

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

**Приладобудівний факультет
Кафедра приладів і систем орієнтації і навігації**

«На правах рукопису»
УДК _____

До захисту допущено:
Завідувач кафедри
_____ Надія БУРАУ
«__» _____ 20__ р.

Магістерська дисертація

на здобуття ступеня магістра

**за освітньо-професійною програмою «Комп'ютерно - інтегровані технології
та системи навігації і керування»**

зі спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології»

на тему: «Автоматизована система оцінки знань студентів»

Виконав:

студент VI курсу, групи ПГ-91мп

Кобзар Владислав Віталійович _____

Науковий керівник:

Доцент каф. ПСОН, к.т.н., доц.,

Цибульник Сергій Олексійович _____

Консультант з розділу розробки стартап проекту:

Доцент кафедри менеджменту, д.е.н., доц.

Бояринова Катерина Олександрівна _____

Рецензент:

Засвідчую, що у цій магістерській
дисертації немає запозичень з праць
інших авторів без відповідних посилань.

Студент _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Приладобудівний факультет

Кафедра приладів і систем орієнтації і навігації

Рівень вищої освіти – другий (магістерський)

Спеціальність (спеціалізація) – 151 «Автоматизація та комп'ютерно-інтегровані технології» («Комп'ютерно-інтегровані технології та системи навігації і керування»)

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Н.І. Бурау

«__» _____ 2020 р.

ЗАВДАННЯ

на магістерську дисертацію студенту

Кобзарю Владиславу Віталійовичу

1. Тема дисертації «Автоматизована система оцінки знань студентів», науковий керівник дисертації Цибульник Сергій Олексійович, к.т.н., доцент, затверджені наказом по університету від «__» _____ 20__ р. № _____
2. Термін подання студентом дисертації: 07 грудня 2020 року
3. Об'єкт дослідження: процес оцінки знань студентів.
4. Предмет дослідження: ідентифікація студента під час дистанційного виконання контрольних заходів за допомогою автоматизованої системи оцінки знань.
5. Перелік завдань, які потрібно розробити:
 - 5.1 Провести огляд стану проблеми.
 - 5.2 Провести огляд новітніх рішень у сфері оцінки знань студентів.
 - 5.3 Провести огляд існуючих засобів для розробки автоматизованої системи знань студентів.
 - 5.4 Спроекувати архітектуру автоматизованої системи.

5.5 Розробити основні модулі та підсистеми автоматизованої системи оцінки знань студентів.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: таблиці, графіки, рисунки

7. Орієнтовний перелік публікацій: 1 стаття

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розроблення стартап-проекту	Доцент кафедри менеджменту, д.е.н., доц. Бояринова К.О.		

9. Дата видачі завдання 15 вересня 2020 року

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Провести огляд стану проблеми	05.10.2020	
2.	Провести огляд новітніх рішень у сфері оцінки знань студентів	19.10.2020	
3.	Провести огляд існуючих засобів для розробки автоматизованої системи знань студентів	26.10.2020	
4.	Спроекувати архітектуру автоматизованої системи	09.11.2020	
5.	Розробити основні модулі та підсистеми автоматизованої системи оцінки знань студентів	23.11.2020	
6.	Оформлення рукопису дисертації	07.12.2020	

Студент

В.В. Кобзар

Науковий керівник дисертації

С.О. Цибульник

РЕФЕРАТ

Магістерська дисертація складається з 112 сторінок, в ній міститься 36 рисунків, 22 таблиці, використано 29 джерел.

Актуальність на сьогоднішній день актуальною залишається проблема оцінки знань студентів та підтвердження достовірності знань на дистанційній формі навчання. Використання системи оцінки знань студентів з поліпшеною системою підтвердження особистості студента дозволить додатково підтвердити набуті знання під час дистанційного навчання студентів.

Мета магістерської дисертації є створення автоматизованої системи оцінки знань студентів з поліпшеною системою підтвердження особистості.

Завдання:

1. Провести огляд стану проблеми.
2. Провести огляд новітніх рішень у сфері оцінки знань студентів.
3. Провести огляд існуючих засобів для розробки автоматизованої системи знань студентів.
4. Спроекувати архітектуру автоматизованої системи.
5. Розробити основні модулі та підсистеми автоматизованої системи оцінки знань студентів.

Об'єкт дослідження: процес оцінки знань студентів.

Предмет дослідження: ідентифікація студента під час дистанційного виконання контрольних заходів за допомогою автоматизованої системи оцінки знань.

Наукова новизна: вперше розроблено поліпшену систему підтвердження особистості студентів під час проходження тестових завдань у дистанційному режимі за допомогою автоматизованої системи оцінки знань студентів.

Практичне значення: спроектовано та розроблено автоматизовану систему оцінки знань студентів з поліпшеною системою підтвердження особистості, що дозволить адміністрації навчального закладу контролювати якість знань студентів.

Публікації:

3. Кобзар В.В. Автоматизована система оцінки знань студентів / В.В. Кобзар // Молодіжна наукова ліга, Модернізація та сучасні українські та світові наукові дослідження. – 2020. С 10-12.

Ключові слова: Автоматизована система оцінки знань студентів, оцінка знань, тестові завдання, контрольні заходи, система, ідентифікація особистості

ABSTRACT

The master's dissertation consists of 112 pages, it contains 36 figures, 22 tables, 29 sources are used.

The problem of assessing students knowledge and confirming the validity of distance learning remains relevant today. The use of a system of assessment of students' knowledge with an improved system of confirmation of the student's identity will allow to additionally confirm the acquired knowledge during distance learning of students.

The purpose of the master's dissertation is to create an automated system for assessing students' knowledge with an improved system of identity verification.

Task:

1. Review the status of the problem.
2. To review the latest solutions in the field of student knowledge assessment.
3. Review the existing tools for developing an automated system of student knowledge.
4. Design the architecture of the automated system.
5. To develop the main modules and subsystems of the automated system of assessment of students' knowledge.

Object of research: the process of assessing students' knowledge.

Subject of research: identification of the student during remote performance of control actions by means of the automated system of an estimation of knowledge.

Scientific novelty: for the first time an improved system of confirmation of students identity during passing test tasks in the remote mode with the help of an automated system of assessment of students knowledge.

Practical significance: an automated system for assessing students 'knowledge with an improved system of identity verification has been designed and developed,

which will allow the administration of the educational institution to control the quality of students' knowledge.

Publications:

3. Kobzar V.V. Automated system for assessing students' knowledge / V.V. Kobzar // Youth Scientific League, Modernization and Modern Ukrainian and World Scientific Research. – 2020. C 10-12.

Keywords: Automated system of assessment of students' knowledge, assessment of knowledge, test tasks, control measures, system, identity identification

ЗМІСТ

РЕФЕРАТ	4
ABSTRACT	6
РОЗДІЛ 1. ОГЛЯД СТАНУ ПРОБЛЕМИ ТА ВИБІР НАПРЯМКУ ДОСЛІДЖЕННЯ.....	10
1.1 Проблема та актуальність дистанційного оцінювання знань Студентів.....	10
1.2 Огляд існуючих систем оцінки знань	14
1.2.1 Kahoot	14
1.2.2 Moodle	15
1.2.3 Google Classroom	15
1.2.4 iSpring Suite	16
1.2.5 Blackboard Learn	17
1.2.6 Adobe Connect	18
1.2.7 Plickers	18
1.2.8 WEB-Тезаурус	20
1.3 Інструменти створення веб-додатків	21
РОЗДІЛ 2. АНАЛІЗ ТА ФОРМАЛІЗАЦІЯ ВИМОГ ДО РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	27
2.1 Опис моделі життєвого циклу програмного забезпечення	27
2.2 Проектування та детальний опис архітектури	34
2.3 Вибір інструменту для реалізації ПЗ	37
Висновок до розділу 2	37
РОЗДІЛ 3. ДЕТАЛЬНИЙ ОПИС ПРОЦЕСУ РЕАЛІЗАЦІЇ	39
3.1 Процес створення та налаштування серверу і бази даних	39
3.2 Реалізація функціоналу для користувачів системи	44
3.3 Процес створення тестового завдання та його проходження	82
Висновок до розділу 3	87
РОЗДІЛ 4. РОЗРОБКА СТАРТАП ПРОЕКТУ «АВТОМАТИЗОВАНА СИСТЕМА ОЦІНКИ ЗНАНЬ СТУДЕНТІВ».....	88
4.1 Опис ідеї проекту	88

4.2	Технологічний аудит ідеї проекту	90
4.3	Аналіз ринкових можливостей запуску стартап-проекту	91
4.4	Розроблення ринкової стратегії та маркетингової програми проекту.....	99
	Висновок до розділу 4	106
	ВИСНОВКИ.....	108
	СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	110

РОЗДІЛ 1

ОГЛЯД СТАНУ ПРОБЛЕМИ ТА ВИБІР НАПРЯМКУ

ДОСЛІДЖЕННЯ

1.1 Проблема та актуальність дистанційного оцінювання знань студентів

В Українському законодавстві під дистанційним навчанням розуміється індивідуалізований процес набуття знань, умінь, навичок і способів пізнавальної діяльності людини, який відбувається в основному за опосередкованої взаємодії віддалених один від одного учасників навчального процесу у спеціалізованому середовищі, яке функціонує на базі сучасних психолого-педагогічних та інформаційно-комунікаційних технологій [1].

Проблема якісного контролю знань і підведення підсумків результатів оцінювання дистанційно до сих пір залишається актуальною, тому останнім часом в систему освіти впроваджуються автоматизовані системи оцінювання знань, що в певній мірі надають змогу отримувати освіту та контролювати якість отриманих знань дистанційно.

До основних функцій процесу підготовки фахівців з вищою освітою належать створення, передавання та поширення знань, що дозволяє сформувати у студентів вміння й усвідомлення навчатися протягом усього життя.

На різних етапах дистанційного навчання використовуються різні види контролю [1]: попередній, поточний і підсумковий. Попередній контроль спрямований на виявлення знань, умінь і навичок, компетенцій студентів з усіх розділів предмета, який буде вивчатися. Результати вхідного тестування дають можливість викладачеві-тьютору спланувати спільну роботу, діагностувати прогалини в знаннях, визначити, на які теми слід виділити більше часу, тобто сформувати індивідуальну освітню траєкторію студента.

Поточний контроль здійснюється в повсякденній роботі з метою перевірки засвоєння матеріалу [1]. Поточний контроль успішності здійснюється методом самоконтролю і за допомогою тестування студентів та фіксування результатів, а також у процесі спілкування викладача-тьютора та студента. Цей вид контролю

зазвичай співвідноситься із завершенням тем курсу або модулів. Такий контроль повинен обов'язково передувати переходу до нової теми, модулю, новому виду навчальної діяльності. Його підсумки викладач–тьютор обов'язково повинен враховувати у своїй подальшій роботі. Поточний контроль у дистанційному навчанні має важливий дидактичний сенс [1], оскільки дає можливість адекватно оцінювати навчальні результати і вчасно коригувати помилки і прогалини в знаннях студента.

Поточний і попередній контроль полягає у виконанні різних контрольних робіт та перевірці викладачем правильності роботи виконаної студентом у формі рефератів, презентацій, глосаріїв, вирішення завдань, курсових робіт. В якості підсумкового контролю з усіх дисциплін введений залік або іспит. Саме в процесі виконання контрольних завдань поточного і попереднього контролю вивчається максимальна частка нового матеріалу [1], тому система завдань і вибір форм контролю відіграє величезну роль при підготовці майбутніх спеціалістів. Сукупність завдань для практичного виконання забезпечує цілеспрямованість, різноманітність, взаємний зв'язок, наступність і поступове ускладнення робіт.

Підсумковий контроль проводиться в кінці вивчення курсу. Його завдання [1] – визначити досягнутий рівень підготовки студента. Під педагогічними методами контролю якості навчальної діяльності ми розуміємо способи діяльності викладача-тьютора та студентів, за допомогою яких визначається результативність навчально-пізнавальної діяльності, виявляється рівень засвоєння навчального матеріалу. Розглянемо різні методи контролю, що можуть бути застосовані в дистанційному навчанні [1].

Одним з досягнень сучасної методики оцінювання при реалізації рейтингових систем вважається тестовий контроль [2], який допомагає більш чітко прослідковувати структуру знань студентів і на підставі цього переоцінити методичні підходи до вивчення дисципліни, індивідуалізувати процес навчання, сприяти самостійній роботі. На відміну від інших завдань, тест є науково-емпіричним методом дослідження, дозволяють перебороти умоглядні оцінки знань студентів і відрізняються своєю технологічністю.

Тестування застосовується в техніці, медицині, психіатрії, освіті для визначення придатності об'єкта тестування для виконання тих чи інших функцій. Якість тестування і достовірність його результатів значною мірою залежить від методів тестування та складу тестів.

Процес тестування включає [2]:

- подачу тестового набору;
- визначення реакції об'єкта тестування на тестовий набір;
- оцінка реакції об'єкта і висновки.

Тестовий набір складається з окремих тестів і розробляється таким чином, щоб забезпечити повне або значне покриття множини ймовірних впливів на об'єкт тестування [2].

У педагогічній діагностиці отримали поширення методи тестування, що не погіршують якості отриманих об'єктом тестувань знань. Ця специфіка пов'язана з тим, що процес тестування є частиною навчального процесу і під час тестування студент не повинен отримувати або закріплювати хибних знань.

Тестування приваблює тим, що [3]:

- ставить усіх студентів в однакові умови;
- може використовуватися, як для формального оцінювання, так і для самоконтролю під час вивчення певної теми;
- виключає суб'єктивне ставлення викладачів;
- можливість автоматизації обробки результатів і практично миттєвий зворотний зв'язок між студентом та навчальним матеріалом;
- містить елементи гри, що сприяє зменшенню рівня стресу.

З позиції кібернетики [4] процес навчання повинен бути процесом систематичного управління, що ґрунтується на інформації про хід засвоєння нових знань, що і є зворотнім зв'язком системи управління навчальним процесом.

Для проведення оперативного поточного контролю при дистанційному навчанні дуже зручно використовувати різноманітні анкети. Анкета є достатньо гнучким інструментом, оскільки питання можна задавати безліччю різних

способів. У дистанційній освіті після освоєння кожної теми рекомендовано використовувати анкети, в яких студент може зробити самооцінку своїх навчальних досягнень за такими показниками: зрозумів / зрозуміла, можу вирішити самостійно; зрозумів / зрозуміла можу вирішити з підказкою; не зрозумів / не зрозуміла, не можу вирішити. Анкета виконує два завдання [1]. Перше – самооцінка студентом своїх навчальних досягнень у відповідності до змісту матеріалу. Друге – співвіднесення самооцінки студента і його реальних результатів викладачем–тьютором. На основі отриманих даних викладачу–тьютору потрібно провести коригувальні заходи. Самоконтроль є одним з найважливіших факторів, що забезпечують самостійну пізнавальну діяльність студентів. Самоконтроль є форма діяльності, що виявляється в перевірці поставленого завдання, у критичній оцінці процесу роботи, у виправленні її недоліків [1].

У системі дистанційного навчання використовуються одночасно як самоконтроль, так і регламентний контроль. Самоконтроль здійснюється за допомогою такого прийому як рішення тестів з пройдених тем. Даний прийом використовується в якості проміжного контролю успішності студентів, активне використання якого допомагає підтримувати потрібний освітній рівень студентів. Виконання поточного, попереднього та підсумкового контролю є формами регламентного контролю.

Усі перераховані форми організації контролю навчальної діяльності та методи перевірки добре реалізуються, як показала практика [5], в умовах рейтингової системи оцінки знань, так як вона враховує результати поточного, попереднього та підсумкового контролю, що підсумовуються, утворюючи рейтинг студента.

Останні дослідження щодо реалізації e-learning проектів у межах системи самоосвіти України та західних країн показали [6], що в майбутньому необхідно, по-перше, звернути увагу на ефективне в дидактичному аспекті використання інформаційних технологій на базі системного підходу, а саме: застосування програмного забезпечення, створеного з урахуванням навчального змісту та орієнтованого на цілі навчання, останні досягнення педагогіки та психології.

Одним із ключових чинників у самоосвіті з використанням e-learning виступає мотивація до навчання. Саме тому перед вищими навчальними закладами постає важливе завдання створення навчального середовища, яке б спонукало студентів до зацікавленості в самостійному отриманні нових знань.

1.2 Огляд існуючих систем оцінки знань

На даний момент існує велика кількість програм та систем для оцінки знань, розглянемо деякі з них.

1.2.1. Kahoot

Kahoot! [7] представляє собою ігрову навчальну платформу, яка використовується в якості освітньої технології в школах та інших навчальних закладах. Навчальні ігри «Kahoots» є вікториною з безліччю виборів, які дозволяють створювати нових користувачів та, які доступні через веб-браузер.

«Kahoot!» може використовуватися для перевірки знань учнів, для формативного оцінювання [8] або в якості перерви в класних заняттях. Kahoot! також включає в себе різноманітні вікторини.

Kahoot! зробив нову функцію в 2016 році, під назвою Jumble [7]. Користувачі створюють вікторини з чотирма варіантами (їх завжди рівно чотири) в правильному порядку. Kahoot автоматично рандомізують відповіді під час гри. Користувачі повинні перемістити трикутник, ромб, коло і квадрат верх і в правильному порядку. Потім користувачі повинні натиснути синю круглу кнопку з текстом «К!» під прямокутниками. Якщо користувач підозрює, що відповіді знаходяться в неправильному порядку, то прямокутники повинні бути переміщені вниз у вихідне положення, перш ніж він може бути переміщений назад в правильне положення [7].

1.2.2. Moodle

Moodle (акронім від Modular Object-Oriented Dynamic Learning Environment — модульне об'єктно-орієнтоване динамічне навчальне середовище) [9] — навчальна платформа призначена для об'єднання педагогів, адміністраторів і учнів (студентів) в одну надійну, безпечну та інтегровану систему для створення персоналізованого навчального середовища.

Moodle — це безкоштовна, відкрита (Open Source) система управління навчанням. Вона реалізує філософію «педагогіки соціального конструктивізму» та орієнтована насамперед на організацію взаємодії між викладачем та учнями, хоча підходить і для організації традиційних дистанційних курсів, а також підтримки очного навчання.

Moodle перекладена на десятки мов, у тому числі і на українську. Система використовувалась у 2014 році — у 197, у 2019 р — 229 країнах світу, понад 90 тисяч офіційно зареєстрованих сайтів що працюють на Moodle [8].

1.2.3 Google Classroom

Google Classroom [10] — безкоштовний веб-сервіс створений Google для навчальних закладів з метою спрощення створення, поширення і класифікації завдань безпаперовим шляхом. Основна мета сервісу — прискорити процес поширення файлів між педагогами та здобувачами освіти. Може використовуватися вчителями та учнями у школах, або у закладах вищої освіти викладачами та студентами.

Google Classroom об'єднує в собі: Google Drive для створення і обміну завданнями, Google Docs, Sheets and Slides для їх написання, Gmail для спілкування і Google Calendar для розкладу. Здобувачі освіти можуть бути запрошені до класу через приватний код, чи автоматично імпортуватися з шкільного сайту. Кожен клас створює окрему папку на Google диску відповідного користувача Google Drive, куди подається робота, котру оцінює викладач. Мобільні додатки, доступні на iOS і Android, дозволяють

користувачам робити фото та прикріпляти їх до завдань, ділитися файлами з інших додатків та мати офлайн-доступ до інформації. Педагог може відстежувати прогрес кожного здобувача освіти, а після оцінки його роботи, повернути її, супроводжуючи коментарями [10].

1.2.4 iSpring Suite

iSpring Suite [11] - це інструментарій для розробки на основі PowerPoint, розроблений iSpring Solutions, який дозволяє користувачам створювати курси, вікторини, симуляції діалогів, скрінкасти, відеолекції і інші інтерактивні навчальні матеріали. Вихідні курси публікуються в форматі HTML5.

До складу iSpring Suite входять кілька автономних інструментів, які можна використовувати як окремо, так і разом: [11]

iSpring Converter Pro - перетворює презентації PowerPoint в інтерактивні курси, зберігаючи при цьому мультимедійні дані, ефекти PowerPoint, анімацію, тригер і переходи після перетворення.

iSpring QuizMaker - редактор тестів і опитувань, дозволяє працювати з аудіо, відео та зображеннями.

iSpring Cam Pro - програма для запису скрінкасти і створення відеотренінгов, дозволяє монтувати відео з різних доріжок, додавати звук і доповнювати відеоряд текстом, або графікою.

iSpring TalkMaster - симулятор діалогів для створення розгалужених сценаріїв розмов з персонажами і звуковими коментарями.

iSpring Flip - редактор електронних книг.

Бібліотека контенту - вбудована колекція готових шаблонів курсів, персонажів, локацій, значків і елементів управління.

Переваги й недоліки

Переваги - простота у використанні, сумісність з PowerPoint, велика бібліотека контенту, естетичний дизайн шаблонів, можливість використання курсів на мобільних пристроях.

Недоліки - продукт не працює в операційній системі Mac OS, власникам пристроїв виробництва Apple доводиться встановлювати операційну систему Windows. Також відзначається нестача можливостей по створенню технічно складних ефектів.

1.2.5 Blackboard Learn

Blackboard Learn (раніше Blackboard Learning Management System) [12] - це віртуальне середовище навчання і система управління навчанням, розроблена Blackboard Inc. Це веб-серверне програмне забезпечення з керуванням курсами, що має відкриту архітектуру, яка дозволяє інтегруватися з інформаційними системами учнів, і протоколами аутентифікації. Середовище може бути встановлене на локальних серверах або розміщене на Blackboard ASP Solutions, основними цілями є додавання онлайн-елементів до курсів, які проводяться вічна-віч, і розробка повністю онлайн-курсів з мінімальною кількістю зустрічей або без них.

Blackboard Learn надає користувачам платформу для спілкування та обміну контентом.

Функціональні можливості Blackboard Learn [12]:

- Публікація оголошень для студентів;
- Можливість спілкування в чаті в режимі реального часу з іншими учнями в своєму класі;
- Створення дискусій між викладачами та учнями;
- Можливість електронної розсилки листів студентам на їхню пошту;
- Зміст курсу дозволяє вчителям публікувати статті, завдання, відео і т. д.;
- Можливість створення модульних курсів та надавати доступ різним класам;
- Розміщення оцінок вчителями на дошці для загального перегляду студентами

- Публікація відео та інших мультимедійних даних для загального доступу.

1.2.6 Adobe Connect

Adobe Acrobat Connect [13] - програмне забезпечення для веб-конференцій, яке дозволяє окремим особам і малим підприємствам миттєво спілкуватися і співпрацювати через простий у використанні онлайн-доступ. Adobe Acrobat Connect є частиною сімейства Adobe і складається з набору модулів:

Adobe Connect Pro Meeting [13] - Засіб організації нарад і семінарів по мережі в реальному часі. Дозволяє користувачам проводити презентації, обмінюватися файлами, потоковим аудіо, відео, а також служить засобом для організації багатокористувацьких відеоконференцій. Наявна можливість зберігати вже створені віртуальні переговорні кімнати і їх вміст для подальшого швидкого доступу до них - така можливість значно скорочує час підготовки до семінарів, переговорів і проведення презентацій.

Adobe Connect Pro Training [13] - Засіб, що дозволяє створювати, управляти, проводити і відстежувати курси дистанційного навчання. Дозволяє розробляти навчальні програми, які можуть поєднувати в собі як індивідуальні навчальні плани на основі курсів, створених за допомогою Adobe Presenter, так і матеріали сторонніх виробників, а також інтерактивне навчання під керівництвом викладача.

Adobe Connect Pro Events [13] - Засіб управління життєвим циклом всіх подій, що відносяться до участі у зустрічах та тренінгах, таких як оцінка учнів, реєстрація на курси, повідомлення та звітність.

1.2.7 Plickers

Plickers - це додаток для оцінки знань студентів прямо на уроці. Провести опитування цілого класу можна буквально за півхвилини. Все що вам потрібно -

це роздруковані листочки (рис 1.1, [14]) для кожного учня в класі і свій телефон або планшет (учням він не потрібен) [14].

Принцип роботи програми полягає в наступному:

- встановіть додаток Plickers на свій телефон;
- роздайте учням роздруковані картки;
- задайте запитання і попросіть їх відповісти за допомогою листочків (на них будуть варіанти a, b, c, d);
- відскануйте варіанти відповідей всіх учнів.

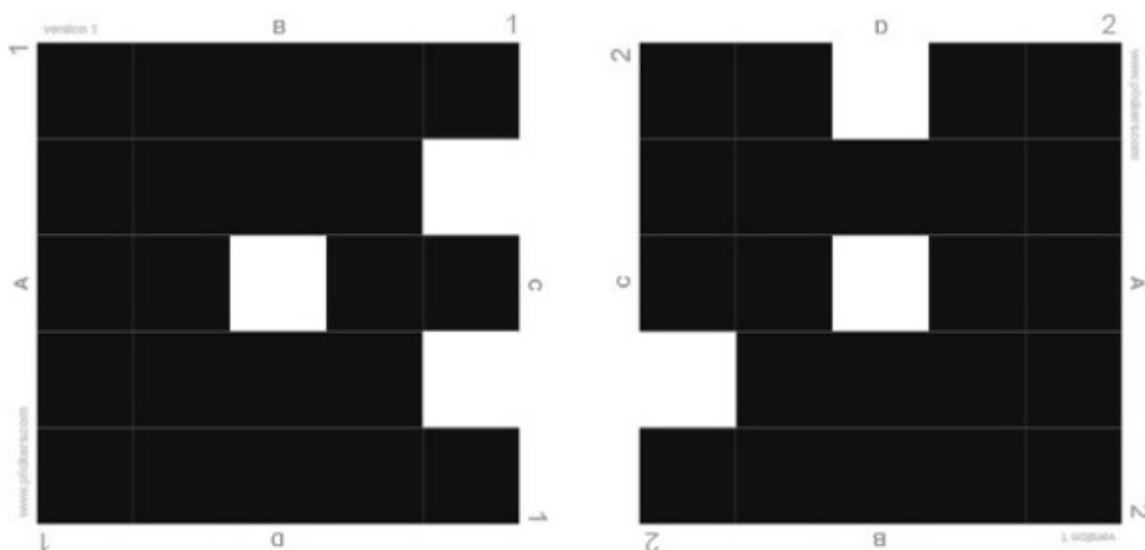


Рисунок 1.1 - Вигляд карток для учнів

Плюси даного додатку:

- Повна залученість класу;
- Неможливість виправити відповідь після завершення опитування певного питання.

Мінуси:

- Можливість повторного зчитування відповіді якщо код випадково потрапляє в камеру ще раз.
- Додаток впливає на навантаженість девайсу, якщо провести більше 1 уроку підряд, необхідно періодично звільняти пам'ять телефону;
- Зчитування інформації займає більше часу, чим прийняття рішення учнем.

1.2.8 WEB-Тезаурус

Система "WEB-Тезаурус" є клієнт-серверним розвитком програми контролю знань "Тезаурус". Її головною особливістю є використання WEB-технологій. Завдяки цьому з'явилася можливість виконувати тестування і самонавчання учнів через інтернет [15].

Характеристики та переваги [15]:

- Можливість вільного використання системи "WEB-Тезаурус" ВНАУ іншими навчальними закладами.
- Можливість безкоштовної публікації власних тестів в системі "WEB-Тезаурус" ВНАУ.
- Тестування і самонавчання здійснюється через інтернет-канали зв'язку.
- Загальна доступність тестової бази для інтернет-спільноти (Виключення складають екзаменаційні тести з пароллями).
- Тестування і самопідготовка здійснюється незалежно від географії і місця розташування комп'ютера.
- "WEB-Тезаурус" - це ефективний інструмент для використання в шкільній, стаціонарній або заочній формах навчання.
- "WEB-Тезаурус" - це високоефективний компонент систем дистанційного навчання.
- Відсутність власне тестової програми. Доступ до сервісу надає WEB-браузер.
- Абсолютна незалежність від типу операційної системи: працювати можна навіть з мобільного телефону.
- Стабільність роботи навіть на поганих каналах зв'язку.
- Повна функціональна сумісність з програмою "Тезаурус".
- Зручність при створенні тестів. Вони як і раніше створюються в програмі "Тезаурус".
- Автоматична робота з електронною системою контролю знань (інтранет-ВНАУ) Вінницького державного аграрного університету.

Як працює "WEB-Тезаурус"?

- "WEB-Тезаурус" працює за технологією LAMP (Linux-Apache-MySQL-PHP).
- Всі тести для "WEB-Тезаурус" створюються звичайною програмою "Тезаурус" за загальними правилами.
- Zip-файли тестів конвертуються в спеціальний формат і публікуються на сервері ВНАУ.
- На цьому ж сервері розташована програма "WEB-Тезаурус", робота з якою здійснюється за допомогою звичайного WEB-браузера.
- Всі опубліковані тести (крім запаролених) є загальнодоступними і можуть вільно використовуватися будь-ким.

1.3 Інструменти створення веб-додатків

Варто звернути увагу на те, що на даний момент стрімко зростає кількість користувачів інтернетом, які при виході в мережу надають перевагу мобільним пристроям, чому сприяє збільшення швидкості мобільного інтернету [16]. Можна припустити, що дана закономірність буде присутня і в найближчому майбутньому. Тому при розробці додатку для автоматизованої системи оцінки знань студентів варто звернути увагу на вибір технологій, які дозволять створити надійний додаток для користувачів персональних комп'ютерів, або ноутбуків та для користувачів мобільних пристроїв.

Варто почати з того, що в сучасному світі важливу роль відіграють веб-сайти, які мають можливість надавати доступ до своїх сервісів за допомогою веб-браузерів без потреби встановлення сторонніх програм.

При розробці варто звернути увагу на швидкість завантаження веб-додатку на мобільних пристроях. Для цього можна розглянути впровадження Progressive Web Apps (далі PWA) або Accelerated Mobile Pages (далі AMP), які є унікальними технологіями, що скорочують час завантаження веб-сторінок.

PWA - це продвинуті сайти, які можуть в деяких випадках замінити нативні додатки. Користувач відкриває в мобільному браузері сторінку, але отримує дещо, що схоже на мобільний додаток в плані користувацького досвіду. При

першому відвідуванні сайту в кеш браузера зберігається все, що потрібно для більш швидких повторних запусках сайту [17].

Особливості PWA

- Швидко та легко встановлюється;
- Займає на телефоні мало місця;
- Швидко відгукується на дії користувача;
- Відкривається майже на всіх пристроях;
- 5. Має деякі можливості нативних додатків (push-сповіщення, створення вікна на головному екрані мобільного телефона, можливість офлайн роботи).

AMP - це пришвидшені сторінки будь-якого сайту. Стандарти створення таких сторінок розвиваються в рамках опенсорсного проекту, запущеного Google в 2015 році. Вони не містять важких фотографій та склановиконуваного JavaScript, тобто нічого, що б заважало завантажуватися контенту першочергової важливості [17].

Особливості AMP:

- Зберігається в кеші пошукових систем, а не на сервері власника сайту;
- Картинки не завантажуються до тих пір, поки користувач не прокрутить сторінку до них;
- Добре відкривається на будь-яких пристроях.

Стандарт AMP не підійде, якщо потрібно берегти динамічні елементи контенту. Наприклад, якщо сайт розважальний і його особливість в інтерактиві, то відобразити AMP буде неможливо. Якщо користувач повинен отримувати push-сповіщення і тд., то AMP також не підійде. В такому випадку варто використовувати PWA, або нативний додаток.

Стек технологій для розробки веб-додатків

При розробці важливо обрати найновіші перевірені технології. Таким чином можна гарантувати, що розроблене програмне забезпечення відповідатиме сучасним стандартам розробки, а також тривалий час матиме можливість легкої зміни функціональних можливостей. Отже, зробимо огляд на популярні технології веб-розробки.

Технології Back-end

Node або **Node.js** - програмна платформа, що базується на двигуні V8 (транслює JavaScript в машинний код), що перетворює JavaScript з вузькоспеціалізованою мови в мову загального призначення. Node.js додає можливість JavaScript взаємодіяти з пристроями введення-виведення через свій API (написаний на C ++), підключати інші зовнішні бібліотеки, написані на різних мовах, забезпечуючи виклики до них з JavaScript-коду. Node.js застосовується переважно на сервері, виконуючи роль веб-сервера, але є можливість розробляти на Node.js і десктопні віконні додатки (за допомогою NW.js, AppJS або Electron для Linux, Windows і macOS) і навіть програмувати мікроконтролери (наприклад, tessel, low.js і espruino). В основі Node.js лежить подієво-орієнтоване і асинхронне (або реактивне) програмування з неблокуючим введенням / висновком [18], логотип платформи має наступний вигляд (рис1.2, [18]).

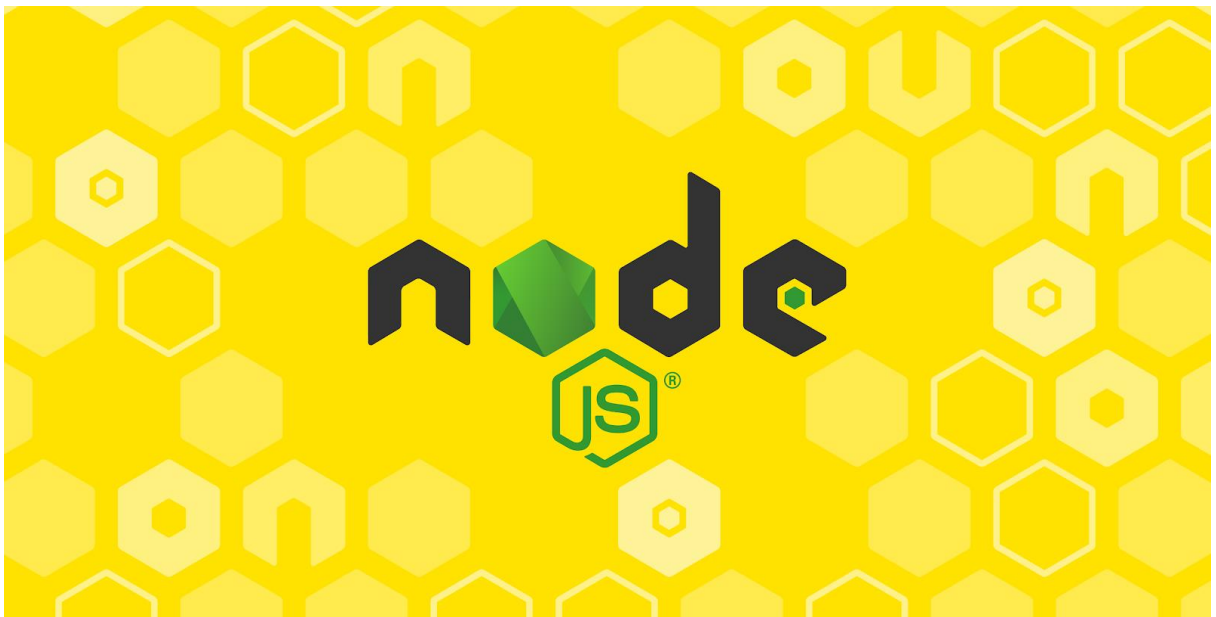


Рисунок 1.2 - Логотип Node.js

Django - вільний фреймворк для веб-додатків на мові Python, що використовує шаблон проектування MVC. Проект підтримується організацією Django Software Foundation (рис 1.3, [19]).

Також, на відміну від інших фреймворків, обробники URL в Django конфігуруються явно за допомогою регулярних виразів.

Для роботи з базою даних Django використовує власний ORM, в якому модель даних описується класами Python, і по ній генерується схема бази даних.



Рисунок 1.3 - Логотип Django

Laravel - безкоштовний веб-фреймворк з відкритим кодом, призначений для розробки з використанням архітектурної моделі MVC (англ. Model View Controller). Laravel випущений під ліцензією MIT (рис 1.4, [20]).



Рисунок 1.4 - Логотип Laravel

Технології Front-end

Angular - це фреймворк Model-View-Controller. Три окремих компонента дозволяють писати добре структурований і простий в підтримці код. Двоспрямована прив'язка даних зручна для простих додатків - будь-які зміни в моделі будуть негайно впроваджені в уявлення і навпаки. Однак якщо ви працюєте над складним проектом, односпрямована прив'язка даних заощадить ваш час і ресурси.

Для максимально ефективного використання Angular доведеться використовувати Typescript.

Стек MEAN - є одним з найпопулярніших. Він включає в себе:

MongoDB - база даних;

Express.js - веб-фреймворк;

Angular - фронтенд фреймворк (рис 1.5, [21]);

Node.js - бекенд.

В описаному стеці Angular часто замінюється на React - бібліотеку Javascript. Стек MERN є відносно молодим, але зростаюча популярність React сприяла швидкому зростанню його популярності (рис 1.6,[22]).

React перевершує за своїми можливостями Angular завдяки віртуальному DOM, який дозволяє швидше і простіше вносити зміни. Однак, оскільки React є бібліотекою, а не фреймворком, що обмежує основні функціональні можливості, розробникам доводиться працювати зі сторонніми сервісами.



Рисунок 1.5 - Логотип Angular

Також варто згадати, що React використовує JSX - модифікацію JavaScript, яка забезпечує безшовну сумісність компонентів.

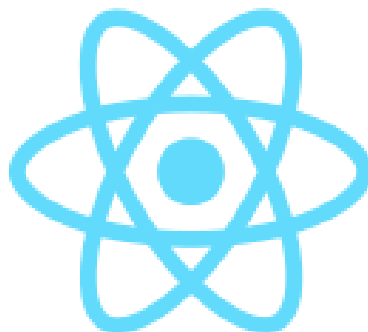


Рисунок 1.6 - Логотип React

Vue.js є більш молодим JS фреймворком, але за останні кілька років він продемонстрував неймовірне зростання популярності. Частково це пов'язано з тим, що це полегшене рішення. У порівнянні з монолітом, подібним Angular, він пропонує тільки базові функції «з коробки». Використовуючи сторонні сервіси, ця функціональність розширюється. В результаті немає необхідності обробляти надлишковий код, як у випадку з Angular (рис 1.7,[23]).

Vue.js також використовується разом з MongoDB, Express.js і Node.js як частина стека MEVN.



Рисунок 1.7 - Логотип Vue

Беручи до уваги розглянуті інструменти оцінювання набутих знань студентів та учнів варто відміти про такі переваги, як інтерактивність, елементи гри та різноманітний вибір можливих варіантів створення тестових завдань, проте в усіх інструментах відсутній засіб візуальної фіксації студентів та учнів.

Для вирішення цієї задачі в даній роботі запропоновано розробити Автоматизовану систему оцінки знань студентів з поліпшеною системою ідентифікації студентів та учнів під час проходження тестових заходів.

РОЗДІЛ 2

АНАЛІЗ ТА ФОРМАЛІЗАЦІЯ ВИМОГ ДО РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Опис моделі життєвого циклу програмного забезпечення(далі ПЗ)

Життєвий цикл ПЗ – це стадії, що проходить програмний продукт від появи ідеї до її реалізації в коді, імплементації у бізнес і подальшої підтримки [24].

Життєвий цикл ПЗ складається з наступних етапів:

- Аналіз вимог
- Проектування
- Програмування
- Тестування
- Налагодження
- Експлуатацію, супровід та підтримку ПЗ

Розглянемо найбільш популярні моделі життєвого циклу програмного забезпечення.

1. Каскадна модель
2. Ітераційна модель
3. Інкрементна модель
4. Спиральна модель

За великим рахунком всі моделі можна розділити на дві великі групи: послідовні та ітераційні моделі.

Waterfall (каскадна модель)

Основна суть моделі Waterfall у тому, що етапи залежать один від одного і наступний починається, коли завершений попередній, утворюючи таким чином поступальний (каскадний) рух уперед [24].

Паралелізм етапів у каскадній моделі, хоч і обмежений, але можливий для абсолютно незалежних між собою робіт. При цьому інтеграція паралельних частин все одно відбувається на якомусь наступному етапі, а не в рамках одного.

Команди різних етапів між собою не комунікують, кожна команда відповідає чітко за свій етап.

Недоліками цієї моделі є отримання результату по проходженню всіх етапів і складність виявлення помилок. Повертатися назад важко. Не зрозуміло що повертати: якщо стався збій на якомусь етапі, його наслідки видно тільки в кінці.

Для замовників дана модель виглядає лінійно і з боку досить просто: за завершеним етапом проектування слідує програмування, а потім тестування - і так крок за кроком поки не буде досягнута фінальна точка і мета, заради якої ведеться розробка.

Дана модель зрозуміло і чисто вкладається в документи, наприклад в договори і роадмапи при наявності чітко визначених контрольних точок. У будь-який момент можна легко зрозуміти чи була пройдена та чи інша точка контролю чи ні, і чи дотримані терміни. З цих причин довготривалі і особливо великі проекти, розраховані на десятиліття і залучення великої кількості організацій-учасників, керуються переважно waterfall [24].

Однак уявлення про простоту каскадної моделі є ілюзорним. Воно з'являється через обмежене бачення клієнтом всього процесу, адже дана модель не має на увазі залучення замовника в деталі процесів розробки і демонструє зрозумілий і кінцевий результат роботи тільки на контрольних точках і в кінці проекту.

У реальності каскадну модель не можна назвати простою, на практиці нею складно керувати. Внесення замовником значних змін під час процесу розробки по waterfall або спрацьовування серйозних, не передбачених проектом ризиків несуть руйнівний характер для всього процесу - модель доводиться перебудовувати, графіки перепланувати [24].

Ітераційна модель передбачає розбиття проекту на частини (етапи, ітерації) і проходження етапів життєвого циклу на кожному з них. Кожен етап є закінченим сам по собі, сукупність етапів формує кінцевий результат (рис 2.1, [24]).

Пояснимо розбиття на етапи на побутовому прикладі. Припустимо, нам потрібен стіл у вітальню [24].

На першому етапі ми зробить стільницю, ніжки і скріпимо їх так, щоб стіл стояв.

На другому, зміцнимо і пофарбуємо його.

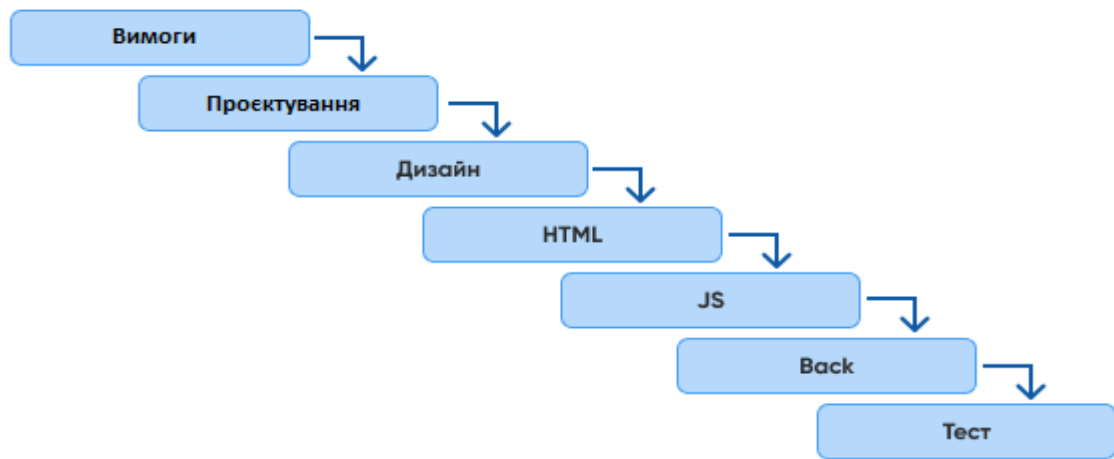


Рисунок 2.1 - Каскадна модель

А на третьому, накриємо скатертиною і купимо до нього стільці.

На кожній ітерації ми працювали з одним і тим же продуктом і в кінці кожної ітерації отримували результат, яким можна користуватися (звісно з певними обмеженнями).

З кожним етапом розробка наближається до кінцевого бажаного результату або уточнюються вимоги до результату по ходу розробки, і відповідно в будь-який момент поточна ітерація може виявитися останньою або черговою на шляху до завершення.

Даний підхід дозволяє боротися з невизначеністю, знімаючи її етап за етапом, і перевіряти правильність технічного, маркетингового або будь-якого іншого рішення на ранніх стадіях.

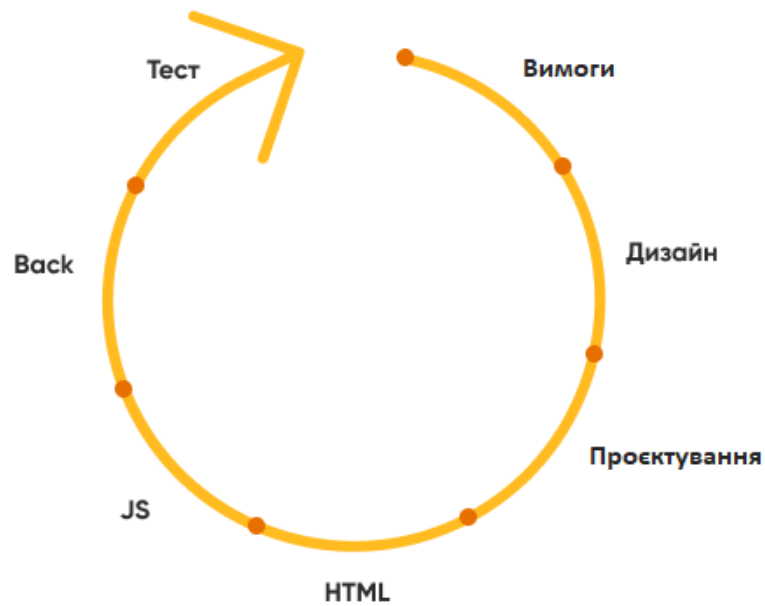


Рисунок 2.2 – Ітераційна модель [24]

Спіральна модель

Усі етапи життєвого циклу при спіральній моделі йдуть витками, на кожному з яких відбуваються проєктування, кодування, дизайн, тестування і т. д. Такий процес відображає суть назви: піднімаючись, проходиться один виток (цикл) спіралі для досягнення кінцевого результату. Причому не обов'язково, що один і той же набір процесів буде повторяться від витка до витка. Але результати кожного з витків ведуть до головної мети (рис. 2.3, [24]).

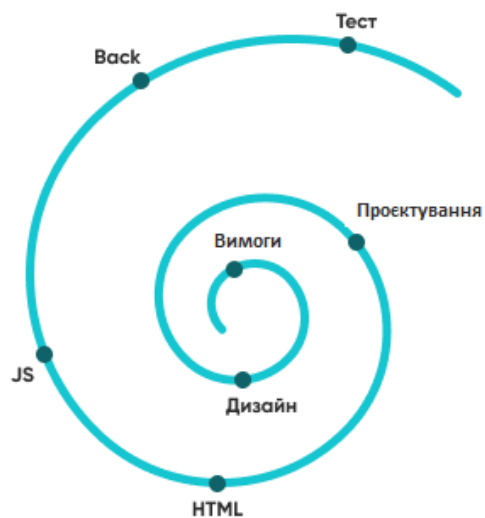


Рисунок 2.3 – Спіральна модель

Інкрементна модель

Принцип, що лежить в основі інкрементної моделі, полягає в розширенні можливостей, добудовування модулів і функцій програми. Буквальний переклад слова інкремент: «збільшення на один». Це «збільшення на один» застосовується в тому числі для позначення версій продукту.

Якщо в каскадній моделі по суті є двома стани продукту: «нічого» і «готовий продукт», то з появою ітераційних моделей стало застосовуватися версіонування продукту. Кожна ітерація позначається цифрою: 1,2,3 і відповідно продукт після кожної ітерації має версію з відповідним номером: v.1, v.2, v.3. Числами після слова версія позначають масштабні зміни в ядро продукту.

А коли одна з версій експлуатується, наступна, з огляду на недоліки попередньої, тільки планується або вже розробляється, а поліпшення замовнику і користувачеві хочеться доставити прямо зараз, тоді з'являються мінорні версії. Туди потрапляють зміни, що не впливають на ядро розробки і представлені як під-версії 1.1,1.2,1.3 або релізи 1.1.1, 1.1.2 і т.п.

У сукупності такі поетапні релізи призводять до повноцінної версії 2.0.

Agile і Lean: принципи розробки ПЗ

Agile - набір принципів гнучкої розробки (всього їх 12) та ідей. Основні ідеї Agile [24]:

- люди і взаємодія важливіші за процеси та інструменти;
- працюючий продукт важливіший за вичерпну документацію;
- співпраця з замовником важливіше узгодження умов контракту;
- готовність до змін важ
- ливіше проходження попереднім планом.

Один з принципів - взаємодія - має на увазі, що замовник взаємодіє з командою, команда з замовником - усі між собою. Це дозволяє обмінюватися досвідом між учасниками команди і клієнтом і кожному з них впливати на прийняття рішень. За рахунок такого підходу знижуються ризики втрати часу і грошей і підвищується здатність команди вирішувати складні нестандартні завдання з високим ступенем невизначеності.

Однак взаємодії всіх і з усіма можуть вилитися у хаос, що впливає на всі сфери розробки. Тому використовуючи Agile потрібно розуміти обмеження: команди повинні бути невеликі, учасники повинні бути компетентні та мотивовані, ітерації короткі з максимально зрозумілими цілями, встановлені чіткі обмеження за часом і кінцевий результат повинен бути очевидним.

Agile чудово справляється з невизначеністю, зумовлюючи майбутнє на більш короткий період. Правило таке: чим вища невизначеність - тим коротша ітерація, причому її тривалість може бути навіть у рамках 24 годин, як і відбувається на хакатоні. На початку кожної ітерації неминуче виконується контроль, ретроспектива, оцінка та аналіз результатів, планування наступної ітерації.

У внутрішньому плануванні та у продуктивній розробці без цього принципу й елементів Agile не обійтися.

Lean

Ідея підходу **Lean** полягає в тому, що ми ощадливо ставимося до ресурсів (у тому числі часу) і вирішуємо завдання найпростішим способом. Наприклад: ми не робимо весь продукт, щоб зрозуміти, що він нікому не потрібен - ми робимо лендінг і форму підписки і даємо на нього рекламу, щоб перевірити, що цей продукт викликає інтерес клієнтів і прийняти усвідомлене рішення про необхідність розробки [24].

Коли доходить до розробки продукту, або робиться якесь поліпшення, виробниче або інженерне, ми спочатку робимо його MVP (minimum viable product). Термін MVP зараз широко поширений і застосовується повсюдно, але він народився саме з Lean підходу. MVP - така версія продукту, що виконує свою головну функцію і при цьому її не відхиляють клієнти і визнають її корисність. Звичайно, її можна покращувати і покращувати, але загалом продукт на стадії MVP повинен бути корисним, зрозумілим клієнтові і таким, щоб можна було прийняти рішення про його подальших поліпшень або визнати експеримент невдалим і тестувати нову гіпотезу, витративши при цьому якомога менше ресурсів [24].

Загалом Lean підхід орієнтується на тестування потреб і цінностей і потрапляння в очікування ринку мінімальними засобами. Lean-підхід не про

"технічні" проблеми і їхні рішення інженерними засобами, а про підприємницькому підході до вирішення завдань. Lean передбачає, що команда шукає найпростіше рішення для досягнення результату: технічно, організаційно і т.п., спрощуючи все, що не є дійсно важливим [24].

У спрощенні сила і головна "пастка" Lean: прагнення все спростити іноді призводить до ситуацій, в яких продукт спрощують настільки, що губляться дійсно важливі функції і продукт по суті виявляється непотрібним, оскільки не несе цінності користувачеві.

Наостанок про Lean хочеться додати: часто, коли говорять про Lean, кажуть, що "Lean — про те, щоб замість сервісу зробити лендінг із формою контактів". Ні, це не Lean. Таке тестування не суперечить Lean, але Lean-методологія пропонує набагато більше прийомів, ніж цей, і варта детального вивчення.

Основні методи розробки ПЗ: гнучкі методології

Нижче наведено короткий огляд основних гнучких методологій розробки з описом їхньої суті. Огляд не претендує на повноту, але дає загальне уявлення, що взагалі існує.

Scrum методологія ґрунтується на понятті спринту (sprint), протягом якого виконується робота над продуктом. Перед початком кожного спринту проводиться планування (Sprint Planning), на якому проводиться оцінка вмісту списку завдань із розвитку продукту (Product Backlog) і формування беклога на спринт (Sprint Backlog), у рамках яких і діє команда. Для спринту завжди існують обмеження по часу, зазвичай від тижня до місяця. Життя продукту таким чином розбита на рівні по тривалості спринти [24].

За **Kanban** методологією проект ділиться на етапи, що візуалізуються у вигляді канбан-дошки. Етапи, це наприклад: планування, розробка, тестування, реліз і т.п. Завдання у вигляді "карток" переміщуються з етапу на етап. Нові завдання можна додавати у будь-який час. Завдання закривається не по закінченню конкретного часу, а по зміні статусу на "завершено". Kanban — методологія з концепції "ощадливого виробництва".

RUP (Rational Unified Process) - розробка продукту за даним методом складається з чотирьох фаз (початкова стадія, уточнення, побудова,

впровадження), кожна з яких включає в себе одну або декілька ітерацій. RUP величезна методологія, котру важко описати одним абзацом тексту, але методи, рекомендовані RUP засновані на статистиці комерційно успішних проектів [24].

DSDM (Dynamic Systems Development Model) - методологія, що демонструє набір принципів, визначених типів ролей і технік.

Принципи спрямовані на головну мету - здати готовий проект вчасно і вкластися у бюджет, з можливістю регулювати вимоги під час розробки. DSDM належить до гнучких методологій розробки програмного забезпечення, а також розробок, що не входять у сферу інформаційних технологій [24].

RAD (Rapid Application Development) - методологія швидкої розробки додатків, що передбачає застосування інструментальних засобів візуального моделювання (прототипування) і розробки. RAD передбачає невеликі команди розробки, терміни до 4 місяців й активне залучення замовника з ранніх етапів. Дана методологія спирається на вимоги, але також існує можливість їхніх змін під час розробки системи. Такий підхід дозволяє скоротити витрати і звести час розробки до мінімуму [24].

XP (Extreme Programming) - методологія, орієнтована на постійну зміну вимог до продукт, пропонує 12 підходів для досягнення ефективних результатів у подібних умовах. Серед них:

- Швидкий план і його постійна зміна
- Простий дизайн архітектури;
- Часте тестування;
- Участь одночасно двох розробників в одному завданні або навіть за одним робочим місцем;
- Постійна інтеграція і часті невеликі релізи.

2.2 Проектування та детальний опис архітектури

Метою роботи є створення веб додатку, який би дозволив контролювати рівень знань студентів за допомогою проходження тестування з можливістю

підтвердження особистості студента під час процесу проходження тестових завдань.

Цільова аудиторія, на яку розрахований сайт – це навчальні заклади, яким необхідно контролювати знання студентів, учні під час дистанційної форми навчання.

Тип сайту: Single Page Application (далі SPA) з адміністративною панеллю та різними рівнями доступу для користувачів.

Мова сайту: українська.

Для розробки системи було виділено наступний функціонал:

1. Реєстрація адміністрації через початкове вікно реєстрації
2. Створення нових дисциплін адміністрацією
3. Створення нових викладачів адміністрацією
4. Створення нових студентів адміністрацією
5. Призначення для певних дисциплін груп студентів та викладачів адміністрацією
6. Створення тестових завдань викладачами для дисциплін, що були назначені адміністрацією
7. Проходження студентами тестових завдань, з предметів, що їм були назначені
8. Оскільки предметом дослідження є ідентифікація студента під час проходження контрольних заходів, необхідно реалізувати функціонал підтвердження особистості студента під час проходження тестового завдання шляхом знімків, які робляться через веб камеру та завантаження їх на сервер.

Архітектура сайту

При створенні архітектури сайту варто звернути увагу, що сайт має 3 рівні доступу, а саме адміністрація, викладачі та студенти. Кожний тип користувача після авторизації отримує персональний набір функціональних можливостей. Структуру сайту зображено на рис. 2.4.

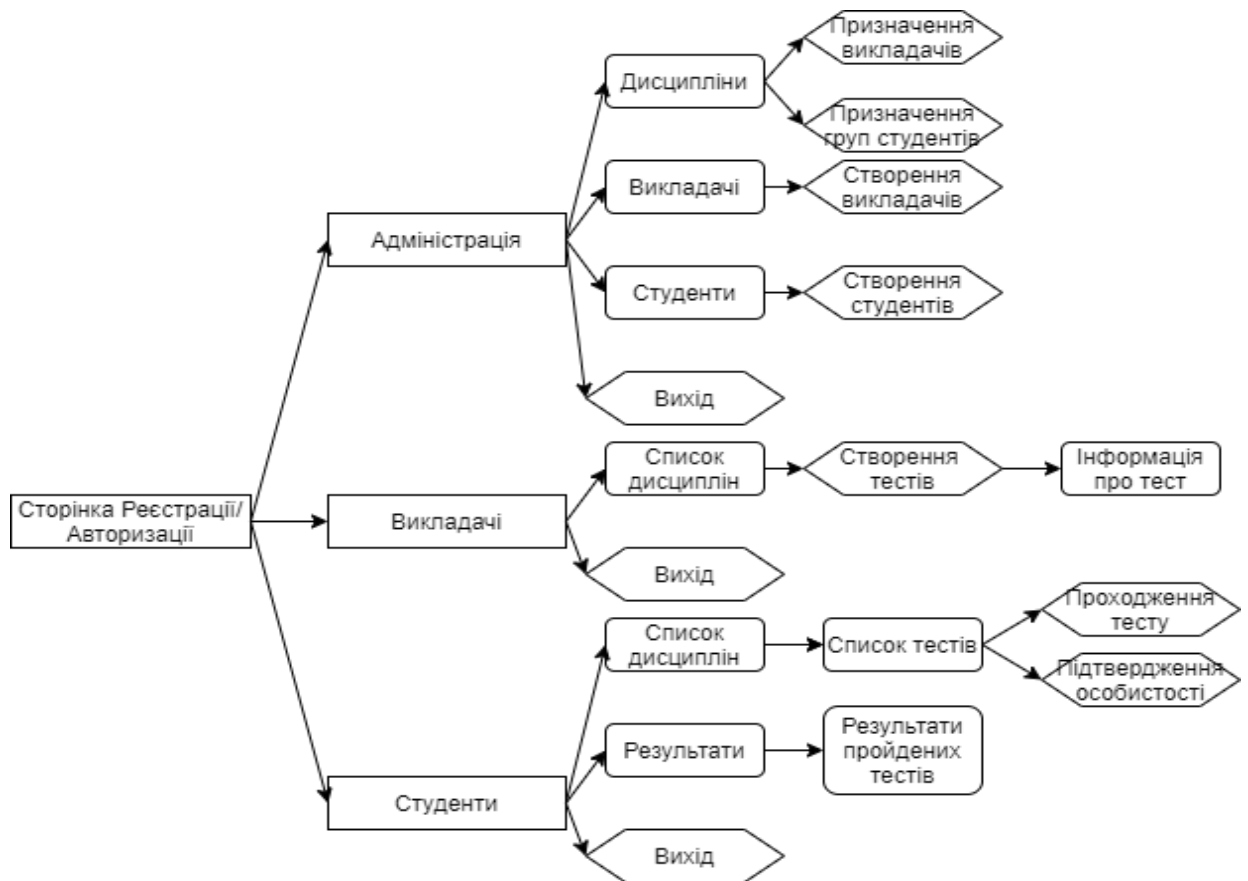


Рисунок 2.4 – Структура сайту

При відвідуванні сайту, користувач має можливість зареєструватися, як адміністратор, або ж авторизуватися, якщо був попередньо зареєстрований, як адміністратор, викладач чи студент. Викладачів та студентів реєструють адміністратори. Після процесу авторизації користувач, в залежності від права його доступу потрапляє на одну з сторінок. Адміністрація має функціонал для створення нових дисциплін та після призначення для них викладачів та студентів, створювати нових викладачів та сторінки для них, а також створювати студентів та персональні сторінки для них.

Викладачі після авторизації потрапляють на сторінку з список дисциплін, що він викладає, при переході на сторінку дисципліни викладач має можливість створювати нове тестове завдання для конкретної дисципліни, після чого може переглядати на сторінці тесту список питань до тесту та правильні відповіді.

Студенти при авторизації отримують список дисциплін, які їм вичитуються та по яким потрібно проходити тести, створені викладачами. Під час проходження тестового завдання особистість студента фіксується шляхом

знімків, щоб робляться через веб-камеру. Після проходження тестового завдання студент може переглянути свої результати в розділі Результати.

Кожний користувач має можливість завершити сеанс, натиснувши в меню на посилання з назвою ‘Вийти’.

2.3 Вибір інструменту для реалізації ПЗ

Серед багатьох інструментів веб розробки вибір прийшовся на стек MERN.

Стек в веб середовищі – це поєднання технологій, що використовуються для створення веб додатків. Будь який веб додаток можна створити, використавши стек технологій(фрейморки, бібліотеки, бази даних і т. д.).

В свою чергу стек MERN – це JavaScript-стек розроблений для спрощення процесу розробки. MERN складається з наступних компонентів з відкритим кодом: бази даних – MongoDB, фреймворку – Express, бібліотеки – React та програмної платформи Node.js.

Комплекс даних компонентів забезпечує комфортне середовище для розробки веб додатків.

Висновок до розділу 2

Враховуючи переваги та недоліки наявних методологій розробки ПЗ для створення автоматизованої системи оцінки знань студентів було обрано Scrum, яка доволить чітко встановити часові обмеження для виконання певних завдань, що позитивно вплине на швидкість процесу розробки ПЗ.

Було розроблено архітектуру сайту, що базується на трьох основних типах користувачів. Реалізовано механізм отримання відповідного функціоналу певними користувачами, що дозволяє їм взаємодіяти з іншими користувачами та в повному обсязі виконувати свою роль в системі.

Для гнучкої розробки веб додатку обрано стек MERN, який дозволяє створити швидкий односторінковий додаток (англ. single page application), та в результаті

реалізації системи надасть користувачам зручний швидкодіючий сервіс для користування.

РОЗДІЛ 3

ДЕТАЛЬНИЙ ОПИС ПРОЦЕСУ РЕАЛІЗАЦІЇ

3.1 Процес створення та налаштування серверу і бази даних

З самого початку варто створити сервер за допомогою фреймворка Express.

Створимо для проекту новий каталог, для зберігання інформації про всі залежності проекту визначимо в цьому каталозі новий файл `package.json`:

```
{  «dependencies»: {    «bcryptjs»: «^2.4.3»,    «body-parser»: «^1.19.0»,    «config»: «^3.3.2»,    «express»: «^4.17.1»,    «express-validator»: «^6.6.1»,    «jsonwebtoken»: «^8.5.1»,    «mongoose»: «^5.10.11»,    «uuid»: «^8.3.1»  },  «devDependencies»: {    «concurrently»: «^5.3.0»,    «nodemon»: «^2.0.6»  }}
```

Після чого виконає команду `npm install`, яка встановить всі залежності та створимо в каталозі новий файл `app.js`, що буде вхідною точкою для сервера.

Підключимо модуль `express`, `config`, `mongoose`, `https` та `fs` за допомогою наступних запитів

```
const express = require('express')
const config = require('config')
const mongoose = require('mongoose')
const https = require('https')
const fs = require('fs')
```

Налаштуємо HyperText Transfer Protocol Secure, щоб підвищити безпеку користування додатком.

```
Const httpsOptions = {
  key: fs.readFileSync('./ssl/key.pem'),
  cert: fs.readFileSync('./ssl/cert.pem')};
```

Створимо об'єкт додатку

```
const app = express()
```

За допомогою функцій проміжної обробки встановимо розміри для json запитів та urlencoded, що дозволяє декодувати отримані дані.

```
App.use(express.json({limit: '10mb', extended: true}))
app.use(express.urlencoded({limit: '10mb', extended: true}))
```

Також варто зв'язати маршрути сторінок з відповідними файлами, що будуть відповідати за обробку даних на даних сторінках.

```
App.use('/api/auth', require('./routes/auth.routes'))
app.use('/api/discipline', require('./routes/discipline.routes'))
)
app.use('/api/teacher', require('./routes/teacher.routes'))
app.use('/api/student', require('./routes/student.routes'))
app.use('/api/tinfo', require('./routes/tinfo.routes'))
app.use('/api/student-
page', require('./routes/studentPage.routes'))
```

З директорії config та файлу default.json отримаємо порт, на якому запустимо сервер.

```
Const PORT = config.get('port')
```

Створимо функцію для підключення бази та створення серверу

```
async function start(){
  try{
    await mongoose.connect(config.get('mongoUri'), {
  useUrlParser: true,
    useUnifiedTopology: true,
  useCreateIndex: true})
```



```
https.createServer(httpsOptions, app).listen(PORT, () => console.log(`Port ${PORT} is running!`))}catch{ console.log(`Server error:`, e.message)process.exit(1)}}
```

Запускаємо сервер наступною функцією

```
start()
```

Наступним кроком варто створити структуру моделей для MongoDB. При розробці моделей варто звернути увагу на зв'язок між моделями, який повинен реалізовувати необхідний функціонал. Таким чином створимо основну модель User, яка матиме дочірні моделі: Teacher, Student, Discipline. В свою чергу модель Discipline повинен виступати батьківською моделлю для моделі Test.

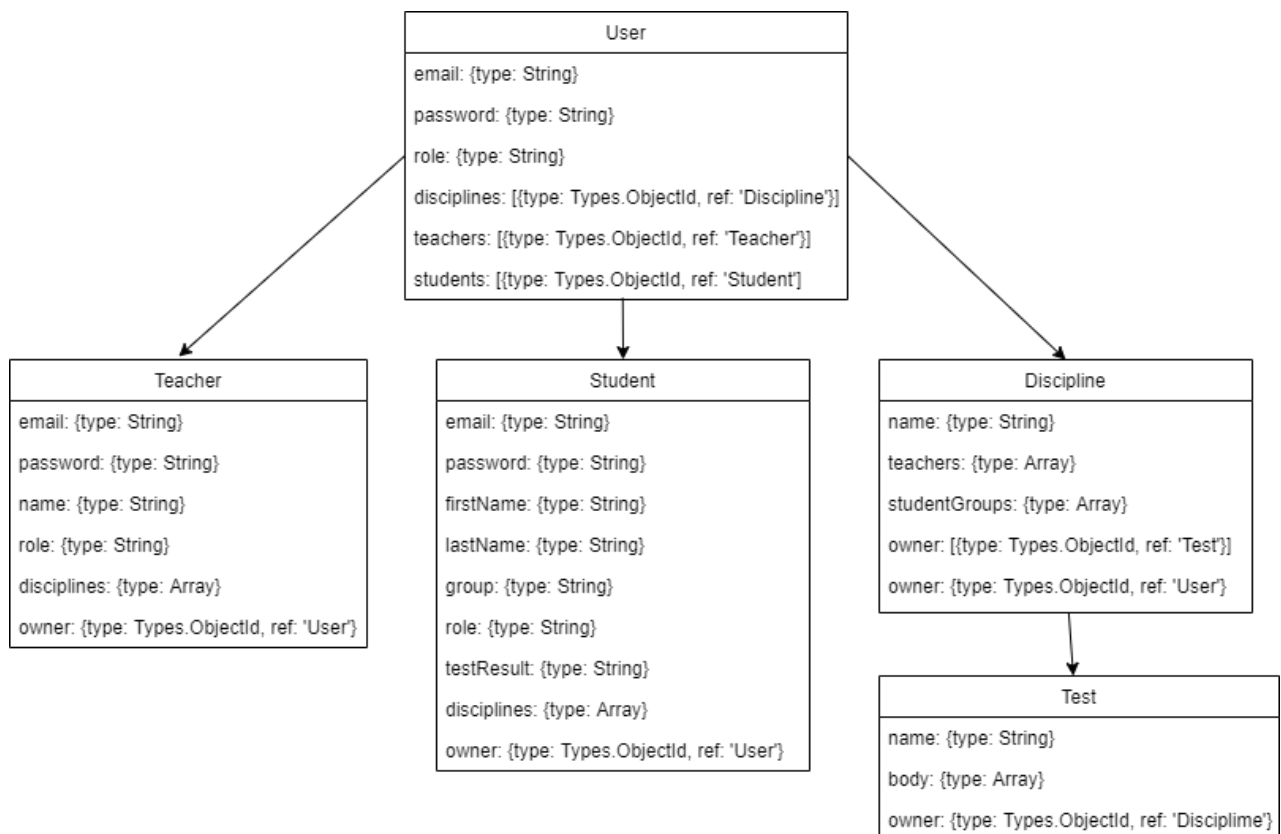


Рисунок 3.1 – Зв'язок між моделями

В результаті ми отримуємо головну модель User. Дана модель відповідає за створення нових користувачів з правами доступу адміністратора. Дана модель має поля email та password, що використовуються при реєстрації користувача.

Поле `role` – це поле, що відповідає за рівень доступу та має значення за замовчуванням `'admin'` тільки для адміністрації.

Поля `disciplines`, `teachers` та `students` мають тип `Types.ObjectId`, та атрибут `ref` – який створює посилання на інші моделі бази даних.

Модель `Teacher` має поля `email`, `password`, `name`, `role`, що використовуються адміністрацією при реєстрації нових викладачів, поле `role` має значення за замовчуванням `'teacher'`. Поле `disciplines` використовуються, коли адміністрація призначає нові дисципліни для викладача. Поле `owner` вказує, що саме модель `User` з певний `id` має доступ до цієї моделі.

Модель `Student` має поля `email`, `password`, `firstName`, `lastName`, `group`, `role`, що використовуються при реєстрації нових студентів, поле `role` також має значення за замовчування `'student'`. Поле `disciplines` має використовуються при призначенні певної дисципліни групі студентів, поле `testResult` використовуються при записі результатів проходження тестів. Поле `owner` вказує, що саме модель `User` з певний `id` має доступ до цієї моделі.

Модель `Discipline` має поле `name`, яке використовується при створенні дисципліни, поля `teachers` та `studentGroups` використовуються для призначення викладачів та груп студентів. Поле `test` вказує на іншу модель бази даних `MongoDB`, а саме на модель `Test`. Поле `owner` вказує, що саме модель `User` з певний `id` має доступ до цієї моделі.

Модель `Test` має поля `name` та `body`, що заповнюються при створенні тесту та його доповненні. Поле `owner` вказує, що саме модель `Discipline` з певний `id` має доступ до цієї моделі.

Наступним кроком створимо директорію `client` в кореневій директорії проекту та встановимо бібліотеку `React.js`.

Після завершення встановлення бібліотеки створимо наступну структуру директорій (рис. 13)

Файл `index.js`, `setupTests` та `App.js` є файлами, що встановлюються разом з бібліотекою, та відповідають за рендер сторінок та тестування додатку.

Створимо директорію `pages` в яку помістимо головні сторінки, а саме сторінку Авторизації, а також сторінки для Адміністрації, Викладачів та Студентів.

Директорія `context` міститиме один файл, що буде передавати контекстні дані користувача після його авторизації.

Директорія `hooks` міститиме допоміжні файли, що допоможуть обробляти та відправляти дані на сервер а також створювати контекст при авторизації.

Директорія `components` відповідатиме за компоненти сторінок сайту, тому міститиме додатково директорії та піддиректорії, наприклад для адміністративної сторінки створимо директорію `admin`, в якій додатково створимо директорії `student` та `teacher`, а також супровідні файли. Для сторінки викладачів та студентів створимо директорії `teacher` та `student` відповідно.

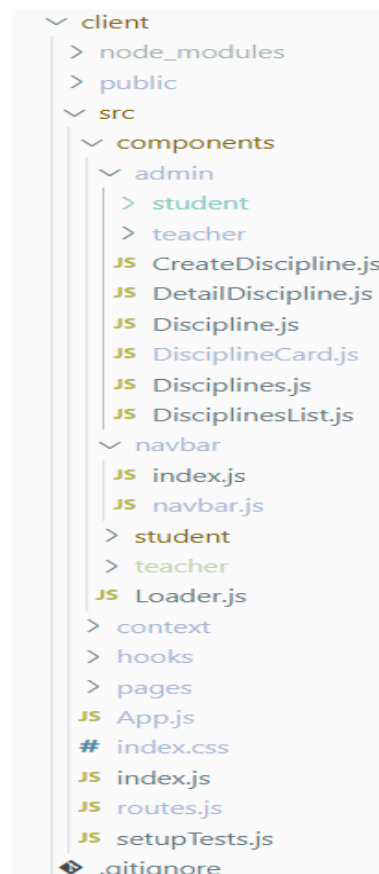


Рисунок 3.2 – Структура директорій та файлі що відповідає за frontend сайту

3.2 Реалізація функціоналу для користувачів системи

Розпочнемо з фалу App.js

```
function App() {
```

Отримаємо з auth.hook.js, що знаходиться в директорії hook об'єкт з необхідними полями для авторизації.

```
  Const {token, login, logout, userId,
    userRole, ready, teacherName
  } = useAuth()

  const isAuthenticated = !!token
```

Для файлу routes передаємо 2 значення, а саме чи авторизований користувач та роль користувача в системі

```
  const routes = useRoutes(isAuthenticated, userRole)
```

Здійснюємо перевірку чи отримали ми значення для контексту, якщо ні, на екрані відображається спінер завантаження

```
  if(!ready) {
    return <Loader />
  }
```

Записуємо отримані значення в контекст

```
  return (
    <AuthContext.Provider value={{
      token, login, logout, userId, isAuthenticated,
      userRole, teacherName
    }}>
      <Router>
        <div className=»container»>
          {routes}
        </div>
```

```

    </Router>

    </AuthContext.Provider>

  ) }

```

Створимо сторінку реєстрації та авторизації, для цього в директорії pages створимо файл AuthPage.js

```
export const AuthPage = () => {
```

Отримаємо контекст для авторизації

```

  const auth = useContext(AuthContext)

  const {loading, request, error, clearError} = useHttp()

```

Використаємо хук useState, щоб записувати значення в змінні, які отримаємо з форми

```

  const [form, setForm] = useState({
    email: '', password: '', role: 'admin'
  })

```

За допомогою функції changeHandler збираємо всі дані в єдиний об'єкт з відповідними полями.

```

  Const changeHandler = event => {
    setForm({ ...form, [event.target.name]: event.target.value })
  }

```

Функція registerHandler отримує об'єкт form та робить запит на відправку даного об'єкту на сервер на наступним адресом: '/api/auth/register', щоб зареєструвати користувача.

```

  Const registerHandler = async () => {
    try {
      const data = await request('/api/auth/register', 'POST', {...form})
    } catch {}
  }

```

```
}
```

Функція `loginHandler` також отримує об'єкт `form` та робить запит на сервер, щоб авторизувати користувача, а також, якщо запит успішний, то передає інформацію про користувача в контекст.

```
Const loginHandler = async () => {
  try {
    const data = await request('/api/auth/login', 'POST', {...
form})

    auth.login(data.token, data.userId, data.userRole, data.te
acherName)

  } catch {}
}
```

Розглянемо процес обробки даних на сервері для реєстрації та авторизації. Розпочнемо з реєстрації. Перед обробкою даних варто перевірити, чи відповідають надіслані дані вірному формату. Для цього використаємо функцію `isEmail()` для перевірки поля `email`, чи дійсно отримане значення відповідає формату електронної пошти, наступним перевіримо пароль за допомогою функції `isLength`, чи відповідає воно довжині, яка задана в мінімум 6 символів

```
router.post(
  '/register',
  [
    check('email', 'Некоректна пошта').isEmail(),
    check('password', 'Мінімальна довжина символів повинна бути
більше 6')
      .isLength({ min: 6 })
  ],
  async (req, res) => {
    try {
      const errors = validationResult(req)
```

Якщо дані, які були надіслані при реєстрації виявляться невірними, отримаємо помилку та виведемо для користувача повідомлення, що він ввів некоректні дані.

```
If (!errors.isEmpty()) {
    return res.status(400).json({
        errors: errors.array(),
        message: 'Некоректні данні при реєстрації'
    })
}

const {email, password, role} = req.body
```

Наступним кроком ми отримуємо данні та за полем email перевіряємо чи існує вже такий користувач в системі.

```
Const candidate = await User.findOne({ email })

if (candidate) {
    return res.status(400).json({ message: 'Такий користувач в
же існує' })
}
```

Якщо ж користувача немає, то хешуємо пароль за допомогою bcrypt та створюємо нового користувача з захешованим паролем.

```
Const hashedPassword = await bcrypt.hash(password, 12)

const user = new User({ email, password: hashedPassword, rol
e })

await user.save()

res.status(201).json({ message: 'Користувача створено' })

} catch {

    res.status(500).json({ message: 'Щось пішло не так, спробуйт
е заново' })

}}
```

Процес обробки даних при авторизації виглядає наступним чином. Перевіряємо чи саме пошта була введена, а також, чи був введений пароль.

```
Router.post(
  '/login',
  [
    check('email', 'Введіть коретну пошту').normalizeEmail().isEmail(),
    check('password', 'Введіть пароль').exists()
  ],
  async (req, res) => {
    try {
      const errors = validationResult(req)
```

Якщо данні, що були отримані не відповідають умовам, то виведемо повідомлення, що дані некоректні

```
    if (!errors.isEmpty()) {
      return res.status(400).json({
        errors: errors.array(),
        message: 'Некоректні дані при вході в систему'
      })
    }
  }
```

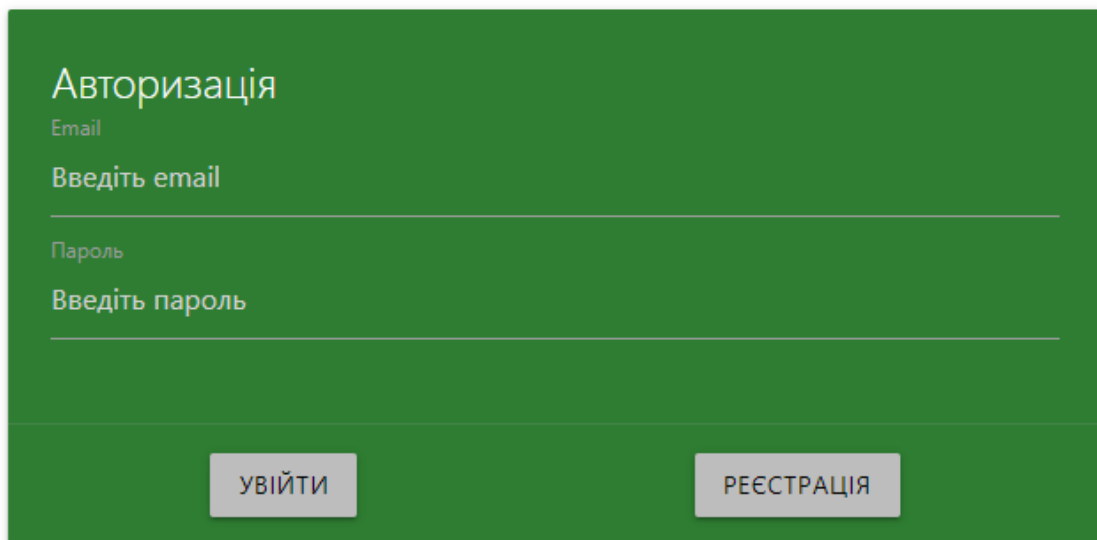
Якщо користувача знайшли в базі даних, то порівнюють пароль який було введемо, з вже існуючим паролем, якщо паролі різні, то виводимо повідомлення, що пароль некоректний.

```
    const {email, password} = req.body
    if(user){
      const isMatch = await bcrypt.compare(password, user.password
    )

      if (!isMatch) {
        return res.status(400).json({ message: 'Некоретний пароль, спробуйте заново' })
      }
    }
```


Для входу в систему використаємо токен, який буде дійсний протягом 8 годин, після чого користувачу потрібно заново пройти процес авторизації.

```
Const token = jwt.sign(  
  {  
    userId: user.id,  
    userRole: user.role  
  },  
  config.get('jwtSecret'),  
  { expiresIn: '8h' }  
)  
  
res.json({ token, userId: user.id, userRole: user.role })  
})
```



Авторизація

Email

Введіть email

Пароль

Введіть пароль

УВІЙТИ

РЕЄСТРАЦІЯ

Рисунок 3.3 – Вікно авторизації та реєстрації

Наступний кроком, потрібно надавати доступ до різних сторінок користувачам, що мають відповідні права доступу, для цього в файлі routes напишемо наступний код, де з файлу App.js ми отримуємо значення стану

авторизації користувача та його роль в системі та відповідно до цього Адміністрацію направляємо на сторінку авторизації.

```
export const useRoutes = (isAuthenticated, userRole) => {
  if (isAuthenticated) {
    switch (userRole){
      case 'admin':
        return(<AdminPage />)
      case 'teacher':
        return(<TeacherPage />)
      case 'student':
        return(<StudentPage />)
      default: break;
    }
  }
  return (<AuthPage />)}
}
```

Після розподілення ролей варто перейти до створення функціоналу для кожного типу користувача. Розпочнемо з адміністрації. Компоненти для адміністрації матимуть наступний вигляд (рис. 3.4):

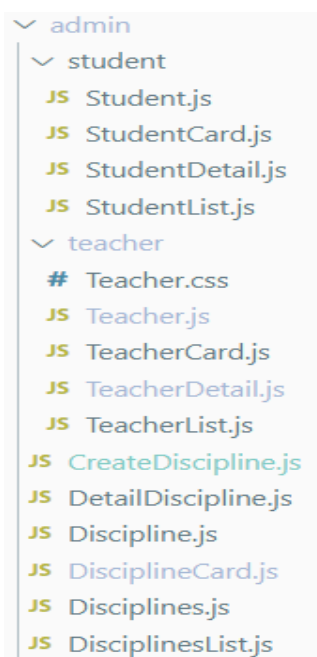


Рисунок 3.4 – Структура компонентів для адміністрації

В свою чергу основна сторінка Адміністрації матиме наступний вигляд:

```
export const AdminPage = () =>{
  return(
    <Router>
      <Navbar />
      <Switch>
        <Route exact path=»/discipline» component={Discipline}></Route>
        <Route exact path=»/discipline/:id» component={DetailDiscipline}><
        /Route>
        <Route exact path=»/teacher» component={Teacher} />
        <Route exact path=»/teacher/:id» component={TeacherDetail} />
        <Route exact path=»/student» component={Student} />
        <Route exact path=»/student/:id» component={StudentDetail} />
      </Switch>
      <Redirect to=»/discipline» />
    </Router>
  )
}
```

З даного фрагменту коду можна побачити, що певні адреси та відповідні до них компоненти, а також те, що при авторизації користувача спрямовують на сторінку ‘./discipline’.

Розглянемо всі компоненти та розпочнемо з компоненту Discipline, який має наступний вигляд:

```
export const Discipline = () =>{
  return(
    <><CreateDiscipline />
    <Disciplines /></> ) }
```

Даний компонент містить в собі два інших компоненти, які відповідають за створення дисциплін та виведення списку вже існуючих дисциплін.

Розглянемо компонент `CreateDiscipline`.

```
Export const CreateDiscipline = () =>{
  const history = useHistory()
  const auth = useContext(AuthContext)
  const {request} = useHttp()
  const [discipline, setDiscipline] = useState('')
```

Даний хук дозволяє очищати поля на сторінці після створення дисципліни

```
useEffect(() => {
  window.M.updateTextFields()
}, [])
```

Функція `pressHandler` дозволяє створити нову дисципліну та сторінку для неї, назвою сторінки виступає `id` новоствореної дисципліни

```
const pressHandler = async event => {
  if (event.key === 'Enter') {
    try {
      const data = await request('/api/discipline/discipline-
create', 'POST', {name: discipline}, {
        Authorization: `Bearer ${auth.token}`
      })
      history.push(`/discipline/${data.discipline._id}`)
    } catch {}
  }
}
```

JSX код виглядає наступним чином та складається з блоків-обгортки та поля `input` з відповідним `label`ом для поля `input`

```
return(
```

```

<div className=»row»>

  <div className=»col s8 offset-
s2» style={{paddingTop: `2rem`}}>

    <div className=»input-field»>

      <input

        placeholder=»»

        id=»discipline»

        type=»text»

        value={discipline}

        onChange={e => setDiscipline(e.target.value)}

        onKeyDown={pressHandler}

      />

    <label htmlFor=»discipline»>Введіть назву нової дисципліни</
label>

      </div>

    </div>

  </div>)}

```

Розглянемо відповідний маршрут для цього компонента, а саме “/api/discipline/discipline-create”, що має наступний вигляд:

```

router.post(`/discipline-create`, auth, async(req, res) =>{
  try {
    const {name} = req.body
    const existing = await Discipline.findOne({ name })
    if(existing){
      return res.json({ discipline: existing})
    }
    const discipline = new Discipline({
      name, owner: req.user.userId
    })
    await discipline.save()
  }
})

```

```

    res.status(201).json({ discipline })

  } catch (e) {

    res.status(500).json({ message: `Щось пішло не так, спробуйте знову` })})})
  }
}

```

З коду видно, що ми отримуємо назву дисципліни, перевіряємо, чи існує вже така дисципліна, якщо існує, надсилаємо у відповідь інформацію про існуючу дисципліну, якщо ж дисципліни немає, то створює нову.

Надалі розглянемо компонент `Disciplines`, що відповідає виведення списку вже створених дисциплін та має наступний код:

```

export const Disciplines = () => {

  const [disciplines, setDisciplines] = useState([])

  const {loading, request} = useHttp()

  const {token} = useContext(AuthContext)

  const fetchDisciplines = useCallback(async () => {

    try {

      const fetched = await request(`/api/discipline`, 'GET', null, {

        Authorization: `Bearer ${token}`

      })

      setDisciplines(fetched)

    } catch (e) {}

  }, [token, request])

  useEffect(() => {

    fetchDisciplines()

  }, [fetchDisciplines])

  if (loading) {return <Loader/>}

  return (<>{!loading && <DisciplinesList disciplines={disciplines} />}</>) }

```

Даний компонент складається з запиту на сервер та передачі отриманої інформації на в інший компонент – `DisciplinesList`.

Розглянемо процес запиту на сервер, що виглядає наступним чином:

```
router.get('/', auth, async (req, res) => {
  try {
    const disciplines = await Discipline.find({ owner: req.user.userId })
    res.json(disciplines)
  } catch {
    res.status(500).json({ message: 'Щось пішло не так, спробуйте знову' })
  }
})
```

При запиті відбувається пошук всіх дисциплін, власником яких є певний користувач (адміністратор), в результаті чого ми отримуємо масив з об'єктами. В результаті дані об'єкти передаються в компонент DisciplinesList, де дані виводяться в таблицю, що реалізовано наступним чином.

```
Export const DisciplinesList = ({ disciplines }) => {
  if(!disciplines.length) {
    return <p className=»center»>Дисциплін поки немає</p>
  }
  return(
    <table>
      <thead>
        <tr>
          <th>№</th>
          <th>Назва</th>
          <th>Сторінка дисципліни</th>
        </tr>
      </thead>
      <tbody>
        { disciplines.map((discipline, index) => {
          return (
            <tr key={discipline._id}>
```

```

<td>{index + 1}</td>

<td>{discipline.name}</td>

<td>

<Link to={`/${discipline}/${discipline._id}`}>Перейти</
Link>

</td>

</tr>

)}}}

</tbody>

</table>)}

```

За допомогою функції `map` ми виводимо всі дисципліни в табличку, яка складається з порядкового номеру дисципліни, назви та посилання на сторінку дисципліни.

Сторінка має наступний вигляд (рис. 3.5)

Адміністративна панель			Дисципліни	Викладачі	Студенти	Вийти
Введіть назву нової дисципліни						
№	Назва	Сторінка дисципліни				
1	Тестова назва	Перейти				
2	Нова дисципліна	Перейти				
3	Дисципліна	Перейти				
4	Тестова назва - 2	Перейти				
5	Додаткові розділи теорії ЧЕ	Перейти				
6	Спеціальні розділи фахових дисциплін	Перейти				
7	Теорія автоматичного управління-1. Теорія лінійних систем автоматичного управління	Перейти				
8	Методологія наукових досліджень у галузі	Перейти				
9	Технології розробки програмного забезпечення-1	Перейти				
10	Системи орієнтації і стабілізації	Перейти				
11	Основи теорії вимірювальних приладів	Перейти				

Рисунок 3.5 – Вигляд сторінки для створення дисциплін

Наступник кроком варто створити функціонал для сторінки дисципліни, на якій адміністрація повинна мати можливість призначати викладачів та студентів для певної дисципліни, заданий функціонал реалізуємо наступним чином.

```

Export const DisciplineCard = ({ discipline }) => {
  const {loading, request, error, clearError} = useHttp()
  const message = useMessage()
  const [form, setForm] = useState({name: ''})
  const [student, setStudent] = useState({group: ''})
  const auth = useContext(AuthContext)
  useEffect(() => {
    message(error)
    clearError()
  }, [error, message, clearError])
  useEffect(() => {
    window.M.updateTextFields()
  }, [])
  const changeHandlerTeacher = event => {
    setForm({ ...form, [event.target.name]: event.target.value })
  }
  const changeHandlerStudent = event => {
    setStudent({ ...student, [event.target.name]: event.target.value })
  }
  const addHandlerTeacher = async () => {
    try {
      const dataDiscipline = await request(`/api/discipline/update-teacher`, 'POST', {...form, discipline},
        {
          Authorization: `Bearer ${auth.token}`
        })
    }
  }
}

```

```

    const dataTeacher = await request(`/api/teacher/update-
discipline`, `POST`, {...form, discipline},

    {
        Authorization: `Bearer ${auth.token}`
    })

    message(dataDiscipline.message)

    message(dataTeacher.message)

} catch € {}

}

const addHandlerStudent = async () => {

    try {

        const dataDiscipline = await request(`/api/discipline/update-
-group`, `POST`, {...student, discipline},

        {Authorization: `Bearer ${auth.token}`})

        const dataGroup = await request(`/api/student/update-
discipline`, `POST`, {...student, discipline},

        {Authorization: `Bearer ${auth.token}`})

    } catch € {}

return (

<><h4>Дисципліна - {discipline.name}</h4>

<div className=»row»>

    <div className=»col s4»>

        <p>Викладачі: {discipline.teachers} </p>

        <div className=»input-field»>

            <input

                placeholder=»Введіть викладача»

                id=»name»

                type=»text»

                name=»name»

                className=»yellow-input»

                value={form.name}

```

```

        onChange={changeHandlerTeacher}
      />
    <label htmlFor=»name»></label>
    <button
      className=»btn grey lighten-1 black-text»
      onClick={addHandlerTeacher}
      disabled={loading}
    >
      Додати викладача
    </button>
  </div>
</div>
</div>
<div className=»row»>
  <div className=»col s4»>
    <p>Групи студентів: {discipline.studentGroups}</p>
    <div className=»input-field»>
      <input
        placeholder=»Введіть групу студентів»
        id=»group»
        type=»text»
        name=»group»
        className=»yellow-input»
        value={student.group}
        onChange={changeHandlerStudent}/>
      <label htmlFor=»group»></label>
      <button
        className=»btn grey lighten-1 black-text»
        onClick={addHandlerStudent}

```

```

disabled={loading}>Додати групу
</button></div></div></div></>)}

```

Дані з відповідних input`ів ми отримуємо та відправляємо на сервер, де обробляємо наступним чином для викладачів, де спочатку записуємо викладача в дисципліну

```

router.post('/update-teacher', auth, async (req, res) => {
  try {
    const {name, discipline} = req.body
    const disc = await Discipline.findOne(discipline)
    disc.teachers.push(name)
    await disc.save()
    res.json(disc)

  } catch € {
    res.status(500).json({ message: 'Щось пішло не так, як треба, спробуйте знову' })
  }
})

```

Другим кроком записуємо дану дисципліну викладачу

```

router.post('/update-discipline', auth, async (req, res) => {
  try {
    const {name, discipline} = req.body
    const teach = await Teacher.findOne({name})
    teach.disciplines.push(discipline.name)
    await teach.save()
    res.json(teach)

  } catch € {
    res.status(500).json({ message: 'Щось пішло не так, як треба, спробуйте знову' }}})

```

У випадку з студентами спочатку записуємо групу студентів до дисципліни

```
router.post('/update-group', auth, async (req, res) => {
  try {
    const {group, discipline} = req.body
    const disc = await Discipline.findOne(discipline)
    disc.studentGroups.push(group)
    await disc.save()

    res.json(disc)} catch € {res.status(500).json({ message: 'Щось
    пішло не так, як треба в дисц-груп, спробуйте знову' }}}})
```

Як результат дисципліну записуємо для кожного студента, який має відповідну групу:

```
router.post('/update-discipline', auth, async (req, res) => {
  try {
    const {group, discipline} = req.body
    const gr = await Student.find({group})
    gr.forEach(item => {item.disciplines.push(discipline.name)
      item.save() })
    res.json(gr)

    } catch € {res.status(500).json({ message: 'Щось пішло не так, я
    к треба з студентами, спробуйте знову' }}}})
```

Дана сторінка з додаванням викладачів та студентів має наступний вигляд (рис. 3.6)

Адміністративна панель

Дисципліни Викладачі Студенти Вийти

Дисципліна - Тестова назва

Викладачі: Сергій Цибульник

Введіть викладача

ДОДАТИ ВИКЛАДАЧА

Групи студентів: ПГ-91-мп

Введіть групу студентів

ДОДАТИ ГРУПУ

Рисунок 3.6 – Вигляд сторінки унікальної дисципліни

Наступним кроком варто створити сторінку викладачів, так само, як і для дисциплін потрібно реалізувати процес створення викладачів, список всіх викладачів та персональні сторінки викладачів. Розпочнемо з процесу створення викладачів.

```
Export const Teacher = () => {
```

```
  const history = useHistory()
```

```
  const message = useMessage()
```

```
  const auth = useContext(AuthContext)
```

```
  const {loading, request, error, clearError} = useHttp()
```

Для викладачів виділимо наступні поля: email, password, role, name, disciplines

```
  const [form, setForm] = useState({
```

```
    email: '', password: '', role: 'teacher', name: '', disciplines:
  []})
```

```
  useEffect(() => {
```

```
    message(error)
```

```
    clearError()}, [error, message, clearError])
```

```
  useEffect(() => {
```

```
    window.M.updateTextFields()}, [])
```

```
  const changeHandler = event => {
```

```
    setForm({ ...form, [event.target.name]: event.target.value })}
```

```
  const registerHandler = async () => {
```

```
    try {
```

```
      const data = await request('/api/teacher/register', 'POST', {...
    .form},
```

```
      {Authorization: `Bearer ${auth.token}`})
```

```
      history.push(`/teacher/${data.teacher._id}`)
```

```
    } catch {}
```

Відповідно створимо 3 input`а для введення необхідних даних при реєстрації, даний компонент також міститиме компонент TeacherList, який виводить список створених викладачів за тим самим принципом, що й виведення дисциплін.

```

Return (
  <div className=»row»>
    <div className=»col s5 offset-s3»>
      <div className=»teacherCard card white»>
        <div className=»card-content white-text»>
          <div>
            <div className=»input-field»>
              <input      placeholder=»Введіть email»      id=»email»
type=»text»
              name=»email» value={form.email}
              onChange={changeHandler}
            /><label htmlFor=»email»>Email</label></div>
            <div className=»input-field»>
              <input
                placeholder=»Введіть пароль»id=»password»
                type=»password»                      name=»password»
value={form.password}
                onChange={changeHandler}/>
              <label htmlFor=»email»>Пароль</label> </div>
            <div className=»input-field»>
              <input
                placeholder=»Введіть ім'я викладача»
                id=»name» type=»text» name=»name»
                value={form.name} onChange={changeHandler}/>
              <label htmlFor=»name»>Ім'я викладача</label>
            </div></div></div>
          <div className=»card-action center»>
            <button
              className=»btn grey lighten-1 black-text»
              onClick={registerHandler} disabled={loading}>

```

```

Додати викладача</button></div></div></div>

<TeacherList /> </div>)}

```

В результаті отримаємо наступну сторінку (рис. 3.7)

Адміністративна панель

ДисципліниВикладачіСтудентиВийти

Email

Введіть email

Пароль

Введіть пароль

Ім'я викладача

Введіть ім'я викладача

ДОДАТИ ВИКЛАДАЧА

№	Ім'я та прізвище викладача	Дисципліни викладача	Email викладача	Сторінка викладача
1	Сергій Цибульник	Тестова назваНова дисциплінаТестова назва - 2	test@ukr.net	Перейти
2	Осіпова Юлія	Нова дисципліна	test3@ukr.net	Перейти
3	Вадим Аврутов		test4@ukr.net	Перейти
4	Надія Бурау		test5@ukr.net	Перейти

Рисунок 3.7 – Сторінка для створення нових викладачів

Компоненти `TeacherCard` та `TeacherDetail` повторюють аналогічний функціонал з розділу дисциплін з різницею в тому, що виводяться інші поля, тому продемонструємо готовий результат (рис. 3.8).

Адміністративна панель

ДисципліниВикладачіСтудентиВийти

Викладач - Сергій Цибульник

Пошта викладача: test@ukr.net

Дисципліни викладача: Тестова назваНова дисциплінаТестова назва - 2

Рисунок 3.8 – Персональна сторінка викладача

Останнім важливим елементом для адміністрації є процес створення студентів, даний процес аналогічний процесу створення викладачів, тому продемонструю відразу результат.

Сторінка створення студентів включатиме в себе наступні поля: Email, Пароль, Ім'я студента, Прізвище студента, Група студента та матиме наступний вигляд (рис. 3.9)

Адміністративна панель

ДисципліниВикладачіСтудентиВийти

Email

Введіть email

Пароль

Введіть пароль

Ім'я студента

Введіть ім'я студента

Прізвище студента

Введіть прізвище студента

Група студента

Введіть групу студента

ДОДАТИ СТУДЕНТА

Рисунок 3.9 – Сторінка створення студентів

Та відповідно, персональна сторінка студента матиме наступний вигляд (рис. 3.10)

Адміністративна панель

ДисципліниВикладачіСтудентиВийти

Студент - Владислав Кобзар

Пошта: test1@ukr.net

Група: ПГ-91-мп

Дисципліни: Тестова назва, Нова дисципліна, Тестова назва - 2,

Результати тестувань:

Новий тест, оцінка 0.00

Новий тест, оцінка 0.00

Новий тест, оцінка 0.67

Новий тест, оцінка 1.00

Новий тест, оцінка 0.00

Новий тест, оцінка 0.67

Тест, оцінка 1.00

Рисунок 3.10 – Персональна сторінка студентів

Наступним важливим етапом є процес створення сторінок для викладачів. Після авторизації викладачі переходитимуть на сторінку з список доступних дисциплін, що матиме наступний вигляд (рис. 3.11)

Викладач Сергій Цибульник			Дисципліни	Вийти
№	Назва	Сторінка дисципліни		
1	Тестова назва	Перейти		
2	Нова дисципліна	Перейти		
3	Тестова назва - 2	Перейти		

Рисунок 3.11 – Сторінка викладачів з доступними дисциплінами

Кожна дисципліна має власну унікальну сторінку, де ви можете побачити інформацію про те, якій групі викладається дана дисципліна, створити новий тест для даної дисципліни та переглянути результати тестування студентів. В свою чергу сторінка матиме наступний вигляд (рис 3.12):

Викладач Сергій Цибульник			Дисципліни	Вийти
<div>Дисципліна - Тестова назва</div> <div>Групи: ПГ-91-мп</div> <div>Існуючі тести</div> <div>Новий тест</div> <div>Тест</div> <div>СТВОРИТИ ТЕСТ</div>				
<div>Сторінка тесту</div> <div>Перейти</div> <div>Перейти</div>				
Результати тестів				
Ім'я	Прізвище	Назва	Оцінка	Результат
Владислав	Кобзар	Новий тест	0.00	0 з 2
Владислав	Кобзар	Новий тест	0.00	0 з 3
Владислав	Кобзар	Новий тест	0.00	0 з 3
Ім'я	Прізвище	Назва	Оцінка	Результат
Марина	Лещук	Тест	0.00	0 з 1

Рисунок 3.12 – Сторінка викладачів з доступними дисциплінами

Реалізуємо функціонал створення тестових завдань наступним чином

```
export const TestCreator = () =>{
```

Об’являємо всі необхідні хуки для роботи з даними

```
const history = useHistory()
const [test, setTest] = useState({})
const {token} = useContext(AuthContext)
const {request, loading} = useHttp()
const disciplineId = useParams().id
const modalRef = useRef()
```

Функція для виклику модального вікна

```
const openModal = () => {
  modalRef.current.openModal()
}
useEffect(() => {
  window.M.updateTextFields()
}, [])
```

Функція для збору введених даних в одну змінну, яка представляє собою об’єкт

```
const changeHandler = event => {
  setTest({ ...test, [event.target.name]: event.target.value })
  console.log({...test})
}
```

Функція для відправки зібраних даних на сервер та створення персональної сторінки тестового завдання

```
const saveHandler = async () =>{
  try {
    const data = await request(`/api/tinfo/${disciplineId}/create-test`, 'POST', {...test},
      {Authorization: `Bearer ${token}`})
    history.push(`/tinfo/${disciplineId}/${data.test._id}`)
```

```

    } catch (e) {}
  }
}

```

Створимо кнопку, в яку передамо функцію з викликом модального вікна

```

return (
  <><button
    className=»btn grey lighten-1 black-text«
    onClick={openModal}
    disabled={loading}>Створити тест</button>

```

Створимо компонент Modal та відповідні поля для створення тесту

```

<Modal ref={modalRef}>
  <div className=»row«>
    <div className=»input-field col s12«>
      <input
        placeholder=»Назва тесту« id=»name« type=»text«
        name=»name« value={test.name} onChange={changeHandler}
      /><label htmlFor=»name«></label></div> </div>
    <form id=»create-test-form«>
      <div className=»row«>
        <div className=»input-field col s12«>
          <input
            placeholder=»Питання« id=»questions«
            type=»text« name=»questions«
            value={test.questions} onChange={changeHandler} />
          <label htmlFor=»questions«></label>
        </div>
      </div>
      <div className=»row answer«>
        <p className=»col s1«>
          <label>

```

```

        <input
            className=»with-
gap» name=»correctAnswer» type=»radio»
            value=»0» onChange={changeHandler}/>
        <span></span></label></p>
<div className=»input-field col s11 «>
    <input
        placeholder=»Текст відповіді» id=»answers_0»
        type=»text» name=»answers_0» value={test.answers_0}
        onChange={changeHandler}/>
    <label htmlFor=»answers_0»></label>
</div>
</div>
<div className=»row answer»>
    <p className=»col s1»>
        <label>
            <input className=»with-gap» name=»correctAnswer»
                type=»radio» value=»1»
                onChange={changeHandler}/><span></span>
        </label>
    </p>
    <div className=»input-field col s11 «>
        <input placeholder=»Текст відповіді»
            id=»answers_1» type=»text» name=»answers_1»
            className=»yellow-input» value={test.answers_1}
            onChange={changeHandler}/>
        <label htmlFor=»answers_1»></label>
    </div></div>
<div className=»row answer»>
    <p className=»col s1»>

```

```

    <label>

      <input className=»with-gap«
        name=»correctAnswer« type=»radio«
        value=»2«onChange={changeHandler}/><span></span></la
bel>

</p><div className=»input-field col s11 «>
  <input placeholder=»Текст відповіді« id=»answers_2«
    type=»text« name=»answers_2« value={test.answers_2}
    onChange={changeHandler}/>
  <label htmlFor=»answers_2«></label>
</div></div></form>

<div className=»row«>
  *Після введення відповідей на питання не забудьте обрати
  правильну відповідь</div>

  <div className=»model-buttons«>
    <button
      className=»btn grey lighten-1 black-text«
      onClick={() => modalRef.current.close()}>
      Закрити вікно</button>
    <button
      className=»btn grey lighten-1 black-text«
      onClick={saveHandler}>
      Створити</button></div>
  </Modal></>)}

```

Також увагу варто приділити створенню компонента `Modal`, в якому створюємо функції для відкриття та закриття модального вікна, яке в результаті створимо за допомогою `ReactDOM.createPortal`

```

export const Modal = forwardRef((props, ref) => {
  const [display, setDisplay] = React.useState(false);

```

```

useImperativeHandle(ref, () => {
  return {
    openModal: () => open(),
    close: () => close()}})
const open = async () => {
  setDisplay(true)}
const close = async () => {
  setDisplay(false)}
if (display === true) {
  return ReactDOM.createPortal(
    <div className={«modal-wrapper»}>
      <div onClick={close} className={«modal-backdrop»} />
      <div className={«modal-box»}>
        {props.children}
      </div>
    </div>, document.getElementById(«modal-root»))
  }return null}))

```

В результаті написання даного коду модальне вікно матиме наступний вигляд:

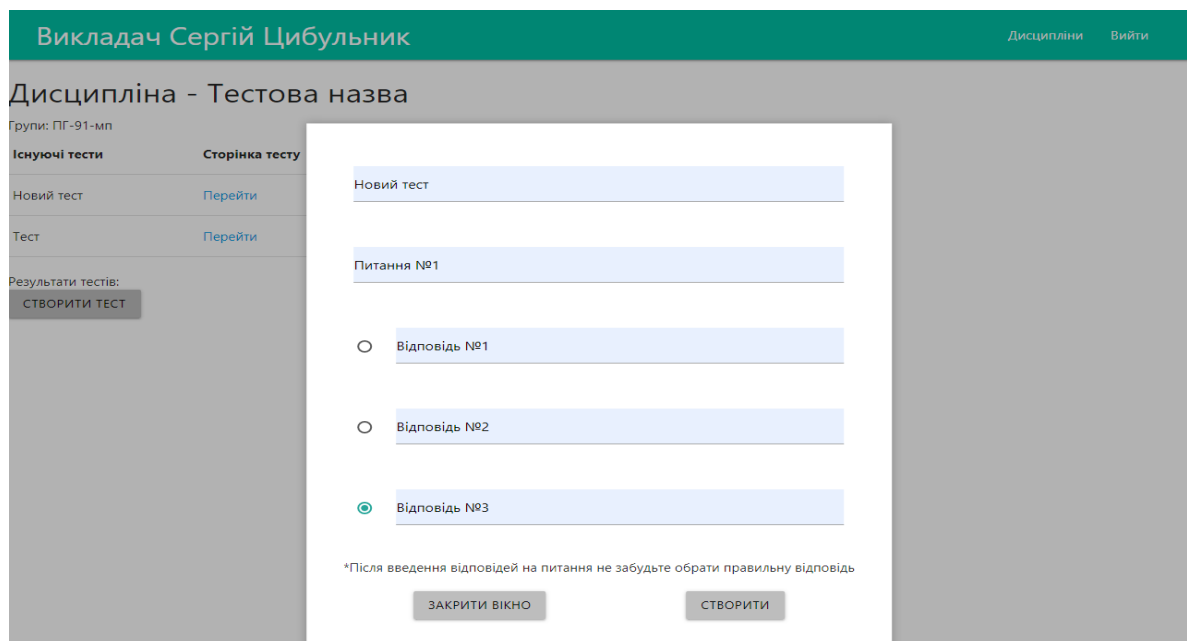


Рисунок 3.13 – Модальне вікно для створення тестового завдання

В свою чергу сторінка з інформацією про тестове завдання матиме наступний вигляд:

Викладач Сергій Цибульник

Дисципліни Вийти

Новий тест

Питання №1

- Відповідь №1
- Відповідь №2
- Відповідь №3

Правильна відповідь: Відповідь №2

Питання №2

- Відповідь №1
- Відповідь №2
- Відповідь №3

Правильна відповідь: Відповідь №3

Питання №3

- Відповідь №1
- Відповідь №2
- Відповідь №3

Правильна відповідь: Відповідь №2

Рисунок 3.14 – Сторінка створеного тесту для певної дисципліни

Наступним важливим етапом буде створення сторінки для студентів, розпочнемо з простого виведення доступних дисциплін

Студент Владислав Кобзар			Дисципліни	Результати	Вийти
№	Назва	Сторінка дисципліни			
1	Тестова назва	Перейти			
2	Нова дисципліна	Перейти			
3	Тестова назва - 2	Перейти			

Рисунок 3.15 – Сторінка зі списком доступних дисциплін для студента

Наступним кроком створимо сторінку з виведення доступних тестів з певної дисципліни

Студент Владислав Кобзар			Дисципліни	Результати	Вийти
Існуючі тести		Сторінка тесту			
Новий тест		Перейти			
Тест		Перейти			

Рисунок 3.16 – Сторінка дисципліни з тестовими завданнями для студента

Ключовим етапом даної роботи є створення функціоналу проходження тестового завдання та підтвердження особистості студенту, програмний код матиме наступний вигляд:

```
export const TestPage = () => {
```

Об'явимо всі необхідні значення

```
  const history = useHistory()
  const [test, setTest] = useState()
  const {loading, request} = useHttp()
  const {token, userId} = useContext(AuthContext)
  const disciplineId = useParams().id
  const testId = useParams().id
  useEffect(() => {
    window.M.updateTextFields()
  }, [])
```

Функція для отримання даних для тестового завдання конкретної дисципліни

```
  const fetchTest = useCallback( async() => {
    try{
      const fetched = await request(`/api/tinfo/${disciplineId}/${testId}`, 'GET', null, {
        Authorization: `Bearer ${token}`
      })
      setTest(fetched)
    } catch {}
  }, [token, disciplineId, testId, request])
  useEffect(()=>{
    fetchTest()
  }, [fetchTest])
```

```

if (loading || test === undefined) {
  return <Loader />
}

```

Функція, що підраховує вірні відповіді та відправляє результат на сервер

```

const Score = async() =>{
  var inputs = document.querySelectorAll('input[type=radio]');
  var score = 0
  var testResult = 0
  var allResults = []
  for (var i = 0; i < inputs.length; i++) {
    if(inputs[i].checked){
      if(inputs[i].value == test.body[ Number(inputs[i].name)].correctAnswer){
        console.log(inputs[i], ' Відповідь вірна')
        score = score + 1
      }else{
        console.log(inputs[i], ' Відповідь не вірна')
      }
    }
  }
  testResult = (score/test.body.length).toFixed(2)
  allResults = [testResult, score, test.body.length]
  try {
    await request(`/api/student-page/save-result`, 'POST', {test, userId, allResults},
      {Authorization: `Bearer ${token}`})
    window.location.reload()
    history.push(`/student-page`)
  } catch {}
}

```

```

return(
  <div className=»row»>
    <div className=»col s8»>
      <h4>{test.name}</h4>
      <div>
        {test.body.map((el, index) => {
          return(
            <div key={index+1}>
              <div >
                <h5>{el.questions}</h5>
                {el.answers.map((ans, idx)=>{
                  return(
                    <div key={idx+1}>
                      <label className=»test-label»>
                        <input className=»with-gap»
                          name={index} type=»radio»
                          value={idx} disabled={true}/>
                        <span>{ans}</span>
                      </label><div></div>
                    </div>)) }</div></div>)) }</div>
              <button
                className=»btn grey lighten-1 black-text» onClick={Score}
                >Завершити тест</button>
            </div> <div className=»col s4»>
              <Camera />
            </div>
          </div>)) }

```

Важливим елементом даної сторінки є компонент Camera, що відповідає за процес підтвердження особистості студента під час проходження тесту та власне, активація якого дає студенту можливість обирати відповідь.

```
Export const Camera = () => {
```

Визначаємо всі необхідні параметри системи

```
const {request} = useHttp()
const auth = useContext(AuthContext)
var width = 220;
var height = 0;
var streaming = false;
var video = null;
var canvas = null;
```

Функція startup вмикатиме відео, використає функцію access(), що дозволить студентам вибирати варіанти відповіді до питань, після чого, використовуючи функцію takepicture() будуть робитися знімки студентів через веб-камеру, які в свою чергу будуть завантажуватися на сервер за допомогою функції photoLoad()

```
const startup = () => {
  access()
  video = document.getElementById('video');
  canvas = document.getElementById('canvas');
  navigator.mediaDevices.getUserMedia({video: true})
    .then(function(stream) {
      video.srcObject = stream;
      video.play();
    })
    .catch(function(err) {
      console.log(«An error occurred: « + err);
```

```

});
video.addEventListener('canplay', function(ev){
    if (!streaming) {
        height = video.videoHeight / (video.videoWidth/width);
        if (isNaN(height)) {
            height = width / (4/3);
        }
        canvas.setAttribute('width', width);
        canvas.setAttribute('height', height);
        streaming = true;
    }
}, false);
setInterval(() => {
    takepicture()
}, 25000);
}
const takepicture = () => {
    var context = canvas.getContext('2d');
    if (width && height) {
        canvas.width = width;
        canvas.height = height;
        context.drawImage(video, 0, 0, width, height);
        var data = canvas.toDataURL('image/png').replace('data:image
/png;base64,' , '');
        photoLoad(data)
    }
}

```

Щоб мати можливість відправити дані на сервер потрібно отримане зображення кодувати в формат base64.

```

Const photoLoad = async (data) =>{
  try {
    console.log({data})

    await request(`/api/student-page/load-photo`, 'POST', {data},
      {
        Authorization: `Bearer ${auth.token}`
      })
  } catch (err) {}

  const access = () =>{
    var inputs = document.querySelectorAll('input[type=radio]');
    for (var i = 0; i < inputs.length; i++) {
      inputs[i].disabled = false;
    }
  }

  return(
    <><div className=»camera»>
      <video width=»240px» height=»220px»id=»video»>Video stream
      not available.</video>
      <button onClick={startup}>Розпочати відео</button> </div>
      <canvas id=»canvas» className=»canvas»></canvas> </>
    </>
  )
}

```

Розглянемо процес обробки даних, що надходять під час проходження тестових завдань та по його завершенню, а саме процеси отримання кодованих зображень та результатів проходження тесту.

```
Router.post('/load-photo', auth, async (req, res) => {
```

Отримуючи кодоване зображення декодуємо його та збережемо в директорію photo, в якості імені використовуючи випадково згенерований код, що генерується модулем uuid.

```

Try {
  const data = req.body

  fs.writeFile(`photo/${v4()}.png`, data.data, {encoding: 'base64'}, function(err){})

  } catch (err) {

```

```

    res.status(500).json({ message: 'Щось пішло не так, як треба,
спробуйте знову' })

  }

})

router.post('/save-result', auth, async (req, res) => {

  try {

```

При отриманні результатів тесту об'єднуємо назву тесту та результати тесту в єдиний масив для зручності, після чого за допомогою методу `push()` і `save()` відправляємо та зберігаємо дані на сервері

```

    const {test, userId, allResults} = req.body

    const studentResult = [test.name, allResults]

    const student = await Student.findOne({_id: userId})

    student.testResult.push(studentResult)

    await student.save()

    res.json(student)

  } catch (e) {

    res.status(500).json({ message: 'Щось пішло не так, як треба,
спробуйте знову' })})})

```

В результаті отримуємо наступну сторінку до початку проходження тесту, де ми не може обирати відповіді

Студент Владислав Кобзар

Дисципліни Результати Вийти

Новий тест

Питання №1

☐ Відповідь №1
 ☐ Відповідь №2
 ☐ Відповідь №3

Питання №2

☐ Відповідь №1
 ☐ Відповідь №2
 ☐ Відповідь №3

Питання №3

☐ Відповідь №1
 ☐ Відповідь №2
 ☐ Відповідь №3

ЗАВЕРШИТИ ТЕСТ

РОЗПОЧАТИ ВІДЕО

Щоб розпочати проходження тесту, увімкніть відео

Рисунок 3.17 – Сторінка проходження тесту до початку проходження тесту



Рисунок 3.18 – Сторінка проходження тесту під час проходження тесту

Після проходження тестового завдання потрібно створити сторінку з виведенням результатів, створимо її

```
export const Results = () => {
```

```
  const {token, userId} = useContext(AuthContext)
```

```
  const {request, loading} = useHttp()
```

```
  const [results, setResults] = useState()
```

Функція запиту даних

```
  const getResults= useCallback(async () => {
```

```
    try {
```

```
      const fetched = await request(`/api/student-  
page/${userId}/results`, 'GET', null, {Authorization: `Bearer ${to  
ken}`})
```

```
      setResults(fetched)
```

```
    } catch (e) {}
```

```
  }, [token, userId, request])
```

```
  useEffect(() => {
```

```
    getResults()
```

```
  }, [getResults])
```

```
  if(loading || results === undefined){
```



```

    return <Loader />
  }
  return(
    <><div className="row">
      <table className="col s6 offset-s3">
        <thead>
          <tr>
            <th>Назва тесту</th>
            <th>Оцінка</th>
            <th className="test-result">Правильні відповіді</th>
          </tr>
        </thead>
        <tbody>

```

Виводимо отримані данні

```

    {results.testResult.map((test, i) => {
      return(
        <tr key={i+1} >
          <td>{test[0]}</td>
          <td>{test[1][0]}</td>
          <td className="test-
result">{test[1][1]} з {test[1][2]}</td>
        </tr>))) }
      </tbody>
    </table>
  </div></>)}

```

Отримання даних на сервері виглядає наступним чином

```

router.get('/:id/results', auth, async (req, res) => {
  try {
    res.json(await Student.findById(req.params.id))
  }

```

```

    } catch (e) {

        res.status(500).json({ message: 'Щось пішло не так, спробуйте знову' }) }) }) })
    }
}

```

Як результат отримаємо наступну сторінку з результатами

Студент Владислав Кобзар			Дисципліни	Результати	Вийти
Назва тесту	Оцінка	Правильні відповіді			
Новий тест	0.00	0 з 2			
Новий тест	0.00	0 з 2			
Новий тест	0.67	2 з 3			
Новий тест	1.00	3 з 3			
Новий тест	0.00	0 з 3			
Новий тест	0.67	2 з 3			
Тест	1.00	1 з 1			
Новий тест	0.00	0 з 3			

Рисунок 3.20 – Сторінка результатів пройдених тестів для певного студента

3.3 Процес створення тестового завдання та його проходження

Продемонструємо можливості процес створення та проходження тестового завдання в створеній автоматизованій системі оцінки знань студентів.

В першу чергу викладачу порібно обрати одну з наданих йому дисциплін, в нашому випадку оберемо дисципліну з назвою “Нова дисципліни”.

Викладач Сергій Цибульник			Дисципліни	Вийти
№	Назва	Сторінка дисципліни		
1	Тестова назва	Перейти		
2	Нова дисципліна	Перейти		
3	Тестова назва - 2	Перейти		

Рисунок 3.21 – Список дступних дисциплін для викладача

Перейшовши на сторінку дисципліни розпочнемо створення тестовго завдання.

Демонстрація тестового завдання

Ідентифікатор - це

☐ блок коду, який компілятор ігнорує при компіляції

☐ особливим чином оформлений коментар до об'єкту програми

☒ Правильної відповіді немає

*Після введення відповідей на питання не забудьте обрати правильну відповідь

Рисунок 3.22 – Процес створення тестового завдання

Як результат отримаємо сторінку тесту з питаннями, варіантами відповідей та правильними відповідями до питання.

Викладач Сергій Цибульник		Дисципліни	Вийти
<p>Демонстрація тестового завдання</p> <p>Ідентифікатор - це</p> <ol style="list-style-type: none"> 1. блок коду, який компілятор ігнорує при компіляції 2. особливим чином оформлений коментар до об'єкту програми 3. Правильної відповіді немає <p>Правильна відповідь: Правильної відповіді немає</p> <p>Коментар - це</p> <ol style="list-style-type: none"> 1. блок коду, який компілятор ігнорує при компіляції 2. спеціальна символна комбінація, яка використовуються у функціях введення і виведення інформації 3. набір символів, які не мають графічного відображення, але використовуються для керування пристроями та організації даних <p>Правильна відповідь: блок коду, який компілятор ігнорує при компіляції</p> <p>Що з переліченого не є роздільником?</p> <ol style="list-style-type: none"> 1. . (крапка) 2. Правильної відповіді немає 3. , (кома) <p>Правильна відповідь: Правильної відповіді немає</p> <p>До якого типу належить byte?</p> <ol style="list-style-type: none"> 1. Цілочисельні 2. З плаваючою точкою 3. Символьні <p>Правильна відповідь: Цілочисельні</p> <p>Літерал - це</p> <ol style="list-style-type: none"> 1. блок коду, який компілятор ігнорує при компіляції 2. спеціальна символна комбінація, яка використовуються у функціях введення і виведення інформації 3. це синтаксичний елемент, який представляє значення <p>Правильна відповідь: це синтаксичний елемент, який представляє значення</p>			

Рисунок 3.23 – Сторінка створеного тесту

Щоб виконати тестове завдання потрібно авторизуватися під обліковим доступом, що має рівень доступу “студент”, та перейти на сторінку з доступним тестовим завданням.

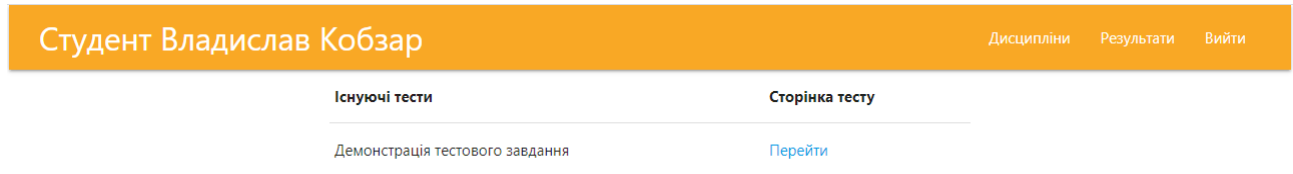


Рисунок 3.23 – Сторінка доступних тестів для студента

Відкриємо сторінку тестового завдання, де побачимо питання з варіантами відповіді та кнопку з назвою “Розпочати відео”, натискання якої розпочинає відео трансляцію, під час якої робляться знімки студент, та надає можливість обирати відповіді на питання

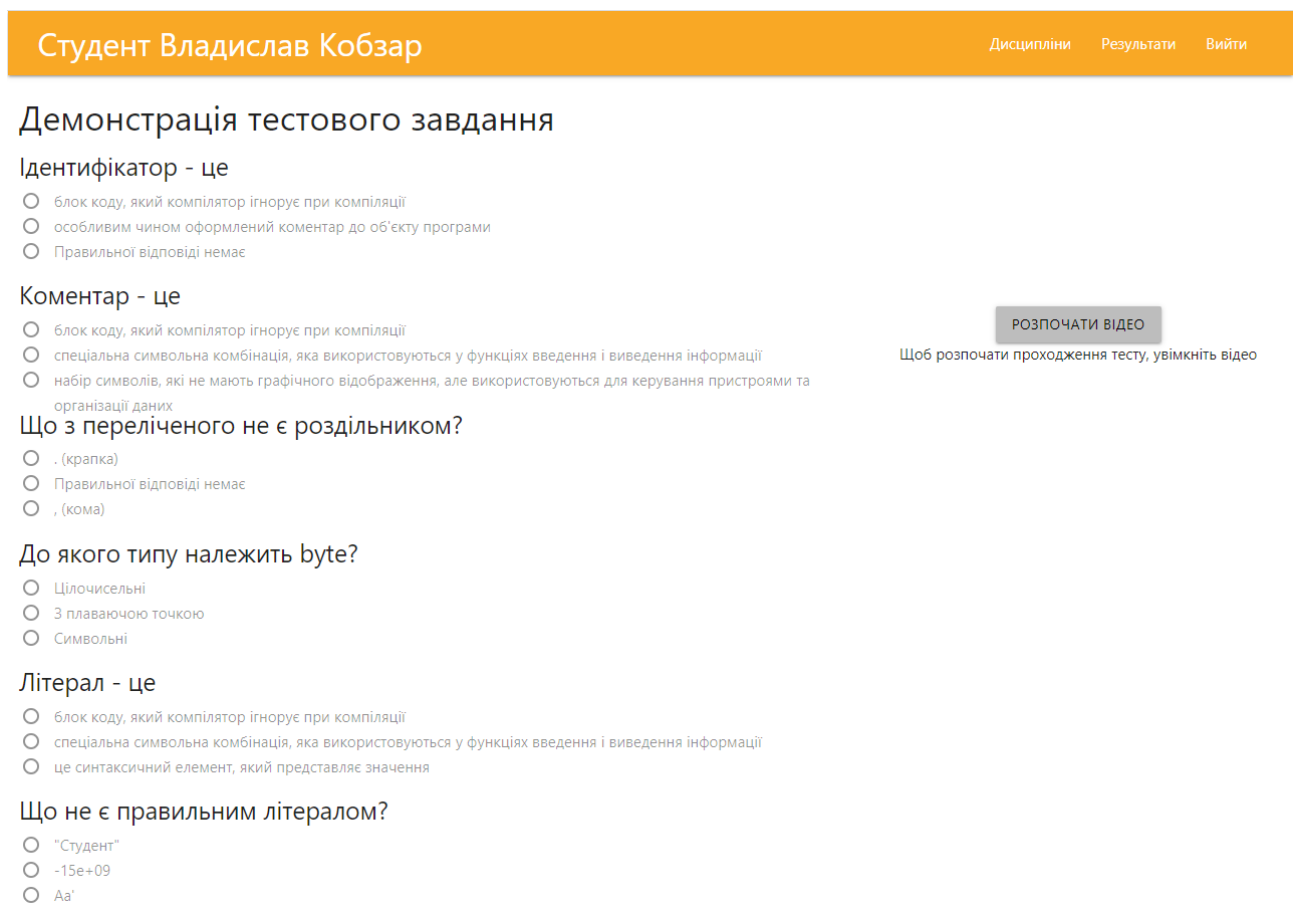


Рисунок 3.24 – Сторінка тестового завдання до натискання на кнопку “Розпочати відео”

Демонстрація тестового завдання

Ідентифікатор - це

- ☐ блок коду, який компілятор ігнорує при компіляції
- ☐ особливим чином оформлений коментар до об'єкту програми
- ☐ Правильної відповіді немає

Коментар - це

- ☐ блок коду, який компілятор ігнорує при компіляції
- ☐ спеціальна символічна комбінація, яка використовується у функціях введення і виведення інформації
- ☐ набір символів, які не мають графічного відображення, але використовуються для керування пристроями та організації даних

Що з переліченого не є роздільником?

- ☐ . (крапка)
- ☐ Правильної відповіді немає
- ☐ , (кома)

До якого типу належить byte?

- ☐ Цілочисельні
- ☐ З плаваючою точкою
- ☐ Символьні

Літерал - це

- ☐ блок коду, який компілятор ігнорує при компіляції
- ☐ спеціальна символічна комбінація, яка використовується у функціях введення і виведення інформації
- ☐ це синтаксичний елемент, який представляє значення

Що не є правильним літералом?

- ☐ "Студент"
- ☐ -15e+09
- ☐ Aa'



РОЗПОЧАТИ ВІДЕО

Щоб розпочати проходження тесту, увімкніть відео

Рисунок 3.25 – Сторінка тестового завдання після натискання на кнопку “Розпочати відео”

Після закінчення вибору відповідей на питання потрібно натиснути на кнопку, що знаходиться після тестових запитань та має назву “Завершити тест”.

Керуючий символ - це

- ☐ блок коду, який компілятор ігнорує при компіляції
- ☒ спеціальна символічна комбінація, яка використовується у функціях введення і виведення інформації
- ☐ Правильної відповіді немає

Які типи даних не належать до простих?

- ☐ int
- ☒ boolean
- ☐ Правильної відповіді немає

Який з наведених символів не є символом перетворення?

- ☐ a
- ☒ b
- ☐ Правильної відповіді немає

Який символ перетворення необхідно використати, щоб рядкову змінну вивести у верхньому регістрі?

- ☒ S
- ☐ c
- ☐ s

ЗАВЕРШИТИ ТЕСТ

Рисунок 3.26 – Сторінка тестового завдання після вибору відповідей на питання

Знімки, що робляться під час тестового завдання зберігаються в відповідну директорию, та мають наступний вигляд (рис. 3.27)

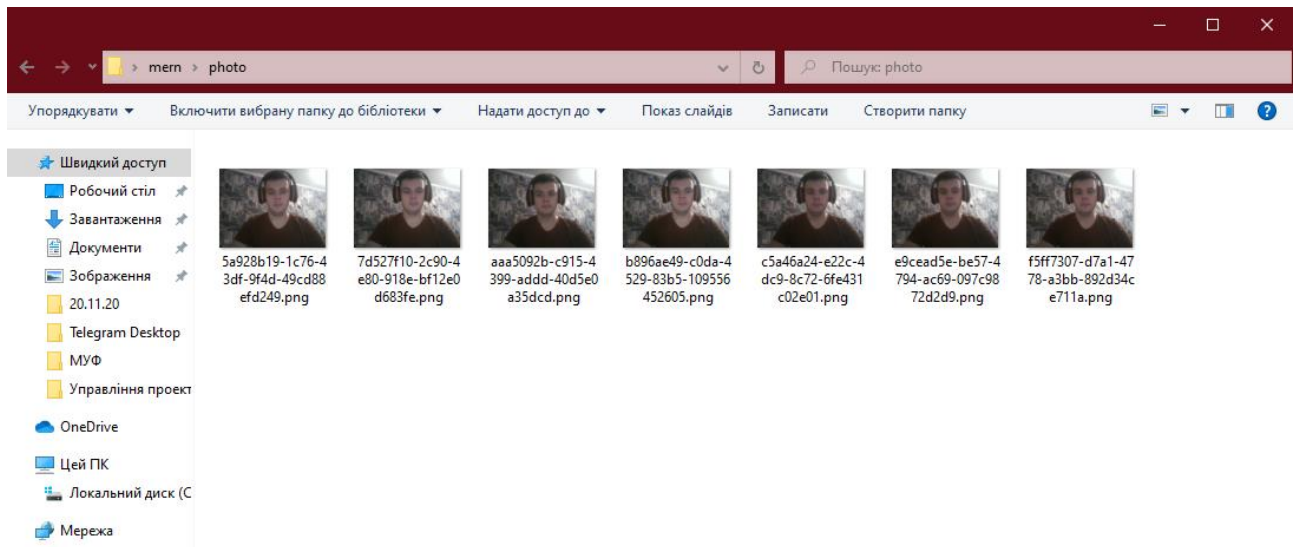


Рисунок 3.27 – Директорія зі знімками, що підтверджують особистість студента, що проходив тестування

Також студент має можливість переглянути результати свого тестування, перейшовши в меню “Результати”.

Студент Владислав Кобзар			Дисципліни	Результати	Вийти
Назва тесту	Оцінка	Правильні відповіді			
Новий тест	0.00	0 з 2			
Новий тест	0.00	0 з 3			
Новий тест	0.00	0 з 3			
Новий тест	0.00	0 з 3			
Демонстрація тестового завдання	0.64	7 з 11			

Рисунок 3.28 – Сторінка з результатами пройдених тестів

В свою чергу обов’язково необхідно, що викладач також мав доступ до результатів студентів. Дана можливість наведена на рис. 3.29

Викладач Сергій Цибульник

ДисципліниВийти

Дисципліна - Нова дисципліна

Групи: ПГ-91-мп

Існуючі тести

Сторінка тесту

Демонстрація тестового завдання

Перейти

СТВОРИТИ ТЕСТ

Результати тестів

Ім'я	Прізвище	Назва	Оцінка	Результат
Владислав	Кобзар	Новий тест	0.00	0 з 2
Владислав	Кобзар	Новий тест	0.00	0 з 3
Владислав	Кобзар	Новий тест	0.00	0 з 3
Владислав	Кобзар	Новий тест	0.00	0 з 3
Владислав	Кобзар	Демонстрація тестового завдання	0.64	7 з 11

Рисунок 3.29 – Сторінка викладача з результатами пройдених тестів відповідними студентами

Висновок до розділу 3

В даному розділі було описано процес створення серверної частини додатку та налаштовано підключення до бази даних та бібліотеки React.

Також було продемонстровано етапи створення систем та підсистем автоматизованої системи оцінки знань студентів, а саме:

- Реалізовано можливість реєстрування для адміністрації та відповідно авторизації.
- Створено підсистему для створення дисциплін, викладачів та студентів адміністрацією
- Створено конструктор тестів для користувачів, що мають рівень доступу “Викладач”
- Реалізовано систему підтвердження особистості студентів під час проходження тестового завдання

Як результат було проведена демонстрація функціональних можливостей автоматизованої системи оцінки знань студентів.

РОЗДІЛ 4

РОЗРОБКА СТАРТАП ПРОЕКТУ «АВТОМАТИЗОВАНА СИСТЕМА ОЦІНКИ ЗНАНЬ СТУДЕНТІВ»

Стартап – це нещодавно створена компанія (можливо ще не зареєстрована офіційно), що формує свій бізнес на основі інновацій або інноваційних технологій, володіє обмеженою кількістю ресурсів (як людських так і фінансових) і планує виходити на ринок [25].

В даному розділі розроблена проектна пропозиція для проекту, який був створений в рамках магістерської дисертації. Даний проект представляє собою систему оцінки знань студентів з поліпшеною системою підтвердження особистості студента.

4.1 Опис ідеї проекту

Ідея автоматизованої системи оцінки знань студентів полягає в створенні системи, яка б дозволила розподілити користувачів системи на три різні типи: адміністрація, викладачі, студенти. Кожний тип користувача отримує доступ до відповідних функціональних можливостей системи, де ключовим моментом є процес проходження тестових завдань студентами, під час яких вони додатково підтверджують свою особистість шляхом знімків, що робляться камерою пристрою, на якому відбувається процес проходження тесту.

Основними користувачами системи виступають заклади освіти, які можуть використовувати систему для оцінки набутих знань студентів або ж учнів під час дистанційного навчання в зручний для них час.

В таблиця 4.1 відображено зміст ідеї та відповідна цільова аудиторія, в якій є потреба у використанні подібної системи.

Таблиця 4.1 Опис ідеї стартап-проекту

<i>Зміст ідеї</i>	<i>Напрямки застосування</i>	<i>Вигоди для користувача</i>
Автоматизована системи оцінки знань студентів	Заклади освіти	Система дозволяє додатково підтверджувати особистість студентів, що разом з результатами проходження тестових завдань демонструє рівень знань студентів
	Власники бізнесу	При найманні нових працівників, кандидатам потрібно пройти тестове завдання, під час якого буде підтверджено особистість кандидата, що в результаті продемонструє його знання та збереже час для рекрутерів.

Висновок: в таблиці приведені основні напрямки застосування, в який можна використовувати систему оцінки знань. Користувачами системи виступають заклади освіти та власники бізнесу.

Визначимо перелік техніко-економічних властивостей та характеристик ідеї, попереднє коло конкурентів або товарів замінників чи товарів-аналогів, що вже існують на ринку, та проведемо збір інформації щодо значень техніко-економічних показників для ідеї власного проекту та проектів-конкурентів. Отримані дані відобразимо в таблиці 4.2

Таблиця 4.2 Визначення сильних, слабких та нейтральних характеристик

№ n/n	Техніко- економічні характеристи ки ідеї	(потенційні) товари/концепції конкурентів				W (слабка сторона)	N (нейтр альна сторо на)	S (сильна сторон а)
		Мій проект	Конку рент1 (Kahoot)	Конкурент 2 (Moodle)	Конку рент3 (G. classroom)			
1.	Популярність проекту	Відсутня	Популя риний	Популяри ний	Популяри ний	+		
2.	Різноманітніс ть вибору створюваних завдань	Невелика	Невели ка	Велика	Середній		+	
3.	Система підтвердженн я особистості	Присутня	Відсут ня	Відсутня	Відсутня			+
4.	Кросс- браузерна робота системи	Присутня	Присут ня	Присутня	Присутня		+	

Висновок: у порівнянні із головними конкурентами товар має перевагу в тому, що дозволяє підтверджувати особистість студента, що проходить тестове завдання

4.2 Технологічний аудит ідеї проекту

В межах даного підрозділу проведемо аудит технологій, за допомогою яких можна реалізувати ідею проекту.

Визначення технологічної здійсненності ідеї проекту передбачає аналіз таких складових (табл. 4.3):

- за якою технологією буде виготовлено товар згідно ідеї проекту?
- чи існують такі технології, чи їх потрібно розробити/додати?
- чи доступні такі технології авторам проекту?

Таблиця 4.3 Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
		Технологія 1 (технологія виготовлення товару, надання послуги)	Чи вони наявні, або ж необхідно їх розробити/додати?	Чи вони доступні авторам проекту?
1.	Можливість створення адміністрацією нових користувачів системи	Використання створеної технології	Наявна	Доступна
2.	Інструмент для створення тестових завдань	Використання створеної технології	Наявна	Доступна
3.	Розширення спектру тестів для конструктору тестів	Використання існуючих технологій для розширення спектру	Необхідно розробити	Доступна
4.	Додаткове підтвердження особистості під час проходження тестового завдання	Використати створену технологію	Наявна	Доступна
Обрана технологія реалізації ідеї проекту: На даний момент ідея вже реалізована, проте, для того, щоб бути конкурентоспроможним на ринку освіти варто розширити вже наявний функціонал та використати “Додаткове підтвердження особистості під час проходження тестового завдання”, що надасть певну перевагу над конкурентами.				

4.3 Аналіз ринкових можливостей запуску стартап-проекту

Рентабельність — поняття, що характеризує економічну ефективність виробництва, за якої підприємство за рахунок грошової виручки від реалізації продукції (робіт, послуг) повністю відшкодовує витрати на її виробництво й одержує прибуток як головне джерело розширеного відтворення [26].

Для кількісного виміру рентабельності в цілому по аграрних підприємствах використовують такі три традиційні показники: рівень рентабельності, норму прибутку і приведену до земельної площі масу прибутку. Рівень рентабельності (R) визначається за формулою [26]:

$$R = \frac{\Pi}{Вв} \cdot 100\% \quad (4.1)$$

де Π — валовий прибуток від реалізації (робіт, послуг);

$Вв$ — виробничі витрати на реалізовану продукцію (її виробнича собівартість).

Для повнішої уяви про реальну ефективність певного виду товарної продукції доцільно цей показник обчислювати з врахуванням витрат на її збут, зменшивши при цьому валовий прибуток на величину цих витрат і водночас збільшивши на них знаменник формули.

Таблиця 4.4 Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	3
2	Загальний обсяг продаж, грн/ум.од	1000
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Конкуренти, що тривалий час знаходяться на ринку
5	Специфічні вимоги до стандартизації та сертифікації	Відсутні
6	Середня норма рентабельності в галузі (або по ринку), %	30%

Висновок: за результатами складеної таблиці можна сказати, що вихід на ринок є рентабельним. Невелика кількість головних гравців свідчить про великий поріг входу на ринок через особливості даного ринку, проте акцентуючи

увагу на особливості запропонованого продукту та компенсувавши в майбутньому недоліки може надати переваги над конкурентами.

Визначимо потенційні групи клієнтів, їх характеристики та сформуємо перелік вимог до товару відповідно кожної групи клієнтів, дані занесемо в таблицю 4.5.

Таблиця 4.5 Характеристика потенційних клієнтів стартап-проекту

<i>№ п/п</i>	<i>Потреба, що формує ринок</i>	<i>Цільова аудиторія (цільові сегменти ринку)</i>	<i>Відмінності у поведінці різних потенційних цільових груп клієнтів</i>	<i>Вимоги споживачів до товару</i>
	Перевірка знань в дистанційному режимі з додатковим підтвердженням особистості	1. Заклади освіти 2. Власники бізнесу	Особливість полягає в тому, з якою метою та для яких груп людей буде застосовуватися система оцінки знань	- простота експлуатації; - стабільність експлуатації

Висновки: формування ринку визначається потребою в системі оцінювання знань, що дає можливість додатково підтверджувати особистість під час проходження тестових завдань.

Сильні сторони підприємства - те, в чому підприємство досягло успіху або якась особливість, що надає додаткові можливості. Сила може полягати в наявному досвіді, доступі до унікальних ресурсів, наявності передової технології і сучасного устаткування, високої кваліфікації персоналу, високій якості продукції, що випускається, популярності торгової марки і т. п [26].

Слабкі сторони підприємства - це відсутність чогось важливого для функціонування підприємства або щось, що поки не вдається в порівнянні з іншими компаніями і ставить підприємство в несприятливе положення. Як приклад слабких сторін можна привести дуже вузький асортимент товарів, що випускається, погану репутацію компанії на ринку, недолік фінансування, низький рівень сервісу і т.п [26].

Ринкові загрози - події, настання яких може несприятливо вплинути на підприємство. Приклади ринкових загроз: вихід на ринок нових конкурентів, зростання податків, зміна смаків покупців, зниження народжуваності й т. п [26].

Таблиця 4.6 Фактори загрози

<i>№ n/n</i>	<i>Фактор</i>	<i>Зміст загрози</i>	<i>Можлива реакція компанії</i>
1.	Конкуренція на ринку	Більша популярність продукції з вже наявним стабільним продуктом	Відмова від продукту
2.	Кібер-атаки на систему	Здійснення DDOS-атак на сервер, або витік даних користувачів	Відтік клієнтів
3.	Створення систем з аналогічним функціоналом	Поява на ринку систем з аналогічним функціоналом	Відтік клієнтів

Висновки: головними факторами загрози є конкуренція на ринку кібер-атаки, що впливають на працездатність системи. В свою чергу конкуренти можуть реалізувати функціонал аналогічний до створеного.

Ринкові можливості - це сприятливі обставини, які підприємство може використовувати для отримання переваг. Як приклад ринкових можливостей можна привести погіршення позицій конкурентів, різке зростання попиту, появу нових технологій виробництва продукції, зростання рівня доходів населення і т. п. Слід зазначити, що можливостями з погляду SWOT-аналізу є не всі можливості, які існують на ринку, а тільки ті, які можна використовувати [26].

Таблиця 4.7 Фактори можливостей

<i>№ n/n</i>	<i>Фактор</i>	<i>Зміст можливості</i>	<i>Можлива реакція компанії</i>
1.	Збільшення попиту	Різне збільшення зацікавленості до системи	Розширення поточного функціоналу
2.	Розширення функціоналу системи	Зростання лояльності клієнтів	Покращення сервісу
3.	Вихід на іноземний ринок	Збільшення кількості клієнтів	Розширення клієнтської бази

Висновок: Наразі ринок дистанційного навчання зростає, тобто зростає аудиторія, що користується даними системами, в свою чергу це призводить до зацікавленості нових систем, що виходять на ринок, та позитивно впливає на їх розвиток.

Проведемо аналіз пропозицій (табл. 4.8), де визначимо загальні риси конкуренції на ринку.

Таблиця 4.8 Ступеневий аналіз конкуренції на ринку

<i>Особливості конкурентного середовища</i>	<i>В чому проявляється дана характеристика</i>	<i>Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)</i>
1. Вказати тип конкуренції - досконала конкуренція	Конкуренти не мають прямого впливу один на одного	Надавати клієнтам продукт, що буде в повній мірі задовольняти їх потреби
2. За рівнем конкурентної боротьби - міжнародний	Продукт легко пристосувати до потреб інших країн	Міжнародний ринок
3. За галузевою ознакою - міжгалузева	Використання в різних галузях	Підвищення рівня пізнаваності продукту за рахунок реклами
4. Конкуренція за видами товарів: - товарно-родова	Продукт конкурує з продуктами, що надають схожий функціонал	Адаптивність до умов ринку
5. За характером конкурентних переваг - нецінова	Залежить від якості продукту	Постійно покращувати продукт
6. За інтенсивністю - марочна	Вибір продукту залежить від бренду	Створити репутацію надійного бренду

Висновок: Провівши аналіз пропозицій було виявлено, що на ринку присутня досконала конкуренція. Попит на продукт на пряму залежить від його якості та відповідно репутації.

Після аналізу конкуренції проведемо більш детальний аналіз умов конкуренції за допомогою моделі 5 сил М.Портера [27]

Аналіз п'яти сил Портера - це корисний інструмент стратегічного планування як для бізнес-планування, так і для ринкового планування, особливо

якщо мова йде про розуміння рівня конкурентоспроможності бізнесу в певній галузі [27].

Аналіз п'яти сил Портера оцінює рівень рентабельності, можливостей та ризику на основі 5 ключових факторів у галузі [27].

5 факторів:

- Постачальники.
- Покупці.
- Перешкоди для входу / виходу.
- Замінники.
- Суперництво.

Навіщо потрібен цей інструмент:

- Проаналізувати привабливість та прибутковість галузі.
- Спостерігати за силою ринкової позиції власного бізнесу.

Як користуватися цим інструментом:

Ця модель допомагає визначити конкурентні сили, які існують у галузі. У свою чергу ці сили допомагають визначити привабливість та прибутковість галузі.

Сила 1. Потужність постачальника (ця сила аналізує владу, яку має постачальник над бізнесом)

- Подивіться, скільки постачальників є на ринку.
- Скільки у вас постачальників.
- Чи тримають ваші постачальники владу над типом вашого бізнесу.
- Що буде вартувати вам і їм, якщо ви вирішили змінити постачальників.
- Чи багато постачальників, які контролюють ціни на ринку.

Сила 2. Потужність покупця (ця сила аналізує владу, яку покупець має над вашим бізнесом)

- Скільки всього покупців.

- Скільки у вас покупців.
- Наскільки чутливими до ціни є ваші покупці.
- Яку інформацію ви маєте про них.

Сила 3. Загроза нових учасників (ця сила аналізує, наскільки легко чи складно для нових конкурентів вийти на ринок).

- Наскільки легко відкрити бізнес на вашому ринку.
- Яких правил та положень потрібно дотримуватися.
- Скільки грошей повинен витратити новий стартап, щоб зайти на ринок.
- Чи є бар'єри, при здоланні яких, ви б отримали більший вплив.

Сила 4 Загроза заміни товарів / послуг (ця сила аналізує, як легко клієнту перейти з продукту одного бізнесу на продукт конкурента).

- Скільки замінників вашого товару існує.
- Наскільки легко вашому покупцеві перейти на інший товар.
- Чи покупець “сплачує” за перемикання на інший товар (чи це щось йому вартує).

Сила 5 Конкурентне суперництво (ця сила вивчає інтенсивність конкуренції на поточному ринку).

- Який рівень конкуренції у вашому ринковому секторі.
- Хто є вашими основними конкурентами.
- Приблизно скільки у вас конкурентів.
- Яка ваша конкурентна стратегія.

Таблиця 4.9 Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Moodle, Classroom	Kahoot	Сервіси, що надають в оренду сервера для веб сайтів	Заклади освіти, власники бізнесу	Продукти, що надаються конкурентами
Висновки:	Конкуренція ж	Присутня можливості входу в ринок.	На даний момент присутня велика кількість	Клієнти не можуть	Наявність великих конкурентів на

	відносно невеликою	Присутні потенційні конкуренти	представників даних послуг, що не дозволяє диктувати їм умови на ринку	диктувати умови	ринку певною мірою ускладнить вихід на ринок
--	--------------------	--------------------------------	--	-----------------	--

На основі аналізу конкуренції, проведеного в (табл. 4.9), а також із урахуванням характеристик ідеї проекту (табл. 4.2), вимог споживачів до товару (табл. 4.5) та факторів маркетингового середовища (табл. 4.6, 4.7) визначимо та обґрунтуємо перелік факторів конкурентоспроможності. Аналіз оформимо в табл. 4.10.

Таблиця 4.10 Обґрунтування факторів конкурентоспроможності

№ n/n	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1.	Простота	Простота використання для нових користувачів
2.	Адаптивність	Система гнучка до змін
3.	Крос-платформленість	Можливість використання продукту за допомогою різних видів девайсів
4.	Підтвердження особистості	Використання поліпшеної системи підтвердження особистості

Висновки: оцінено основні фактори конкурентоспроможності. Простота використання дозволяє швидше ознайомлюватися новим користувачам з продукцією. Адаптивність в свою чергу дозволить швидко реагувати на потребу клієнтів. В сукупності всі фактори дають можливість бути конкурентоспроможним на ринку.

За визначеними факторами конкурентоспроможності (табл. 10) проведемо аналіз сильних та слабких сторін стартап-проекту (табл. 4.11).

Таблиця 4.11 Порівняльний аналіз сильних та слабких сторін проекту

№ n/n	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з "АСОЗС"						
			-3	-2	-1	0	+1	+2	+3
1	Простота	19							+
2	Адаптивність	18						+	

3	Крос-браузерність	18						+	
4	Підтвердження особистості	20							+

Висновки: Беручи до уваги дані з табл. 4.11 та 4.10 можна підсумувати, що продукт матиме достатньо переваг, щоб бути конкурентоспроможним на ринку з іншими продуктами.

Фінальним етапом ринкового аналізу можливостей впровадження проекту є складання SWOT-аналізу (матриці аналізу сильних (Strength) та слабких (Weak) сторін, загроз (Troubles) та можливостей (Opportunities) (табл. 4.12) на основі виділених ринкових загроз та можливостей, та сильних і слабких сторін (табл. 4.12).

Перелік ринкових загроз та ринкових можливостей складається на основі аналізу факторів загроз та факторів можливостей маркетингового середовища. Ринкові загрози та ринкові можливості є наслідками (прогнозованими результатами) впливу факторів, і, на відміну від них, ще не є реалізованими на ринку та мають певну ймовірність здійснення. Наприклад: зниження доходів потенційних споживачів – фактор загрози, на основі якого можна зробити прогноз щодо посилення значущості цінового фактору при виборі товару та відповідно, – цінової конкуренції (а це вже – ринкова загроза).

Таблиця 4.12 SWOT-аналіз стартап-проекту

Сильні сторони:		Слабкі сторони:	
1.	Простота	1.	Невеликий асортимент видів створюваних тестів
2.	Адаптивність		
3.	Крос-платформленість(браузерність)		
4.	Підтвердження особистості		
Можливості:		Загрози:	
1.	Збільшення попиту	1.	Конкуренція на ринку

Продовження таблиці 4.12

2.	Розширення функціоналу системи	2.	Кібер-атаки на систему
3.	Вихід на іноземний ринок	3.	Створення систем з аналогічним функціоналом

На основі SWOT-аналізу розробимо альтернативну ринкову поведінку для виведення стартап-проекту на ринок та орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проекти конкурентів, що можуть бути виведені на ринок (див. табл. 4.9, аналіз потенційних конкурентів).

Визначені альтернативи аналізуються з точки зору строків та ймовірності отримання ресурсів (табл. 4.13).

Таблиця 4.13 Альтернативи ринкового впровадження стартап-проекту

<i>№ п/п</i>	<i>Альтернатива (орієнтовний комплекс заходів) ринкової поведінки</i>	<i>Ймовірність отримання ресурсів</i>	<i>Строки реалізації</i>
1.	Використати готовий MVP для демонстрації можливостей	Ресурси не потрібні для даного варіанту	Реалізований
2.	Розширити асортимент для створення тестів, поліпшити функціонал решти додатку	Необхідно доробити існуючу архітектуру, створити новий дизайн елементів сайту	40-60 днів

Висновки: Кращим варіантом буде використання другого варіанту, оскільки такий варіант складе більш позитивне враження на таких клієнтів.

4.4 Розроблення ринкової стратегії та маркетингової програми проекту

Визначимо стратегію охоплення ринку, опишемо цільові групи потенційних споживачів.

Таблиця 4.14 Вибір цільових груп потенційних споживачів

<i>№ п/п</i>	<i>Опис профілю цільової групи потенційних клієнтів</i>	<i>Готовність споживачів сприйняти продукт</i>	<i>Орієнтовний попит в межах цільової групи (сегменту)</i>	<i>Інтенсивність конкуренції в сегменті</i>	<i>Простота входу у сегмент</i>
1.	Заклади освіти	Висока	Високий	Висока	Висока
2.	Власники бізнесу	Висока	Високий	Низька	Середня
Які цільові групи обрано: Було обрано Заклади освіти, оскільки саме в їх інтересах контролювати якість знань, що вони надають.					

Для роботи в обраних сегментах ринку необхідно сформулювати базову стратегію розвитку (табл. 4.15).

Існують три базові стратегії розвитку, що відрізняються за ступенем охоплення цільового ринку та типом конкурентної переваги, що має бути реалізована на ринку.

1. Стратегія лідерства по витратах. Передбачає, що компанія за рахунок чинників внутрішнього і/або зовнішнього середовища може забезпечити більшу, ніж у конкурентів маржу між собівартістю товару і середньоринковою ціною (або ж ціною головного конкурента).

2. Стратегія диференціації. Передбачає надання товару важливих з точки зору споживача відмінних властивостей, які роблять товар відмінним від товарів конкурентів.

3. Стратегія спеціалізації. Передбачає концентрацію на потребах одного цільового сегменту, без прагнення охопити увесь ринок.

Таблиця 4.15 Визначення базової стратегії розвитку

<i>№ п/п</i>	<i>Обрана альтернатива розвитку проекту</i>	<i>Стратегія охоплення ринку</i>	<i>Ключові конкурентоспроможні позиції відповідно до обраної альтернативи</i>	<i>Базова стратегія розвитку</i>
	Індивідуалізм	Стратегія диференціального маркетингу	- адаптація до потреб користувачів. - створення нових відмінностей від конкурентів	Стратегія диференціації

Висновки: Обрана стратегія диференціації через існування на ринку сильних конкурентів. Мета стратегії полягає в надані клієнтам продукту відмінного від продукту конкурентів.

Наступним кроком є вибір стратегії конкурентної поведінки (табл. 4.16)

Стратегія лідера. Залежно від міри сформованості товарного(галузевого) ринку, характеру конкурентної боротьби компанії-лідери обирають одну з трьох стратегій: розширення первинного попиту, оборонну або наступальну стратегію або ж застосувати демаркетинг або диверсифікацію [25].

Стратегія виклику лідера. Стратегію виклику лідеріві найчастіше вибирають компанії, які є другими, третіми на ринку, але бажають стати лідером ринку. Теоретично, ці компанії можуть прийняти два стратегічні рішення: атакувати лідера у боротьбі за частку ринку або ж йти за лідером.

Стратегія наслідування лідеру. Компанії, що приймають слідування за лідером – це підприємства з невеликою часткою ринку, які вибирають адаптивну лінію поведінки на ринку, усвідомлюють своє місце на ній і йдуть у фарватері фірм-лідерів. Головна перевага такої стратегії – економія фінансових ресурсів, пов'язаних з необхідністю розширення товарного(галузевого) ринку, постійними інноваціями, витратами на утримання домінуючого положення [25].

Стратегія заняття конкурентної ніші. При прийнятті стратегії зайняття конкурентної ніші (інші назви – стратегія фахівця або нішера) компанія в якості цільового ринку вибирає один або декілька ринкових сегментів. Головна особливість – малий розмір сегментів/сегменту. Ця конкурентна стратегія являється похідною від такої базової стратегії компанії, як концентрація.

Таблиця 4.16 Визначення базової стратегії конкурентної поведінки

№ n/n	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або	Чи буде компанія копіювати основні характеристики	Стратегія конкурентної поведінки*
----------	--	---	---	---

		<i>забирати існуючих у конкурентів?</i>	<i>товару конкурента, і які?</i>	
	Ні	Компанія буде як забирати існуючих клієнтів так і шукати нових	Планується розширення видів завдань аналогічні вже існуючи на ринку	Наступальна стратегія

Висновки: Використання наступальної стратегії дозволить компанії збільшити частку на ринку шляхом впровадження інновацій у власний продукт, що дасть в свою чергу перевагу над конкурентами.

На основі вимог споживачів з обраних сегментів до постачальника (стартап-компанії) та до продукту (див. табл. 4.5), а також в залежності від обраної базової стратегії розвитку (табл. 4.15) та стратегії конкурентної поведінки (табл. 4.16) розробляється стратегія позиціонування (табл. 4.17). що полягає у формуванні ринкової позиції (комплексу асоціацій), за яким споживачі мають ідентифікувати торгівельну марку/проект.

Таблиця 4.17 Визначення стратегій позиціонування

<i>№ n/n</i>	<i>Вимоги до товару цільової аудиторії</i>	<i>Базова стратегія розвитку</i>	<i>Ключові конкурентоспроможні і позиції власного стартап-проекту</i>	<i>Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)</i>
1.	Простота товару	Стратегія спеціалізації	Простота використання товару для нових користувачів	Продукт зручний для використання
2.	Якість	Стратегія спеціалізації	Гнучкість продукту	Ціна, якість
3.	Постійне покращення продукту	Стратегія спеціалізації	Постійна підтримка та покращення створеної системи	Легка модернізація елементів системи

Висновки: З врахуванням простоти продукту та інших вимог у користувачів продукт повинен викликати асоціацію у простоті використання та в інноваційності

Першим кроком є формування маркетингової концепції товару, який отримає споживач. Для цього у табл. 4.18 потрібно підсумувати результати попереднього аналізу конкурентоспроможності товару.

Таблиця 4.18 Визначення ключових переваг концепції потенційного товару

<i>№ n/n</i>	<i>Потреба</i>	<i>Вигода, яку пропонує товар</i>	<i>Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)</i>
	Додаткове підтвердження особистості	Система що дозволяє підвищити достовірність знань певної людини	Відсутність в конкурентів даної можливості

Висновок: Використовуючи дану перевагу над іншими конкурентами можна під час проведення рекламної компанії акцентувати увагу саме на цій особливості

Надалі розробляється трирівнева маркетингова модель товару: уточняється ідея продукту та/або послуги, його фізичні складові, особливості процесу його надання (табл. 4.19).

1-й рівень

При формуванні задуму товару вирішується питання щодо того, засобом вирішення якої потреби і / або проблеми буде даний товар, яка його основна вигода. Дане питання безпосередньо пов'язаний з формуванням технічного завдання в процесі розробки конструкторської документації на виріб [28].

2-й рівень

Цей рівень являє рішення того, як буде реалізований товар в реальному/ включає в себе якість, властивості, дизайн, упаковку, ціну [28].

3-й рівень

Товар з підкріпленням (супроводом) - додаткові послуги та переваги для споживача, що створюються на основі товару за задумом і товару в реальному виконанні (гарантії якості , доставка, умови оплати та ін.) [28].

Таблиця 4.19 Опис трьох рівнів моделі товару

<i>Рівні товару</i>	<i>Сутність та складові</i>
---------------------	-----------------------------

I. Товар за задумом	Система оцінки знань студентів представляє собою single page application, що створений з використанням MERN стеку технологій	
II. Товар у реальному виконанні	Властивості/характеристики	
	1. Програмне забезпечення	Надає можливість в короткі терміни покращити продукт
	Продукт представляє собою веб-сайт, що розташовується на сервері з відкритим доступом	
III. Товар із підкріпленням	До продажу Доступ до системи	
	Після продажу Підтримка користувачів під час використання системи	
Товар собою представляє програмний шляхом накладання ліцензії на продукт		

Висновок: основних засобом захисту даного програмного продукту виступатиме ліцензійна угода, що регулюватиме можливість та право використання системи

Наступним кроком є визначення цінових меж, якими необхідно керуватись при встановленні ціни на потенційний товар (остаточне визначення ціни відбувається під час фінансово-економічного аналізу проекту), яке передбачає аналіз ціни на товари-аналоги або товари субституту, а також аналіз рівня доходів цільової групи споживачів (табл. 4.20). Аналіз проводиться експертним методом.

Таблиця 4.20 Визначення меж встановлення ціни

№ n/n	Рівень цін на товари- замінники	Рівень цін на товари- аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
	0 - 10000	0 - 10000	100000	1000-2000

Висновок: Вартість використання продукту залежить від того, яка кількість студентів або працівників буде тестуватися з певний період, чи більша буде кількість від одного адміністратора, тим менша ціна буде за 1 протестовану особу

Наступним кроком є визначення оптимальної системи збуту, в межах якого приймається рішення (табл. 4.21):

- проводити збут власними силами або залучати сторонніх посередників (власна або залучена система збуту);
- вибір та обґрунтування оптимальної глибини каналу збуту;
- вибір та обґрунтування виду посередників.

Таблиця 4.21 Формування системи збуту

<i>№ п/п</i>	<i>Специфіка закупівельної поведінки цільових клієнтів</i>	<i>Функції збуту, які має виконувати постачальник товару</i>	<i>Глибина каналу збуту</i>	<i>Оптимальна система збуту</i>
	Надання доступу до продукту на період, щоб було сплачено користувачем	Повна підтримка товару	Пряма	Пряма

Висновок: Використання прямого каналу збуту надає можливість встановлювати ціну на ринку.

Останньою складовою маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів (табл. 4.22).

Визначення ідеї та теми рекламного звернення зумовлює всі наступні етапи планування рекламної діяльності. Тому цей етап є основним для професіоналів — творчих працівників рекламних агенцій чи рекламних підрозділів підприємств. Один із класиків рекламного бізнесу Д. Огілві радить ретельно вивчити те, що необхідно рекламувати: «Отримавши завдання на рекламування автомобіля «Роллс Ройс», я витратив три тижні на вивчення цієї моделі. І тоді народилася ідея, яка втілилась у рекламному зверненні: на швидкості 60 миль за годину найголосніший звук, який можна почути в машині, — цокання годинника» [29].

Таблиця 4.22 Концепція маркетингових комунікацій

<i>№ п/п</i>	<i>Специфіка поведінки цільових клієнтів</i>	<i>Канали комунікацій, якими користуються цільові клієнти</i>	<i>Ключові позиції, обрані для позиціонування</i>	<i>Завдання рекламного повідомлення</i>	<i>Концепція рекламного звернення</i>
	Зовнішні обставини спонукають клієнтів до пошуку систем, щоб дозволяють оцінювати знання певних осіб	- Сайт з впровадженням продуктом - Соціальні мережі	- Зручність користування - Нова система підтвердження особистості	Розповсюдження інформацію про продукт та формування репутації для бренду	Продемонструвати особливості та переваги даного продукту

Висновки: Рекламна компанія проходить шляхом створення бренду в соціальних мережах та шляхом “холодних дзвінків” закладам освіти.

Висновок до розділу 4

В розділі розглядається стартап-проект для проекту “Автоматизована система оцінки знань студентів”, що реалізований за допомогою стеку технологій MERN та представляє з себе single page application. Дана система дозволяє новим користувачам створювати свою особисту базу з викладачів та студентів. В свою чергу користувачі з рівнем доступу викладача мають можливість створення тестових завдань для відповідних дисциплін. Користувачі з рівнем доступу студент виконують тестові завдання під час чого робляться знімки користувачів за допомогою веб камери, зроблені знімки відправляються на сервер та зберігаються і відповідних директоріях.

В результаті виникнення сумнівів щодо того, чи саме відповідна особа проходила тестові завдання адміністрація завжди матиме доступ до знімків, що демонструють, хто саме проходив завдання.

Основною цільовою аудиторією є заклади освіти, яким потрібно контролювати рівень знань студентів та учнів під час дистанційного навчання, а

також власникам бізнесу, які можуть використовувати систему для оцінки знань нових та потенційних працівників.

Під час проведення аналізу ринку та виявлення прямих та непрямих конкурентів було виявлено, що в усіх системах відсутня можливість додаткового підтвердження особистості людини, що проходить тестове завдання. Дана особливість в значній мірі виокремлює створену систему відносно конкурентів, та є її сильною стороною.

В результаті розробки стартап проекту були визначені слабкі та сильні сторони, а також загрози та можливості, що притаманні системі даного типу. В свою чергу було визначено стратегію розвитку та комунікації з потенційними клієнтами.

Підсумовуючи виконану роботу варто відмітити, щоб система має можливість виходу на ринок, проте попередньо потрібно розширити її функціональну базу та розробити стратегію створення бренду для системи та її популяризації.

ВИСНОВКИ

В даній роботі було розглянуто процес навчання відповідно законодавства України та здійсненого огляд недоліків даного навчання, які виражаються в оцінці знань студентів та підтвердження достовірності знань на дистанційній формі навчання.

В свою чергу було здійсненого огляд існуючих систем оцінки знань, розглянуті їхні переваги та недоліки, та як результат було виокремлено основний недолік, що стосується всіх систем, а саме низький рівень достовірності знань при дистанційному процесі навчання.

Було вирішено створити автоматизовану систему оцінки знань студентів з поліпшеною системою підтвердження особистості студента, що дозволить додатково підтвердити набуті знання під час дистанційного навчання студентів.

Наступним кроком було проведено огляд існуючих засобів для розробки їх переваги та недоліки, як результат було обрано стек технологій MERN, який надав можливість розробити автоматизовану систему оцінки знань студентів.

Після завершення вибору інструментів для розробки були формалізовані умови до архітектури автоматизованої система оцінки знань і було створено, та описано структуру, та архітектуру системи.

Наступним кроком було розроблено автоматизовану систему оцінки знань студентів її системи та підсистеми, а саме:

- Реєстрація та авторизація користувачів;
- Створення нових дисциплін;
- Призначення дисциплін відповідних викладачам та студентам;
- Створення тестових завдань;
- Система додаткового підтвердження особистості студента під час проходження тестового завдання.

Як результат виконаної роботи було продемонстровано всі можливості автоматизованої системи оцінки знань студентів.

Завершальним кроком стала розробка стартап проекту, в результаті чого було визначено, що після певного поліпшення системи вона може бути конкурентноздатною на ринку з іншими конкурентами так як вона має функціонал, що відсутній в інших системах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- 1) Актуальність і проблеми дистанційного навчання Наталія Самолюк, Микола Швець.
- 2) Застосування системи автоматизованого опитування студентів ВНЗ[Електронний ресурс] : матеріали міжвузівського вебінару (м.Вінниця, 15 грудня 2015 р.) / відп. ред. Л.Б.Ліщинська. –Вінниця : ВТЕІ КНТЕУ, 2015. – 141с.
- 3) Модернізація вищої освіти України і Болонський процес: [Матеріали до першої лекції]/М.Ф. Степко, Я. Я.Болубаш, К.М. Левківський та ін. -: НМЦ вищої освіти МОН України, 2004. - 24 с.
- 4) Автоматизоване керування навчанням: Методичний посібник. Під редакцією А.М. Кутєпова -М.:1991. -235 с.
- 5) Умови ефективного застосування дистанційного навчання [Електронний ресурс]. – <http://www.vtei.com.ua/doc/materialuvebinary.pdf>
- 6) Ноздріна Л. В. Исследование результатов e-learning проектов в высшей школе Украины / Л. В. Ноздріна. [Електронний ресурс]. – Режим доступу: <http://ifets.ieee.org/russian/depository>.
- 7) Kahhot! Learning Games [Електронний ресурс]. – Режим доступу: <https://kahoot.com/>
- 8) Як оцінювати не за помилки, а за досягнення [Електронний ресурс]. – Режим доступу:<https://osvitoria.media/experience/yak-otsinyuvaty-ne-za-pomylyky-a-zadosyagnennya/>
- 9) Moodle. Opensource learning platform [Електронний ресурс]. – Режим доступу: <https://moodle.org/>
- 10) Google Classroom [Електронний ресурс]. – Режим доступу: <https://classroom.google.com/>
- 11) iSpring. Course creation software for PowerPoint [Електронний ресурс]. – Режим доступу: <https://www.ispringsolutions.com/ispring-suite>
- 12) Blackboard. Learning management system[Електронний ресурс]. – Режим доступу: <https://www.blackboard.com/teaching-learning/learning-management>

- 13) Adobe Acrobat[Електронний ресурс]. – Режим доступу: <https://acrobat.adobe.com/ua/en/acrobat.html>
- 14) Plickers [Електронний ресурс]. – Режим доступу: <https://get.plickers.com/>
- 15) WEB-Тезаурус [Електронний ресурс]. – Режим доступу: [Електронний ресурс]. – Режим доступу: http://socrates.vsau.org/thesaurus/web_thesaurus_ru.html
- 16) [Електронний ресурс]. – Режим доступу: [Електронний ресурс]. – Режим доступу: <https://itu.foleon.com/itu/measuring-digital-development/home/>
- 17) ITU. Measuring Digital Development [Електронний ресурс]. – Режим доступу: <https://livesotyping.com/ru/blog/pwa-i-amp-uskorenie-dlya-vashego-internet-biznesa>
- 18) Node.js [Електронний ресурс]. – Режим доступу: [Електронний ресурс]. – Режим доступу: <https://nodejs.org/uk/>
- 19) Django project[Електронний ресурс]. – Режим доступу: <https://www.djangoproject.com/>
- 20) Laravel [Електронний ресурс]. – Режим доступу: <https://laravel.com/>
- 21) Angular [Електронний ресурс]. – Режим доступу: <https://angular.io/>
- 22) React.js[Електронний ресурс]. – Режим доступу:<https://ru.reactjs.org/>
- 23) Vue.js[Електронний ресурс]. – Режим доступу: <https://vuejs.org/>
- 24) Моделі життєвого циклу, принципи і методології розробки програмного забезпечення (ПЗ) [Електронний ресурс]. – Режим доступу: <https://evergreens.com.ua/ua/articles/software-development-metodologies.html>
- 25) Розробка стартап-проектів: Конспект лекцій [Електронний ресурс] : навч. посіб. для студ. спеціальностей 151 – «Автоматизація та комп'ютерно-інтегровані технології» та 152 – «Метрологія та інформаційно-вимірювальна техніка» / О. А. Гавриш, К. О. Бояринова, К. О. Копішинська; КПІ ім. Ігоря Сікорського. – Електронні текстові данні (1 файл: X,XX Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2019. – 188 с.
- 26) Рентабельність виробництва і методика визначення її показників . [Електронний ресурс]. – Режим доступу: <http://buklib.net/books/29473/>

- 27) 5 сил Портера [Електронний ресурс]. – Режим доступу:
<https://business.diia.gov.ua/handbook/marketing/5-sil-portera>
- 28) Разработка продуктовой стратегии компании [Електронний ресурс]. –
Режим
доступу:https://pidru4niki.com/19670511/marketing/razrabotka_produktovoy_strategii_kompanii#891
- 29) Маркетингова діяльність [Електронний ресурс]. – Режим доступу:
<http://referat-ok.com.ua/marketing/marketingova-diyalnist-2>