.js

```
const express =
require('express');
                    const cors = require('cors');
                    const authRouter = require('./routers/authRouter');
                    const sendInviteRouter = require('./routers/sendInviteRouter');
                    const userRouter = require('./routers/userRouter');
                    const testRouter = require('./routers/testRouter');
                    const middleware = require('./middleware/handleError');
                    const app = express();
                    app.use(cors());
                    app.use(express.urlencoded({ extended: true }));
                    app.use(express.json());
                    app.use(authRouter);
                    app.use('/mail', sendInviteRouter);
                    app.use('/user', userRouter);
```

```
app.use('/test', testRouter);
app.use(middleware.handleError);
module.exports = app;
```

Файл utils/getResult.js

```
module.exports
= {
                async getResult(selected, questions) {
                  let result = [];

                  const varies = questions.questions.map(a => a.options.map(a => a));

                  varies.forEach((a, idx) => {
                    let flag = 0;
                    if (questions.questions[idx].multiply) {
                      const correct = a.filter(({correct}) => correct).map(({_id}) => _id);
                      correct.forEach((id) => {
                        if (selected[idx].selected.includes(id.toString())) flag++;
                      });
                      if (flag === correct.length
                        && selected[idx].selected.length === correct.length) result.push(true);
                      else result.push(false);
                    } else {
                      if (selected[idx].selected.length > 1) {
                        result.push(false);
                        return;
                      }

                      const correct = a.filter(({correct}) => correct).map(({_id}) => _id);
                      correct.forEach(id => {
                        if (selected[idx].selected.includes(id.toString())) {
                          result.push(true);
                        } else result.push(false);
                      })
                    }
                  });

                  return result;
                }
              };
```

Файл utils/securityManager.js

```
const crypto =
require('crypto');
                const jwt = require('jsonwebtoken');
                const md5 = (password) => crypto
```

```
  .createHash('md5')
  .update(password)
  .digest('base64');
const createAccessJWT = (email) => {
  return jwt.sign({email: email}, process.env.SECRET, {
    expiresIn: 60 * 60,
  });
};

module.exports = {
  md5,
  createAccessJWT,
};
```

Файл utils/sendEmail.js

```
const nodemailer =
require('nodemailer');

module.exports = {
  async sendEmail(email, message) {
    const transporter = nodemailer.createTransport({
      service: 'gmail',
      auth: {
        user: '456square123@gmail.com',
        pass: 'vinnitsa123',
      }
    });
    let result = await transporter.sendMail({
      from: '"Node js" <nodejs@example.com>',
      to: email,
      subject: "Magisoft 'Test'",
      text: message,
      html: `<strong>${message}</strong>`
    });
    console.log(result);
  }
};
```

Файл controllers/authController.js

```
const
httpError =
require('http-
errors');
const jwtDecode = require('jwt-decode');
const jwt = require('jsonwebtoken');
const securityManager = require('./../utils/securityManager');
```

```
const User = require('../models/User');
const Test = require('../models/Test');


const register = async (req, res) => {
  const {name, password, token} = req.body;

  const decode = jwtDecode(token);
  const user = await User.findOne({email: decode.email});

  if (!user) throw httpError[400]("user was not invited");
  if (user.activated) throw httpError[400]("user has already exist !");

  const hash = securityManager.md5(password);
  await User.findOneAndUpdate({email: decode.email}, {
    name,
    email: decode.email,
    password: hash,
    activated: true,
  });
  res.status(201).json({token: token, message: 'User was registered
successfully!'});

};
const login = async (req, res) => {
  const {email, password} = req.body;
  const user = await User.findOne({email});
  if (!user) {
    return res.status(401).json('user is not found');
  }

  const hash = securityManager.md5(password);
  const userConfirm = await User.findOne({name: user.name, password: hash});
  // console.log(userConfirm);
  if (!userConfirm) {
    return res.status(401).json('email or password are wrong');
  }

  const token = securityManager.createAccessJWT(email);

  const tests = await Test.find({_id: userConfirm.accessToTest});
  const testsId = userConfirm.results.map(({testId}) => testId);
  const availableTests = tests.filter(({_id}) => !testsId.includes(_id));
  const completedTests = tests.filter(({_id}) => testsId.includes(_id));
```

```
  return res.status(200).json({user: userConfirm, test: availableTests,
completedTests, token});
};
const verifyToken = async (req, res) => {
  const token = req.headers.authorization.split(' ')[1];
  if (!token) {
    return res.status(400).json({
      error: true,
      message: "Token is required."
    });
  }
  jwt.verify(token, process.env.SECRET, async (err, user) => {
    if (err) return res.status(401).json({
      error: true,
      message: "Invalid token."
    });

    const userData = await User.findOne({email: user.email});
    if (!userData) {
      return res.status(401).json({
        error: true,
        message: "Invalid user."
      });
    }
    const testData = await Test.find({_id: userData.accessToTest});
    const testsId = userData.results.map(({testId}) => testId);
    const availableTests = testData.filter(({_id}) => !testsId.includes(_id));
    const completedTests = testData.filter(({_id}) => testsId.includes(_id));

    return res.status(200).json({user: userData, test: availableTests,
completedTests, token});
  });
};

module.exports = {
  register,
  login,
  verifyToken
};
```

Файл controllers/sendInviteController.js

```
const User =
require('../models/User');
                    const utils = require('./../utils/sendEmail');
                    const securityManager = require('./../utils/securityManager');
                    const generateLink = async (req, res) => {
```

```
  const {id} = req.body;
  if (!id) {
    return res.status(401).json({err: 'Invalid email!'});
  }
  const email = await User.find({_id: id});

  const token = await
securityManager.createAccessJWT(email[0].email);

  res.status(201).json({token:
`${process.env.LINK}/register/?=${token}`});
};
const sendInvite = async (req, res) => {
  const {id, link} = req.body;
  if (!id || !link) {
    return res.status(401).json({message: `invite was not sent`});
  }
  const email = await User.find({_id: id});

  await utils.sendEmail(email[0].email, link);
  await User.findOneAndUpdate({email: email[0].email}, {invited:
true});
  res.status(201).json({message: 'Invite was sent!'});
};
const sendEmail = async (req, res) => {
  const {textMessage, id} = req.body;
  if (!textMessage || !id) {
    return res.status(400).json({message: `email was not sent`});
  }
  const email = await User.findOne({_id: id});
  console.log(email.email, textMessage);
  await utils.sendEmail(email.email, textMessage);

  res.status(200).json({message: 'Invite was sent!'});
};

module.exports = {
  generateLink,
  sendInvite,
  sendEmail,
};
```

Файл controllers/testsController.js

```
  const httpError =
  require('http-errors');
        const Test = require('../models/Test');
```

```
const User = require('../models/User');
const jwtDecode = require('jwt-decode');
const utils = require('./../utils/getResult');

const getAllTest = async (req, res) => {
  const tests = await Test.find();

  if (!tests) throw httpError[401](`Something went wrong:
${e.message}`);
  return res.status(200).json({data: tests});
};
const getUserResult = async (req, res) => {
  const {selected, testId, token} = req.body;
  const decode = jwtDecode(token);

  if (!selected || !testId || !token) httpError[400](`Something went
wrong: ${e.message}`);

  const userTest = await Promise.all([
    Test.findOne({_id: testId}),
    User.findOne({email: decode.email}),
  ]);
  const result = await utils.getResult(selected, userTest[0]);
  const userResult = [...userTest[1].results, {results: result, testId:
testId}];
  await User.findOneAndUpdate({email: decode.email}, {results:
userResult});

  return res.status(200).json({result: result});
};
const createNewTest = async (req, res) => {
  const {test} = req.body;

  if (!test) throw httpError[402](`something went wrong
${e.message}`);

  await Test.create({title: test});
  return res.status(201).json({message: 'test was created
successfully'});
};
const deleteTest = async (req, res) => {
  const id = req.params.id;
  if (!id) throw httpError[400]('incorrect data');

  await Test.deleteOne({_id: id});
```

```
  return res.status(202).json({message: 'test was deleted
successfully'});
};
const updateTest = async (req, res) => {
  const id = req.params.id;
  const {testName} = req.body;
  if (!testName || !id) throw httpError[400]('invalid data');
  await Test.findOneAndUpdate({_id: id}, {title: testName});
  return res.status(202).json({message: 'test was deleted
successfully'});
};
const addQuestion = async (req, res) => {
  const {testId, questionName, variant1, variant2, variant3,
variant4} = req.body;
  const multiply = [variant1[1], variant2[1], variant3[1],
variant4[1]].filter(a => a);
  if (!testId || !questionName || !variant1 || !variant2 || !variant3 ||
!variant4) {
    throw httpError[400]('data is not found');
  }
  const questionsList = await Test.findOne({_id: testId});
  const newQuestionList = [...questionsList.questions, {
    options: [
      {title: variant1[0], correct: variant1[1]},
      {title: variant2[0], correct: variant2[1]},
      {title: variant3[0], correct: variant3[1]},
      {title: variant4[0], correct: variant4[1]},
    ],
    title: questionName,
    multiply: multiply.length > 1,
  }];
  await Test.findOneAndUpdate({_id: testId}, {questions:
newQuestionList});
  return res.status(201).json({message: 'question was added
successfully'});
};
const deleteQuestion = async (req, res) => {
  const {testId, questionId} = req.params;
  if (!testId || !questionId) throw httpError[400]('data is not found
!');
  const test = await Test.findOne({_id: testId});
  await Test.findOneAndUpdate({_id: testId}, {
    questions: test.questions.filter(({_id}) => _id.toString() !==
questionId)
  });
```

```
  return res.status(202).json({message: 'question was deleted
successfully'});
};
const updateQuestion = async (req, res) => {
  const {testId, questionId} = req.params;
  const {questionName, variant1, variant2, variant3, variant4} =
req.body;
  const multiply = [variant1[1], variant2[1], variant3[1],
variant4[1]].filter(a => a);
  try {
    const test = await Test.findOne({_id: testId});
    const newQuestions = test.questions.map((a) => {
      if (a._id.toString() === questionId) {
        return {
          correctAns: a.correctAns,
          subject: a.subject,
          _id: a._id,
          options: [
            {title: variant1[0], correct: variant1[1]},
            {title: variant2[0], correct: variant2[1]},
            {title: variant3[0], correct: variant3[1]},
            {title: variant4[0], correct: variant4[1]},
          ],
          multiply: multiply.length > 1,
          title: questionName,
        }
      } else return a;
    });
    await Test.findOneAndUpdate({_id: testId}, {questions:
newQuestions});

    res.status(201).json({message: 'question was updated
successfully'});
  } catch (e) {
    throw httpError[403](`something went wrong ${e.message}`);
  }
};
module.exports = {
    getAllTest,
    getUserResult,
    createNewTest,
    deleteTest,
    updateTest,
    addQuestion,
    deleteQuestion,
    updateQuestion,
```

```
};
```

Файл controllers/userController.js

```
const
httpError =
require('http-
errors');
                const User = require('../models/User');
                const Test = require('../models/Test');
                const jwt = require('jsonwebtoken');

                const addNewUser = async (req, res) => {
                 const {name, surname, email, comment, group, accessToTest} = req.body;
                 await User.create({
                   name,
                   surname,
                   email,
                   comment,
                   group,
                   accessToTest
                 });

                 return res.status(201).json({message: 'User was added successfully!'});
                };
                const getAllUser = async (req, res) => {
                 const filter = req.params.filter;
                 let users = [];
                 // console.log(filter);
                 if (filter === 'NotActivated') users = await User.find({activated: false, invited:
                true});
                 else if (filter === 'NotInvited') users = await User.find({invited: false});
                 else if (filter === 'Activated') users = await User.find({activated: true});
                 else users = await User.find();

                 if (!users.length) throw httpError[404]('users not found !');

                 return res.status(200).json({users});
                };
                const deleteUser = async (req, res) => {
                 const id = req.params.id;
                 await User.deleteOne({_id: id});
                 return res.status(202).json({message: 'user was deleted successfully!'});
                };
                const updateUser = async (req, res) => {
                 const id = req.params.id;
                 const {name, surname, email, comment, group, accessToTest} = req.body;
```

```
  await User.findOneAndUpdate({_id: id}, {name, surname, email, comment,
group, accessToTest});
  return res.status(202).json({message: 'user was updated successfully!'});
};
const getCurrentResult = async (req, res) => {
  const {token} = req.params;

  jwt.verify(token, process.env.SECRET, async (err, user) => {
    if (err) return res.status(404).json({message: 'token is required !'});

    const currentUser = await User.findOne({email: user.email});
    let currentResult = {};
    if (currentUser.results.length) {
      currentResult = await Test.findOne(
          {_id: currentUser.results[currentUser.results.length - 1].testId}
        );
    }
    if (!currentResult) return res.status(200).json({currentUser});
    return res.status(200).json({
      results: currentUser.results[currentUser.results.length - 1].results.map(a => a
=== 'true'),
      testsName: currentResult.title,
      questions: currentResult.questions.map(({title, _id}) => ({title, _id})),
    });
  })
};
const getCurrentTest = async (req, res) => {
  const {testId} = req.params;
  const test = await Test.findOne({_id: testId});
  if (!test) throw httpError[404]('test not found !');
  return res.status(200).json({test});
};
module.exports = {
    addNewUser,
    getAllUser,
    deleteUser,
    updateUser,
    getCurrentResult,
    getCurrentTest
};
```

Файл middleware/autoriz.js

```
const jwt =
require('jsonwebtoken');
                        module.exports = {
                          async verifyToken(req, res, next) {
                            try {
```

```
      const token = req.headers.authorization.split(' ')[1];
      jwt.verify(token, process.env.SECRET, (err, user) => {
       if (err) return res.status(401).json({message: 'token is required'});
       else next();
      });
     } catch (e) {
      return res.status(500).json({message: 'something went wrong !'});
     }
    }
   };
```

Файл middleware/handleError.js

```
const
httpError =
require('http-
errors');
        const catchAsyncError = fn => (req, res, next) => fn(req, res, next).catch(next);
        const routeNotFound = (req, res, next) => next(new httpError[404]('Not Found!
        Wrong api endpoint'));

        const handleError = (err, req, res) => {
         const {message, status, data} = err;
         res.status(status || 500).json({message, status, data});
        };

        module.exports = {
         catchAsyncError,
         routeNotFound,
         handleError,
        };
```

Файл models/Test.js

```
const arr =
require('./tests');
            const mongoose = require('mongoose');
            const testScheme = new mongoose.Schema(
               {
                  title: {
                     type: String,
                     required: [true, 'title must be here'],
                  },
                  description: {
                     type: String,
                     default: 'description must be here'
                  },
                  questions: [
```

```
{
  correctAns: {
    type: [],
    default: [],
  },
  subject: [{
    name: {
      type: String,
      default: ['name must be here']
    },
  }],
  options: [
    {
      title: String,
      correct: {
        type: Boolean,
        default: false
      },
    },
    {
      title: String,
      correct: {
        type: Boolean,
        default: false
      },
    },
    {
      title: String,
      correct: {
        type: Boolean,
        default: false
      },
    },
    {
      title: String,
      correct: {
        type: Boolean,
        default: false
      },
    },
    {
      title: String,
      correct: {
        type: Boolean,
        default: false
      },
    }
  ],
  title: String,
  multiply: {
    type: Boolean,
    default: false
  },
```

```
            code: {
              type: String,
              default: 'code must be here',
            },
          }
        ]
      },
      {
        timestamps: true,
      }
);
const Test = mongoose.model('Test', testScheme);
module.exports = Test;
```

Файл models/User.js

```
const mongoose =
require('mongoose');
        const userSchema = new mongoose.Schema(
          {
            name: String,
            surname: String,
            email: {
              type: String,
              unique: true,
              required: true,
            },
            group: String,
            comment: String,
            password: String,
            invited: {
              type: Boolean,
              default: false,
            },
            activated: {
              type: Boolean,
              default: false,
            },
            accessToTest: {
              type: [String],
              required: true,
            },
            results: [
              {
                results: [String],
                testId: String,
              }
```

```
        ],
    },
    {
        timestamps: true,
    }
);


const User = mongoose.model('User', userSchema);
module.exports = User;
```

ДОДАТОК Ж

Код клієнтської частини

Файл index.js

```
import
React
from
'react';
        import ReactDOM from 'react-dom';
        import { BrowserRouter } from 'react-router-dom';
        import App from './App';
        import * as serviceWorker from './serviceWorker';
        import './index.css';
        ReactDOM.render(
          <BrowserRouter>
            <div className="back">
              <App />
            </div>
          </BrowserRouter>,
          document.getElementById('root'),
        );
```

Файл App.js

```
import
React, {
useState,
useEffect
} from
'react';
            import { Router } from './router';
            import { LoginContext } from './context';
            import './App.css';
            import AdminAPIService from './pages/AdminPage/AdminAPIService';
            import { getToken, removeUserSession } from './utils/common';
            const App = () => {
              const api = new AdminAPIService();
              const [user, setUser] = useState({});
              useEffect(() => {
                const token = getToken();
                if (!token) {
                  return;
                }
                const verifyToken = async () => {
                  const response = await api.verifyToken(token);
```

```
      if (response.status === 401) removeUserSession();
      setUser(response);
    };
    verifyToken();
  }, []);
  return (
    <div className="App">
      <LoginContext.Provider value={{ user, setUser }}>
        <Router />
      </LoginContext.Provider>
    </div>
  );
};
export default App;
```

Файл utils/ComponentVisisble.js

```
import {
useEffect,
useRef,
useState
} from
'react';
const ComponentVisible = initialIsVisible => {
  const [isComponentVisible, setIsComponentVisible] = useState(initialIsVisible);
  const ref = useRef(null);
  const handleHideDropdown = event => {
    if (event.key === 'Escape') {
      setIsComponentVisible(false);
    }
  };
  const handleClickOutside = event => {
    if (ref.current && !ref.current.contains(event.target)) {
      setIsComponentVisible(false);
    }
  };
  useEffect(() => {
    document.addEventListener('keydown', handleHideDropdown, true);
    document.addEventListener('click', handleClickOutside, true);
    return () => {
      document.removeEventListener('keydown', handleHideDropdown, true);
      document.removeEventListener('click', handleClickOutside, true);
    };
  });

  return {
    ref,
```

```
    isComponentVisible,
    setIsComponentVisible,
  };
};

export default ComponentVisible;
```

Файл router/AdminRouter.js

```
import
React
from
'react';
        import { Route, Switch } from 'react-router-dom';
        import AdminPageUsers from '../pages/AdminPage/pages/AdminPageUsers';
        import AdminPageTests from '../pages/AdminPage/pages/AdminPageTests';
        const AdminRouter = () => {
        return (
            <Switch>
              <Route path="/admin-page/users" component={AdminPageUsers}/>
              {/*<Route path="/admin-page/sets" component={AdminPageSets} />*/}
              <Route path="/admin-page/tests" component={AdminPageTests} />
            </Switch>
          );
        };
        export default AdminRouter;
```

Файл router/PrivateRouter.js

```
import
React
from
'react';
        import { Route, Redirect } from 'react-router-dom';
        import { getToken } from '../utils/common';
        function PrivateRouter({ component: Component, ...rest }) {
          return (
            <Route
              {...rest}
              render={(props) => getToken() ? <Component {...props} /> : <Redirect to={{
        pathname: '/login', state: { from: props.location } }} />}
            />
          )
        }

        export default PrivateRouter;
```

Файл router/PublicRouter.js

```
import
React
from
'react';
        import { Route, Redirect } from 'react-router-dom';
        import { getToken } from '../utils/common';
        function PublicRouter({ component: Component, ...rest }) {
         return (
           <Route
             {...rest}
            render={(props) => !getToken() ? <Component {...props} /> : <Redirect to={{
        pathname: '/user-profile/all-test' }} />}
           />
         )
        }
        export default PublicRouter;
```

Файл router/Router.js

```
import
React
from
'react';
        import { Switch, Route } from 'react-router-dom';
        import RegisterForm from 'src/pages/RegisterForm';
        import UserProfile from '../pages/UserProfile';
        import Authorization from 'src/pages/AuthorizationForm';
        import Header from '../components/Header';
        import AdminPage from '../pages/AdminPage';
        import PrivateRouter from './PrivateRouter';
        import PublicRouter from './PublicRouter';


        const Router = () => {
         return (
           <div>
             <Header />
             <Switch>
               <PublicRouter path="/register" component={RegisterForm} />
               <PublicRouter path="/login" component={Authorization} />
               <PrivateRouter path="/user-profile" component={UserProfile} />
               <Route path="/admin-page" component={AdminPage} />
             </Switch>
           </div>
         );
        };
        export default Router;
```

Файл router/index.js

```
import
Router
from
'./Router';
                import AdminRouter from './AdminRouter';
                import UserRouter from './UserRouter';

                export {
                  Router,
                  AdminRouter,
                  UserRouter,
                };
```

Файл components/Accordion/Accordion.js

```
import
React
from
'react';
              import { string, bool } from 'prop-types';
              import styles from './Accordion.module.scss';
              import buttonDown from '../../pages/AdminPage/images/buttonDown.png';
              import activeButton from '../../pages/AdminPage/images/activeButton.png';
              import ComponentVisible from '../../utils';
              const Accordion = ({ children, title, isImg }) => {
                const { ref, isComponentVisible, setIsComponentVisible } = ComponentVisible(false);
                const handleClick = e => {
                  if (e.target.tagName === document.getElementById('accordion').tagName) {
                    setIsComponentVisible(!isComponentVisible);
                  }
                };

                return (
                  <div className={styles.root} ref={ref}>
                    <p
                      role="presentation"
                      id="accordion"
                      className={`${styles.title} ${isComponentVisible ? styles.active : null}`}
                      onClick={handleClick}
                    >
                      {title}
                      {isImg ? (
                        <img
                          className={styles.img}
                          src={isComponentVisible ? activeButton : buttonDown}
```

```
            alt="buttonDown"
          />
        ) : null}
        <div className={styles.accordionChildren}>{isComponentVisible &&
children}</div>
      </p>
    </div>
  );
};
Accordion.propTypes = {
  isImg: bool,
  // list: element,
  title: string,
};
Accordion.defaultProps = {
  isImg: bool,
  title: string,
  // list: array,
};

export default Accordion;
```

Файл components/Checkbox/Checkbox.js

```
import
React
from
'react';
import { func, number, oneOfType, string } from 'prop-types';
import './Checkbox.scss';
const Checkbox = ({ id, onChange }) => {
  const handlerChange = e => {
    if (typeof onChange === 'function') {
      onChange(e);
    }
  };

  return (
    <div className="checkbox">
      <input
        type="checkbox"
        id={id}
        className="checkbox__input"
        onClick={handlerChange}
        onKeyPress={handlerChange}
      />
```

```
        <label htmlFor={id} className="checkbox__label" />
      </div>
    );
  };

  Checkbox.propTypes = {
    id: oneOfType([number, string]).isRequired,
    onChange: func,
  };
  Checkbox.defaultProps = {
    onChange: null,
  };
  export default Checkbox;
```

Файл components/ModalDeleteUser/ModalDeleteUser.js

```
import Button from '../Button';
import AdminAPIService from
'../../pages/AdminPage/AdminAPIService';
const ModalDeleteUser = ({ titleModal, userId, testId,
questionId, setIsModal }) => {
  const api = new AdminAPIService();
  const handleSubmit = async () => {
    if (userId) await api.deleteUser(userId);
    if (testId && !questionId) await api.deleteTest(testId);
    if (testId && questionId) await
api.deleteQuestion(testId, questionId);
  };
  return (
    <div className={styles.container}>
      <p className={styles.delete}>{titleModal}</p>
      <form className={styles.containerButtons}>
        <Button
          type="submit"
          className="secondary"
          size="modalDeleteButtonSize"
          onClick={handleSubmit}
        >
          Удалить
        </Button>
```

```
        <Button type="button" className="primary"
size="modalCancelButtonSize" onClick={setIsModal}>
          Отменить
        </Button>
      </form>
    </div>
  );
};
export default ModalDeleteUser;
```

ДОДАТОК И

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи:___Інформаційно-аналітична система комплексного оцінювання рівня підготовки фахівців навчальним закладом_____.

Тип роботи:_____магістерська кваліфікаційна робота _

(кваліфікаційна роботи, курсовий проект (робота), реферат, аналітичний огляд, інше (зазначити))

Підрозділ _____кафедра обчислювальної техніки, 1КІ-20м

(кафедра, факультет (інститут), навчальна група)

Науковий керівник ____ Захарченко С.М., проф. кафедри ОТ_____

(прізвище, ініціали, посада)

Показники звіту подібності

Plagiat.pl (StrikePlagiarism)		Unicheck	
КП1			
КП2		Оригінальність	83,2
Тривога/Білі знаки	/	Схожість	16,8

Аналіз звіту подібності (відмінити подібне)

- ✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.
- □ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності і відсутності самостійності її автора. Робот направити на доопрацювання.
- □ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлений(-на) з повним звітом подібності, який був згенерований Системою щодо роботи (додається)

Автор _____ Степанова Т.М.

Опис прийнятого рішення

____Ступінь оригінальності роботи відповідає вимогам, що висуваються до МКР

_____

_____

_____

Особа, відповідальна за перевірку_____ _Захарченко С.М.__

Експерт _____ _____