Міністерство освіти і науки України
Рівненський державний гуманітарний університет
Кафедра інформаційних технологій та моделювання

Кваліфікаційна робота

за освітнім ступенем «бакалавр»

на тему:

ВЕБЗАСТОСУНОК ДЛЯ ПЕРЕГЛЯДУ ТА ПОПЕРЕДНЬОГО ВИБОРУ ВИБІРКОВИХ ДИСЦИПЛІН СТУДЕНТАМИ

Виконав:
здобувач IV курсу
групи КН-41
спеціальності 122 «Комп’ютерні науки»
Король Григорій Ігорович

Науковий керівник:
к.п.н., доц. Петренко С. В.

Рівне – 2026

ЗМІСТ

ВСТУП

Актуальність дослідження. Актуальність теми зумовлена потребою цифрової трансформації освітніх процесів у закладах вищої освіти, де вибіркові дисципліни є важливим етапом формування індивідуального освітнього шляху студента. Законодавство України закріплює право здобувача вищої освіти на вибір навчальних дисциплін у межах освітньої програми, а загальний обсяг вибіркової частини має становити не менше ніж 25 відсотків кредитів ЄКТС для відповідного рівня вищої освіти. Це означає, що процес ознайомлення з вибірковими дисциплінами, їх порівняння та попередній вибір є не допоміжною, а структурно значущою складовою освітнього процесу. Разом із тим, у багатьох ЗВО України механізм вибору дисциплін і досі організований фрагментарно: через таблиці, PDF-файли, оголошення, Google-форми або інші розрізнені канали. Такий підхід знижує прозорість процедури, ускладнює доступ до актуальної інформації та створює додаткове адміністративне навантаження. У результаті студент витрачає більше часу на пошук потрібних відомостей, а адміністрація — на збирання, перевірку та узагальнення результатів вибору. Саме тому створення прототипу вебзастосунку для перегляду та попереднього вибору вибіркових дисциплін є обґрунтованим і практично значущим завданням.

Об’єктом дослідження є процес організації вибору вибіркових дисциплін у закладах вищої освіти.

Предметом дослідження є методи та засоби розроблення вебзастосунку для перегляду, попереднього вибору та збереження вибіркових дисциплін студентами.

Метою роботи є розроблення вебзастосунку, який забезпечує зручний перегляд вибіркових дисциплін, авторизацію студентів РДГУ через корпоративний акаунт, попередній вибір дисциплін і збереження результатів у базі даних.

Для досягнення поставленої мети необхідно розв’язати такі завдання:

- проаналізувати нормативно-правові засади реалізації права студентів на вибір дисциплін;
- сформувати функціональні та нефункціональні вимоги до системи;
- обґрунтувати архітектуру та технологічний стек веб застосунку;
- спроєктувати структуру даних і основні API-інтерфейси;
- реалізувати та протестувати прототип системи;
- оцінити можливості його практичного застосування в освітньому процесі РДГУ.

Методами дослідження є аналіз нормативної та науково-методичної літератури, порівняльний аналіз цифрових сервісів, моделювання предметної області, проєктування вебархітектури, програмна реалізація клієнтської і серверної частин, а також функціональне та інтеграційне тестування. Для технічного обґрунтування використано положення сучасної документації React, Express, Mongoose, Google OAuth 2.0, JWT та CORS.

Наукова новизна роботи полягає не у створенні нової теорії, а в адаптації наявних цифрових підходів до предметної області конкретного ЗВО та у формалізації прототипу вебзастосунку як цілісної моделі взаємодії студента й адміністрації під час перегляду та попереднього вибору вибіркових дисциплін.

Практичне значення роботи полягає у створенні функціонального прототипу, який може бути використаний як основа для подальшого впровадження в інформаційну інфраструктуру РДГУ або для розвитку подібних сервісів в інших університетах. Запропоноване рішення потенційно спрощує доступ до інформації про дисципліни, підвищує прозорість вибору та зменшує організаційні витрати.

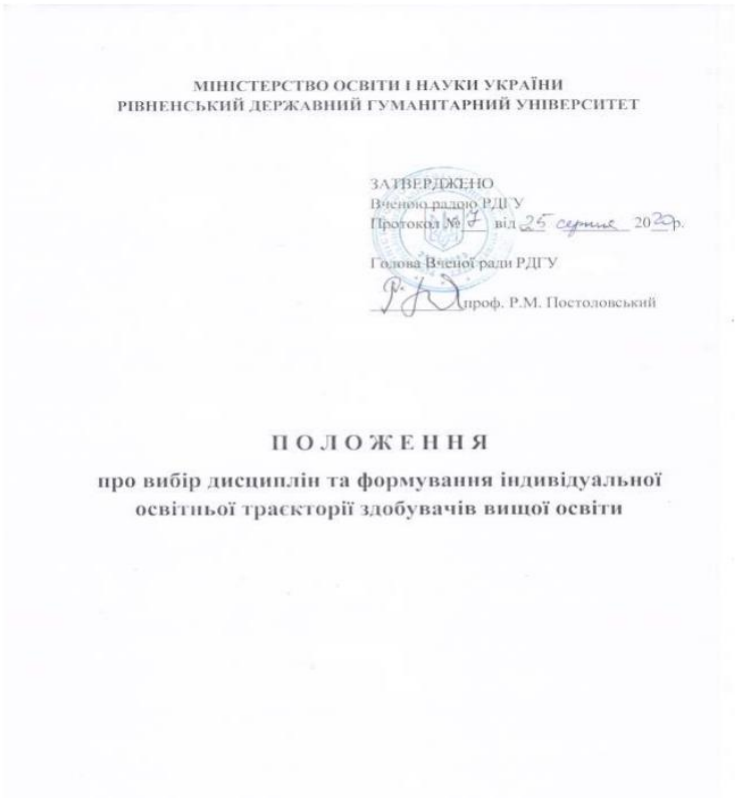
Структура роботи включає вступ, чотири розділи, загальні висновки, список використаних джерел і додатки. У першому розділі розглянуто теоретико-аналітичні засади організації вибіркових дисциплін. У другому розділі описано проєктування системи, її вимоги, архітектуру та модель даних. У третьому розділі наведено особливості реалізації та тестування прототипу. У четвертому розділі подано

організаційно-економічне й перспективне обґрунтування, а також напрямки подальшого розвитку системи.

РОЗДІЛ 1
ТЕОРЕТИКО-АНАЛІТИЧНІ ЗАСАДИ

1.1. Теоретико-аналітичні засади та сутність вибіркових освітніх компонентів

Право здобувача вищої освіти на вибір навчальних дисциплін закріплене в Законі України «Про вищу освіту», де передбачено можливість самостійного вибору дисциплін у межах освітньої програми та навчального плану. При цьому частина вибіркових компонентів має бути не меншою ніж 25 відсотків загального обсягу кредитів ЄКТС відповідного рівня вищої освіти. Нормативна вимога є важливою не лише з формальної точки зору, а й як інструмент забезпечення академічної свободи студента.



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
РІВНЕНСЬКИЙ ДЕРЖАВНИЙ ГУМАНІТАРНИЙ УНІВЕРСИТЕТ

ЗАТВЕРДЖЕНО
Вченою радою РДГУ
Протокол № 4 від 25 серпня 2020 р.

Голова Вченої ради РДГУ
проф. Р.М. Постоловський

ПОЛОЖЕННЯ
про вибір дисциплін та формування індивідуальної
освітньої траєкторії здобувачів вищої освіти

Рисунок 1.1 ПОЛОЖЕННЯ
про вибір дисциплін та формування індивідуальної освітньої траєкторії

На рівні конкретного ЗВО загальні положення закону уточнюються внутрішніми документами (рис 1.1.): положеннями про порядок вільного вибору навчальних дисциплін, рекомендаціями, правилами формування індивідуальних навчальних планів і розпорядженнями факультетів. Такі документи визначають терміни вибору, порядок формування груп, мінімальну та максимальну чисельність студентів у групі, а також процедури підтвердження вибору. Внутрішнє регулювання є необхідним, оскільки університети мають різну організаційну структуру, різні освітні програми та різну логіку формування вибіркової частини. Особливість вибіркових дисциплін полягає в тому, що вони повинні бути не формальною «додатковою опцією», а справжнім інструментом освітнього планування. Отже, система їх вибору має забезпечувати доступ до повної, структурованої та актуальної

інформації про кожну дисципліну. До такої інформації належать назва дисципліни, короткий опис, очікувані результати навчання, семестр викладання, кафедра, викладач, а в окремих випадках — попередні вимоги та тематичне спрямування.

У межах української освітньої системи важливо враховувати також принципи автономії закладу освіти та академічної доброчесності. Це означає, що ЗВО самостійно встановлює процедури вибору, але такі процедури мають бути прозорими, недискримінаційними та зрозумілими для студентів. Саме тому цифровий сервіс вибору дисциплін повинен не просто повторювати паперову логіку, а покращувати її за рахунок доступності, інтерактивності й наочності.

Вибіркові освітні компоненти є складовою освітньої програми, яка дає змогу студенту формувати індивідуальну освітню траєкторію, обираючи дисципліни з визначеного переліку. На відміну від обов’язкових компонентів, які є однаковими для всіх студентів програми, вибіркові курси дозволяють враховувати індивідуальні інтереси, кар’єрні наміри та рівень підготовки студента. У практиці ЗВО вибіркові дисципліни можуть виконувати різні функції. Частина з них поглиблює фахову підготовку в межах спеціальності, частина розширює загальнокультурний чи міждисциплінарний кругозір, а частина дає змогу розвивати додаткові навички. Для бакалаврського рівня така гнучкість є особливо важливою, оскільки студенти лише формують професійну ідентичність і можуть коригувати освітній маршрут відповідно до здобутого досвіду. Слід підкреслити, що вибірковість не означає випадковість. Університет повинен запропонувати таку структуру дисциплін, яка одночасно забезпечує свободу вибору та зберігає академічну логіку освітньої програми. Тому в сервісі для попереднього вибору варто передбачати подання дисциплін з описами, тематичними ознаками, фільтрами та зрозумілою навігацією. Це дозволяє студенту зробити не випадковий, а усвідомлений вибір.

1.2. Проблеми чинного процесу вибору вибіркових дисциплін у ЗВО та цифровізація освітніх процесів

Попри нормативну визначеність, в реальності процес вибору вибіркових дисциплін у багатьох ЗВО залишається недостатньо зручним. Найпоширенішою проблемою є розпорошеність інформації: частина відомостей міститься на сайті факультету, частина — у PDF-документах, на фізичних дошках та списках, окремі оголошення публікуються в месенджерах або на електронній пошті. Через це студенту складно отримати цілісне уявлення про доступні дисципліни, а викладачам і адміністрації — забезпечити однаковий доступ до актуальних відомостей. Ще однією проблемою є низький рівень структурованості інформації. Часто описи дисциплін подаються у вигляді коротких текстових блоків без єдиного цілісного шаблону. У результаті одна дисципліна описана детально, інша — лише назвою та кредитами, що ускладнює порівняння варіантів. Відсутність уніфікованої форми також ускладнює подальше цифрове опрацювання таких даних. Окремо слід зазначити організаційні складнощі, пов’язані з ручним або напівручним збиранням заявок. Якщо вибір відбувається через таблиці, форми або списки, адміністрація змушена додатково перевіряти правильність заповнення, усувати дублювання, контролювати кількість заявок і переносити дані до внутрішніх відомостей. Такий підхід збільшує ризик помилок і створює зайве навантаження на працівників деканату чи кафедри. Проблемою є також обмежена зрозумілість для самого студента. Після подання заявки він не завжди має змогу перевірити її статус, побачити історію свого вибору або швидко змінити рішення в межах дозволеного періоду. У цифровому середовищі такі функції є базовими, оскільки користувач очікує не лише подання даних, а й зворотного зв’язку та можливості керувати власними діями.

Наведені недоліки свідчать, що традиційний механізм вибору дисциплін потребує покращення. У цьому контексті саме вебзастосунок є найдоцільнішим рішенням, оскільки він забезпечує доступ через браузер, не потребує складної інсталяції, легко

оновлюється та може бути інтегрований із корпоративною інфраструктурою університету.

Цифровізація вищої освіти передбачає переведення частини освітніх, управлінських і комунікаційних процесів у електронну (онлайн) форму. Йдеться не лише про використання технічних засобів, а про зміну підходів до організації взаємодії між усіма учасниками освітнього процесу. Університети дедалі частіше використовують електронні кабінети, системи управління навчанням, електронний документообіг, вебпортали дисциплін і сервіси самообслуговування студентів.

Рисунок 1.2 Цифровізація процесів

Переваги такого підходу полягають у підвищенні оперативності доступу до інформації, зменшенні обсягу адміністративних операцій, підвищенні прозорості процедур і покращенні комунікації (рис 1.2.). Для студента це означає можливість швидко переглянути потрібні дані, а для адміністрації — ефективніше керувати освітніми процесами та аналітикою.

Вебтехнології в освітніх сервісах мають особливе значення, оскільки браузерний інтерфейс є універсальним, доступним і гнучким. Саме вебзастосунок може забезпечити одночасно інформаційну, комунікаційну та операційну функцію: показати дисципліни, зібрати вибір, зберегти його та надати адміністратору доступ до результатів. На відміну від статичного сайту, інтерактивний вебзастосунок може реагувати на дії користувача, оновлювати дані без перезавантаження сторінки та підтримувати різні сценарії. Важливо також, що така модернізація не зводиться до простого перенесення паперового процесу в електронну форму. Вона передбачає покращення користувацького досвіду, логіки взаємодії та якості даних. Якщо сервіс побудований правильно, він може зменшити кількість помилок, прискорити прийняття рішень і підвищити задоволеність користувачів системою.

1.3. Аналіз існуючих підходів

Існуючі підходи до організації вибору дисциплін можна умовно поділити на кілька груп. Першу групу становлять паперові або напівпаперові механізми, коли студент подає заяву, список або анкету в деканат чи на кафедру. Такі підходи є простими у запуску, але вони не забезпечують достатньої зручності, прозорості й масштабованості.

Другу групу утворюють вебсторінки з переліком дисциплін без інтерактивного вибору. Вони виконують довідкову функцію, але не розв’язують задачу обліку попереднього вибору. Третю групу становлять форми на основі Google Forms або подібних сервісів. Вони зручні на старті, однак обмежені щодо контролю доступу, структурування даних і інтеграції з іншими університетськими системами.

Порівняння підходів до організації вибору дисциплін доцільно подати у вигляді узагальненої таблиці.

Таблиця 1.1

Підхід	Переваги	Недоліки
Паперові списки	Простота запуску, не потребує розробки	Високий ризик помилок, низька прозорість, складність обліку
Статичні веб сторінки	Доступність інформації 24/7	Немає інтерактивного вибору, слабкий зворотний зв’язок
Google-форми	Швидке розгортання, мінімальні витрати	Обмежений контроль доступу, слабка інтеграція, фрагментарний облік
Спеціалізований веб застосунок	Централізація, авторизація, інтерактивність, збереження вибору	Потребує розробки, супроводу та адміністрування

У висновку ми бачимо, що вебзастосунок має найбільш збалансовані можливості. Він дозволяє реалізувати авторизацію, персональні списки вибору, централізоване зберігання даних та інструменти для адміністратора. Саме тому для проєкту РДГУ доцільно орієнтуватися не на універсальні зовнішні сервіси, а на власний прототип, адаптований до локальної освітньої структури та корпоративної пошти студентів.

1.4. Постановка задачі автоматизації процесу перегляду та попереднього вибору та висновок

Постановка задачі автоматизації полягає у створенні прототипу вебзастосунку, який забезпечує студентам РДГУ можливість авторизуватися через корпоративний акаунт, переглядати перелік вибіркових дисциплін, ознайомлюватися з їх

характеристиками, здійснювати попередній вибір і зберігати його в базі даних. Для адміністратора система повинна надавати можливість керувати переліком дисциплін і переглядати результати вибору.

Автоматизація має охоплювати такі процеси:

- автентифікацію користувача;
- доступ до каталогу дисциплін;
- перегляд детальної інформації;
- додавання дисциплін до попереднього вибору;
- збереження обраних дисциплін;
- адміністративне керування даними;
- контроль доступу та захист інформації.

Для досягнення цієї мети вебзастосунок має бути побудований за принципами модульності, масштабованості та безпеки. Клієнтська частина повинна забезпечувати зручний інтерфейс, а серверна — надійну обробку запитів, валідацію даних і збереження результатів. Оскільки мова йде про освітній сервіс, важливим є також адаптивний дизайн, простота навігації та сумісність із типовими браузерами.

Проведений аналіз показує, що вибіркові дисципліни є важливою складовою освітньої програми, а їх організація має спиратися на нормативні вимоги, принципи академічної свободи та цифрові інструменти управління освітнім процесом. Чинні підходи до вибору дисциплін у багатьох ЗВО не забезпечують належного рівня зручності, прозорості та інтерактивності, тому потребують модернізації. Вебзастосунок як форма реалізації такого сервісу є доцільним рішенням, оскільки поєднує доступність, масштабованість і можливість централізованого зберігання даних.

Що перевірити перед фінальною версією

- Уточнити точну назву кафедри, факультету та офіційне формулювання спеціальності у РДГУ.

- Перевірити, чи треба у вступі додатково подати «методи дослідження» окремим підпунктом за вимогами кафедри.
- Уточнити, чи має у роботі бути згадка про конкретний внутрішній документ РДГУ щодо вибіркових дисциплін.
- За потреби узгодити термінологію: скрізь лишити «вебзастосунок» або замінити на «вебзастосунок».
- Перевірити, чи потрібен окремий підрозділ про аналоги з прикладами українських університетів.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ВЕБЗАСТОСУНКУ ДЛЯ ПОПЕРЕДНЬОГО ВИБОРУ ВИБІРКОВИХ ДИСЦИПЛІН

2.1. Формалізація функціональних та нефункціональних вимог

Функціональні вимоги визначають, які саме дії має виконувати система. Для прототипу застосунку перегляду та попереднього вибору вибіркових дисциплін ці вимоги формуються з урахуванням потреб студентів РДГУ, а також адміністративного супроводу освітнього процесу.

До основних функціональних вимог належать:

- реєстрація та вхід користувача через корпоративний акаунт;
- обмеження доступу до системи лише для студентів РДГУ;
- перегляд переліку вибіркових дисциплін;
- перегляд розширеної інформації про конкретну дисципліну;
- фільтрація та пошук дисциплін за основними ознаками;
- додавання дисциплін до попереднього вибору;
- збереження обраних дисциплін у базі даних;
- перегляд власного списку обраних дисциплін;
- видалення дисципліни зі списку попереднього вибору до завершення періоду вибору;
- надання адміністратору можливості створювати, редагувати та видаляти дисципліни;
- перегляд адміністративної інформації про вибір студентів.

Функціональність системи доцільно подати у вигляді таблиці.

Таблиця 2.1

Функція	**Призначення**	**Роль користувача**
Авторизація через корпоративний акаунт	Підтвердження особи студента	Студент
Перегляд каталогу дисциплін	Ознайомлення з доступними дисциплінами	Студент
Перегляд деталей дисципліни	Отримання повного опису курсу	Студент
Попередній вибір дисциплін	Формування персонального списку	Студент
Збереження вибору	Фіксація результату в базі даних	Студент
Керування дисциплінами	Адміністрування каталогу	Адміністратор
Перегляд статистики вибору	Контроль результатів та аналіз	Адміністратор

Система має підтримувати логіку, за якої студент спочатку проходить авторизацію, далі отримує доступ до каталогу дисциплін, після чого формує власний вибір. Це означає, що функції доступу, перегляду та збереження повинні бути тісно пов’язані між собою, а інтерфейс — не перевантажувати користувача зайвими діями.

Нефункціональні вимоги визначають якісні характеристики системи. Для освітнього вебзастосунку особливо важливими є швидкодія, надійність, безпека, масштабованість і зручність використання.

До основних нефункціональних вимог належать:

- швидке завантаження сторінок каталогу дисциплін;
- коректна робота системи в популярних браузерах;
- адаптивний інтерфейс для різних пристроїв;
- захист персональних даних користувачів;
- підтримка безпечної авторизації;
- можливість подальшого розширення функціоналу;
- стабільна робота при одночасному доступі багатьох користувачів;
- логічна структура інтерфейсу та мінімальна кількість зайвих дій.

Особливу увагу слід приділити безпеці. Оскільки система працює з персональними даними студентів, навіть прототип має передбачати захищений канал передачі даних, коректне зберігання токенів доступу та обмеження доступу до адміністративних функцій. Крім того, інтерфейс повинен бути достатньо простим, аби студент міг без додаткового навчання виконати основні дії.

Порівняльно нефункціональні характеристики можна подати у наступному вигляді.

Таблиця 2.2

Характеристика	**Вимога**
Продуктивність	Швидке завантаження каталогу та детальної сторінки
Безпека	Захист авторизації, даних і сесій

Надійність	Стабільна робота без втрати обраних дисциплін
Масштабованість	Можливість обслуговувати зростання кількості користувачів
Зручність	Мінімальна кількість кроків для виконання основних операцій
Адаптивність	Коректна робота на десктопах, планшетах і смартфонах

2.2. Ролі користувачів і сценарії взаємодії

У проєкті передбачено дві основні ролі користувачів: студент і адміністратор. Студент є основним користувачем системи. Він отримує доступ до каталогу вибіркових дисциплін, переглядає їх опис, додає дисципліни до попереднього вибору та зберігає результат. У межах прототипу студент взаємодіє з системою через особистий кабінет, який доступний після авторизації. Адміністратор відповідає за наповнення та актуалізацію каталогу дисциплін. Його завдання полягають у створенні записів про дисципліни, редагуванні їх характеристик, а також у перегляді загальних результатів вибору студентів. Адміністратор не виконує функції вибору дисциплін, але керує інформаційною основою процесу.

Типові сценарії взаємодії можна описати як user stories:

- як студент, я хочу увійти через корпоративний акаунт, щоб система ідентифікувала мене як здобувача РДГУ;
- як студент, я хочу переглянути перелік вибіркових дисциплін, щоб обрати найбільш доречний варіант;

- як студент, я хочу додати дисципліну до свого списку, щоб не втратити її серед інших варіантів;
- як студент, я хочу бачити свій попередній вибір, щоб мати змогу його уточнити;
- як адміністратор, я хочу додавати нові дисципліни, щоб каталог завжди залишався актуальним;
- як адміністратор, я хочу змінювати або видаляти дисципліни, щоб підтримувати відповідність реальному навчальному плану.

Такі сценарії задають логіку системи на рівні користувацького досвіду. Вони допомагають не лише сформувати інтерфейс, а й правильно спроєктувати маршрути, структуру бази даних і права доступу.

2.3. Архітектура системи та стек

Для реалізації прототипу обрано трьохшарову архітектуру, що є доцільною для вебзастосунків освітнього типу. Така архітектура розділяє систему на рівень представлення, рівень бізнес-логіки та рівень доступу до даних.

Рівень представлення відповідає за інтерфейс користувача. Саме тут студент бачить каталог дисциплін, сторінки авторизації, деталі курсу та список обраного. Для цього рівня використано React, оскільки цей фреймворк добре підходить для створення компонентних інтерфейсів, повторного використання частин коду та керування станом сторінок.

Рівень бізнес-логіки обробляє запити від клієнта, перевіряє права доступу, виконує валідацію даних і формує відповіді. Для реалізації цього рівня використано Node.js і Express, оскільки така зв’язка дозволяє швидко будувати REST API, організовувати маршрути й впроваджувати проміжне програмне забезпечення для перевірок і авторизації.

Рівень доступу до даних відповідає за збереження інформації у базі даних. Для цього обрано MongoDB, а для роботи з нею — Mongoose. Такий підхід зручний на

етапі прототипування, оскільки структура документів є гнучкою і дозволяє поступово розширювати модель без складних міграцій. Обґрунтування вибору технологій доцільно подати у вигляді порівняння.

Таблиця 2.3

Технологія	**Переваги для проєкту**	**Причина вибору**
React	Компонентність, повторне використання, зручний UX	Підходить для динамічного інтерфейсу
Vue	Простий старт, добра реактивність	Альтернатива, але менша відповідність наявному стеку
Angular	Потужна структура, вбудовані інструменти	Надмірно складний для прототипу
Node.js + Express	Швидка розробка API, JavaScript на сервері	Єдина мова на фронтенді й бекенді
MongoDB	Гнучка структура документів	Зручна для швидкого розвитку моделі
Реляційна БД	Сувора схема, SQL-запити	Менш гнучка для прототипу з мінливою структурою

Вибір саме цієї архітектури є виправданим через потребу в швидкому створенні прототипу, легкому супроводі та можливості подальшого розширення. Оскільки

система орієнтована на реальний освітній процес, важливо, щоб вона могла згодом інтегруватися з іншими університетськими сервісами.

Системи РДГУ

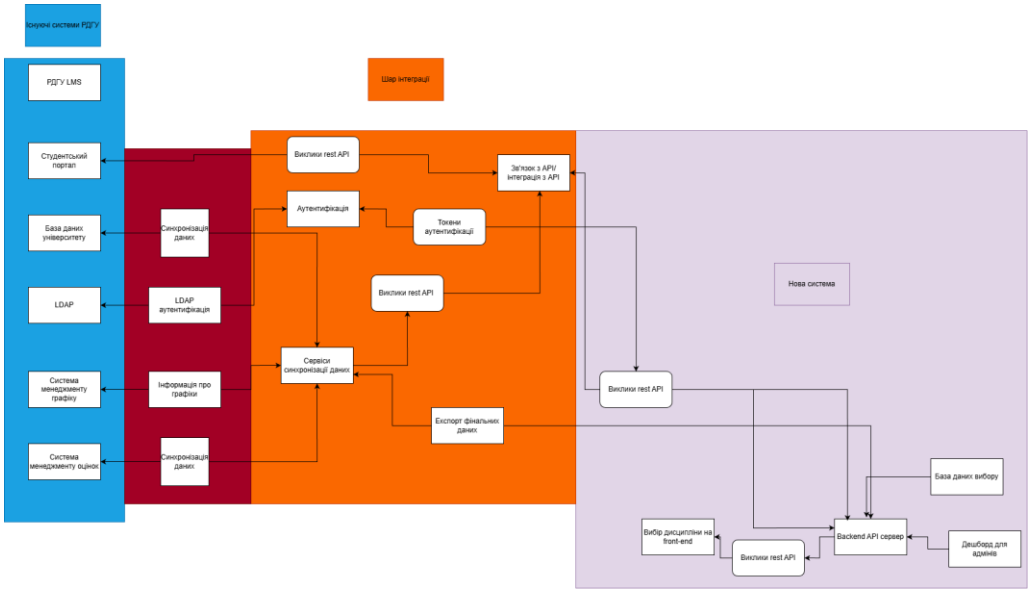


Рисунок 2.1 Модель прототипу системи

2.4. Проєктування моделі даних та API

Модель даних для вебзастосунку повинна відображати предметну область достатньо просто, але без втрати суттєвих зв’язків. Для цієї системи доцільно виділити три основні сутності: користувач, дисципліна та запис про вибір.

Користувач містить дані про студентів і адміністраторів: ім’я, електронну пошту, роль, статус авторизації, ідентифікатор Google-акаунта за потреби.

Дисципліна містить назву, опис, викладача, кредити, семестр, кафедру, форму контролю та інші характеристики.

Запис про вибір фіксує, які дисципліни обрав конкретний студент.

Приклад логіки зв’язків:

- один користувач може мати багато обраних дисциплін;
- одна дисципліна може бути обрана багатьма студентами;
- один запис про вибір належить конкретному студенту.

Модель MongoDB може виглядати наступним чином

```
User {
  _id,
  name,
  email,
  role,
  googleId,
  createdAt
}
```

```
Discipline {
  _id,
  title,
  description,
  credits,
  semester,
  teacher,
  department,
  formOfControl,
  isActive
}
```

```
Selection {
  _id,
  userId,
  disciplineIds: [],
  status,
  updatedAt
}
```

Таке проєктування зберігає гнучкість бази даних і дає змогу легко модифікувати структуру в майбутньому. Наприклад, можна додати поля для максимального набору студентів, пріоритету дисципліни, часових обмежень або рекомендацій.

Серверна частина системи повинна надавати набір зрозумілих endpoint-ів для взаємодії з клієнтом. Оскільки система будується за REST-підходом, кожен маршрут відповідає за певний ресурс або дію. Основні маршрути наведено в таблиці нижче.

Таблиця 2.4

Метод	**Маршрут**	**Призначення**
GET	/api/disciplines	Отримання списку дисциплін
GET	/api/disciplines/:id	Отримання деталей конкретної дисципліни
POST	/api/auth/google	Авторизація через Google
GET	/api/auth/me	Отримання даних поточного користувача
POST	/api/selections	Створення або оновлення вибору
GET	/api/selections/me	Отримання списку обраних дисциплін студента
DELETE	/api/selections/:id	Видалення дисципліни з вибору
POST	/api/admin/disciplines	Створення дисципліни
PUT	/api/admin/disciplines/:id	Оновлення дисципліни
DELETE	/api/admin/disciplines/:id	Видалення дисципліни

API має бути побудоване так, щоб клієнт отримував лише ті дані, які йому потрібні. Це зменшує навантаження на мережу, спрощує структуру відповіді та покращує користувацький досвід. Для доступу до закритих маршрутів слід використовувати токенну перевірку, а для адміністративних дій — додаткову перевірку ролі.

2.5. Безпека системи

Оскільки система працює з персональними даними студентів, питання безпеки є одним із пріоритетних. Для прототипу передбачено декілька рівнів захисту (рис 2.2).

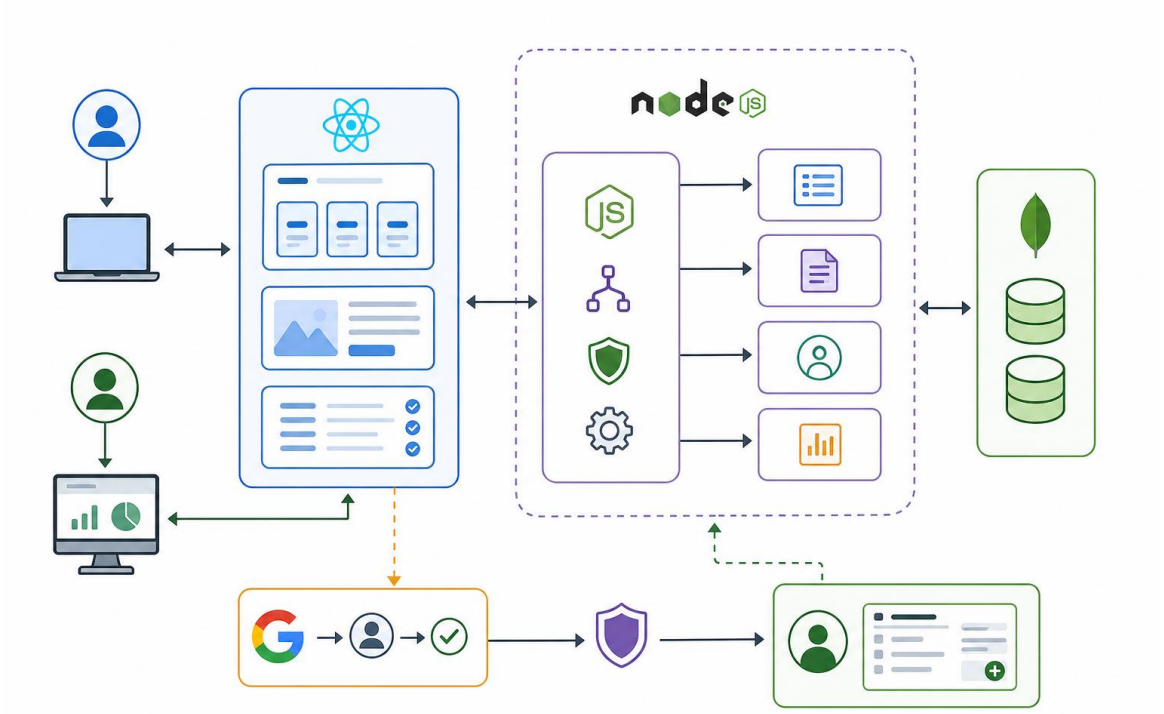


Рисунок 2.2 Схема забезпечення безпеки вебзастосунку та взаємодії його компонентів

По-перше, авторизацію доцільно реалізувати через Google OAuth 2.0. Це дає змогу знизити ризик створення слабких паролів і спрощує вхід користувача. Крім того, авторизація через корпоративну пошту РДГУ дає змогу обмежити доступ лише

студентам або працівникам університету. Такий підхід особливо важливий для освітнього сервісу, де потрібно контролювати, хто саме має право переглядати та зберігати вибір.

По-друге, після успішного входу система може використовувати JWT для підтримки сесії користувача. Токен дозволяє серверу не зберігати стан сесії для кожного користувача окремо, що спрощує масштабування й робить архітектуру більш гнучкою.

По-третє, слід налаштувати CORS таким чином, щоб доступ до API мали лише дозволені домени фронтенду. Це зменшує ризик небажаних міждоменних запитів і підвищує загальний рівень захищеності.

По-четверте, конфіденційні дані, зокрема ключі доступу, мають зберігатися у змінних середовища, а не прямо в коді. Для цього використовується файл конфігурації середовища, який не повинен потрапляти до відкритого репозиторію. Також доцільно передбачити базовий захист від типових вразливостей: перевірку вхідних даних, обмеження ролей, обробку помилок без розкриття внутрішньої структури сервера та коректну роботу з HTTP-заголовками.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОТОТИПУ ВЕБ ЗАСТОСУНКУ

3.1. Реалізація frontend та backend-частин

Frontend-частина прототипу відповідає за взаємодію користувача із системою. Вона має забезпечити зрозумілий доступ до каталогу дисциплін, авторизації, сторінки детальної інформації та особистого списку попереднього вибору. Для цього використано React як компонентну бібліотеку, що дає змогу будувати інтерфейс із повторно використовуваних блоків і підтримувати його в актуальному стані без повного перезавантаження сторінки.

Інтерфейс системи доцільно поділити на кілька основних сторінок:

- головна сторінка з коротким описом сервісу;
- сторінка авторизації;
- сторінка каталогу вибіркових дисциплін;
- сторінка детальної інформації про дисципліну;
- сторінка обраних дисциплін;
- адміністративна сторінка для керування записами.

Компонентний підхід у React дозволяє винести повторювані елементи, такі як шапка сторінки, картка дисципліни, кнопка вибору та форма авторизації, в окремі модулі. Це спрощує підтримку коду та зменшує ймовірність помилок під час розширення системи. Наприклад, картка дисципліни може бути однаковою для каталогу та для адміністративного перегляду, а відрізнятися лише набором доступних дій. Для маршрутизації між сторінками використано react-router-dom, що є стандартним інструментом для SPA-застосунків. Завдяки цьому користувач переходить між сторінками без повного перезавантаження браузера, а стан інтерфейсу

зберігається плавно. Такий підхід особливо зручний для освітнього сервісу, де важлива швидка навігація між великим числом відомостей.

Окрему увагу приділено адаптивності інтерфейсу. Оскільки студенти можуть заходити до системи не лише з комп'ютера, а й зі смартфона чи планшета, сторінки мають коректно відображатися на різних екранах. Для стилізації використано CSS, що забезпечує достатню гнучкість для адаптації макетів без надмірного ускладнення структури проєкту. Фронтенд також містить логіку відображення авторизованого стану користувача. Після входу студент бачить доступ до особистого кабінету, а адміністратор — до службових функцій. Це дозволяє на рівні інтерфейсу розмежувати користувацькі сценарії та не перевантажувати студента зайвими елементами.

Backend-частина системи виконує функції обробки запитів, перевірки прав доступу, взаємодії з базою даних і формування відповідей для клієнта (рис 3.1). Для цього використано Node.js та Express, оскільки вони дають змогу швидко реалізувати REST API, організувати маршрути, підключати middleware і зручно структурувати код.

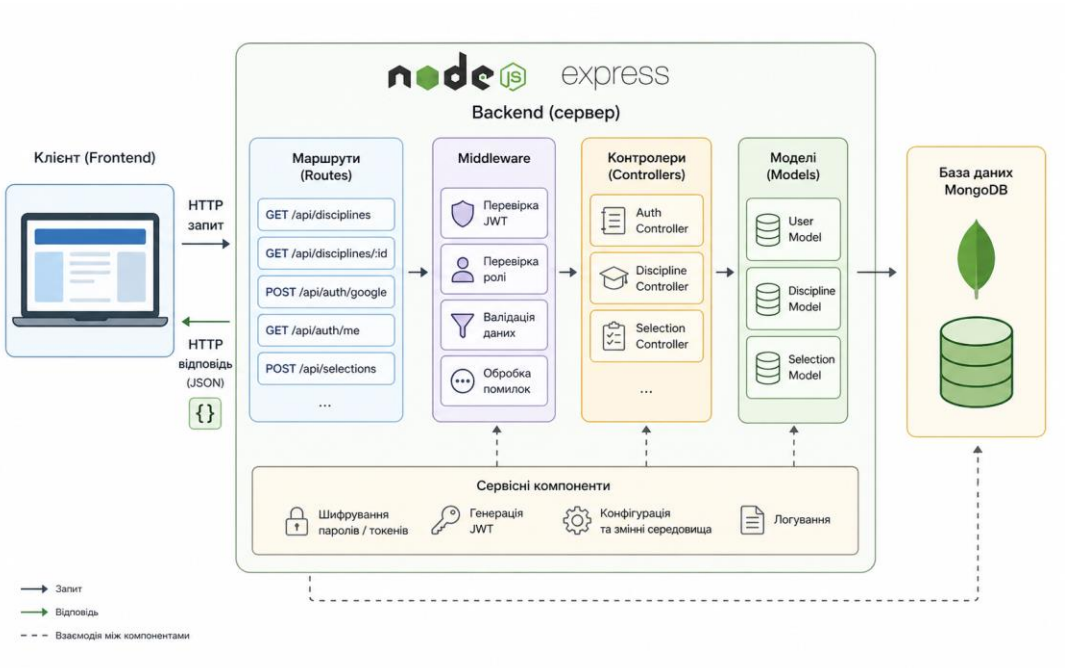


Рисунок 3.1 Архітектура backend-частини веб застосунку та обробка HTTP-запитів

На сервері реалізовано кілька основних груп логіки:

- маршрути авторизації;
- маршрути для дисциплін;
- маршрути для збереження вибору;
- адміністративні маршрути;
- middleware для перевірки токена;
- middleware для перевірки ролі користувача;
- middleware для обробки помилок.

У контролерах зосереджено основну бізнес-логіку: отримання списку дисциплін, пошук конкретного запису, створення нового елемента, оновлення обраних дисциплін і повернення відповідей у форматі JSON. Такий підхід відповідає принципу розділення відповідальностей, коли кожен модуль виконує свою чітку функцію.

Для роботи з базою даних використано Mongoose, який спрощує створення схем, валідацію даних і виконання запитів до MongoDB. Це зручно для прототипу, оскільки дає можливість задавати структуру документів без складних реляційних зв’язків, але водночас зберігати цілісність даних. У моделі даних визначено основні поля для користувача, дисципліни та результату вибору. Наявність middleware є важливою з точки зору безпеки та підтримуваності. Наприклад, окремий модуль може перевіряти, чи передано дійсний JWT-токен, інший — чи має користувач права адміністратора. Це дає змогу не дублювати однакову логіку в кожному маршруті та зменшує обсяг коду.

3.2. Ключові сценарії роботи та збереження даних

Прототип реалізує кілька основних сценаріїв користувацької взаємодії. Нижче наведено найважливіші з них.

Вхід студента

Користувач відкриває сторінку авторизації та входить через корпоративний акаунт. Після успішного входу система перевіряє домен електронної пошти та ідентифікує користувача як студента РДГУ. У разі успішної перевірки формується токен доступу, який надалі використовується для авторизованих запитів. Такий сценарій дає змогу уникнути окремої реєстрації та зменшує ризик утворення зайвих облікових записів. Оскільки користувач входить через уже наявний корпоративний акаунт, система краще контролює доступ до персональних даних.

Перегляд дисциплін

Після входу студент переходить до каталогу дисциплін. На цій сторінці відображається базова інформація: назва курсу, кредитність, короткий опис і, за потреби, викладач або семестр. Користувач може відкрити детальну картку дисципліни для ознайомлення з повним описом. Для такого сценарію важливо, щоб інтерфейс був лаконічним. Якщо користувач бачить надто багато інформації одночасно, йому складніше швидко зорієнтуватися. Тому каталог повинен бути організований так, щоб основні дані були видимі одразу, а розширений опис відкривався окремо.

Додавання дисципліни

Після перегляду дисципліни студент може додати її до попереднього вибору. Система надсилає запит на сервер, де перевіряється коректність даних і прив’язка дисципліни до конкретного користувача. Після цього інформація зберігається в базі даних і відображається у списку обраних дисциплін. Цей сценарій є ключовим для всієї системи, оскільки саме він перетворює інформаційний каталог на робочий інструмент вибору. Якщо збереження виконано коректно, студент може повернутися до системи пізніше й побачити свій вибір у незмінному стані.

Перегляд обраного

У персональному кабінеті студент бачить перелік дисциплін, які вже додав до попереднього вибору. У разі потреби він може видалити окрему дисципліну до завершення дозволеного періоду вибору. Така можливість важлива, оскільки студенти

нерідко змінюють рішення після додаткового ознайомлення з описом курсу або консультації з викладачем. Персональний список має бути зручним і зрозумілим. Найкраще, якщо він відображатиме не лише назву дисципліни, а й основні параметри, щоб користувач міг швидко оцінити свій вибір.

Збереження даних реалізовано через MongoDB, а доступ до неї здійснюється через Mongoose. У процесі створення прототипу було визначено, що дані користувачів і дисциплін повинні зберігатися окремо, а зв’язок між ними реалізовуватися через ідентифікатор студента та список обраних дисциплін.

Такий підхід зручний, тому що:

- дає змогу окремо оновлювати інформацію про дисципліни;
- не дублює великі масиви даних;
- забезпечує швидкий доступ до персонального вибору;
- дає змогу легко додавати нові поля в майбутньому.

У прототипі передбачено базові операції створення, читання, оновлення та видалення. Для студента доступні лише дії, пов’язані з вибором, а для адміністратора — розширене керування каталогом. Це підтримує принцип обмеження доступу за ролями.

3.3. Тестування прототипу та Результати апробації

Тестування є обов’язковою частиною реалізації, оскільки навіть функціонально правильний код може містити логічні помилки або недоліки у взаємодії компонентів. Для прототипу доцільно використовувати кілька рівнів перевірки: функціональне, інтеграційне та базове тестування безпеки.

Функціональне тестування

Функціональне тестування спрямоване на перевірку того, чи відповідає реалізована система вимогам. Для цього перевіряються такі сценарії:

1. успішний вхід студента через корпоративний акаунт;

2. доступ до каталогу дисциплін після авторизації;
3. відкриття детальної сторінки дисципліни;
4. додавання дисципліни до списку обраного;
5. збереження вибору після повторного входу;
6. обмеження доступу до адміністративних функцій.

Кожен сценарій повинен завершуватися очікуваним результатом. Якщо система поводиться інакше, це означає, що необхідно коригувати логіку маршруту, стану або валідації.

Інтеграційне тестування

Інтеграційне тестування перевіряє взаємодію між frontend, backend і базою даних. У цьому випадку важливо впевнитися, що:

- фронтенд коректно надсилає запити;
- сервер повертає очікувані дані;
- база даних зберігає записи без втрати структури;
- токен авторизації передається й перевіряється правильно.

Особливо важливо протестувати сценарій, коли студент додає дисципліну до вибору, потім перезавантажує сторінку або входить повторно. У такому випадку система має відтворити попередній стан без втрати даних.

Базова перевірка безпеки

Базова перевірка безпеки включає:

- валідацію вхідних даних;
- перевірку ролей користувачів;
- контроль доступу до закритих маршрутів;
- коректну роботу CORS;
- недопущення розкриття службової інформації у відповідях сервера.

Також перевіряється, чи не доступні адміністративні маршрути студенту без відповідних прав. Для освітнього сервісу це особливо важливо, оскільки помилка в правах доступу може призвести до зміни або видалення важливих даних.

Апробація прототипу показала, що обрана архітектура є придатною для реалізації системи попереднього вибору вибіркових дисциплін. Система дозволяє виконати основні користувацькі сценарії без надлишкових дій, а інтерфейс є достатньо простим для студентів, які вперше працюють із сервісом.

До позитивних результатів апробації належать:

- швидка навігація між сторінками;
- зрозуміла структура каталогу дисциплін;
- можливість авторизації через корпоративний акаунт;
- збереження вибору в базі даних;
- поділ доступу між студентом і адміністратором.

Разом із тим, прототип має й обмеження. Він не є повноцінною промисловою системою та потребує подальшого розвитку. Зокрема, можна розширити аналітичний модуль, додати механізми повідомлень, удосконалити пошук і фільтрацію, а також інтегрувати систему з іншими університетськими сервісами.

3.4. Рекомендації щодо впровадження

Для можливого впровадження у ЗВО доцільно врахувати такі рекомендації:

- забезпечити інтеграцію з корпоративною поштою університету;
- налаштувати ролі користувачів відповідно до внутрішньої структури;
- передбачити резервне копіювання даних;
- створити адміністративний інтерфейс для наповнення каталогу;
- провести тестування на реальних сценаріях студентів і працівників;
- передбачити можливість подальшого масштабування системи.

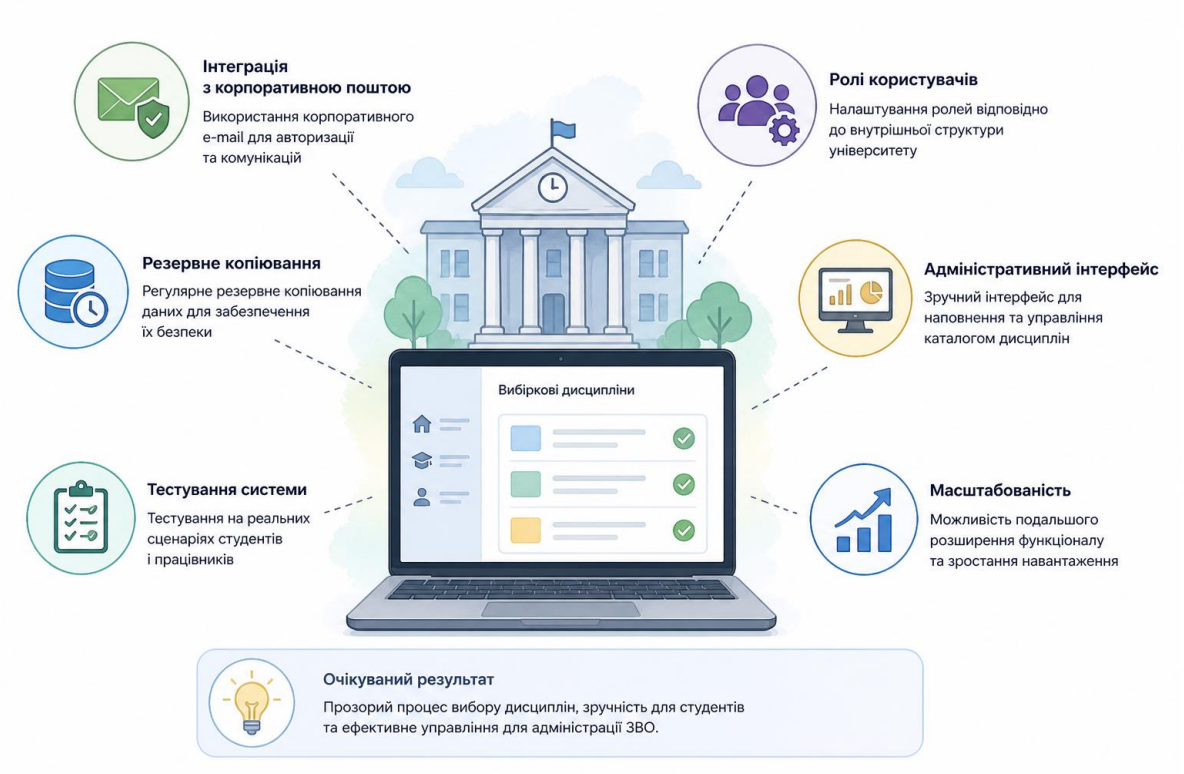


Рисунок 3.2 Схема впровадження веб застосунку в РДГУ

У випадку РДГУ система може стати корисною як для студентів, так і для адміністрації, оскільки спрощує облік вибору та підвищує прозорість процесу. Однак впровадження має відбуватися поступово, з урахуванням внутрішніх нормативних процедур і технічної інфраструктури університету (рис. 3.2).

РОЗДІЛ 4
ОРГАНІЗАЦІЙНО-ЕКОНОМІЧНЕ ТА ПЕРСПЕКТИВНЕ ОБҐРУНТУВАННЯ

4.1. Організаційні аспекти впровадження

Запропонований вебзастосунок орієнтований на використання в освітньому процесі РДГУ як інструмент для перегляду та попереднього вибору вибіркових дисциплін. Його впровадження не вимагає повної перебудови всієї інформаційної інфраструктури університету, але передбачає узгодження з існуючими правилами обліку студентів, корпоративною поштою та внутрішніми регламентами факультетів.

Організаційно впровадження доцільно здійснювати поетапно. На першому етапі система може використовуватися в режимі апробації на окремій освітній програмі або факультеті. Це дає змогу перевірити зручність інтерфейсу, коректність авторизації, роботу з базою даних і зрозумілість сценарію вибору. На другому етапі можливе розширення на інші освітні програми, якщо результати апробації будуть позитивними.

Для успішного впровадження потрібні такі організаційні умови:

1. наявність актуального переліку вибіркових дисциплін;
2. відповідальний адміністратор або уповноважена особа;
3. доступ до корпоративних облікових записів студентів;
4. визначений період для попереднього вибору;
5. погоджений порядок оновлення даних про дисципліни;
6. технічна підтримка для супроводу системи.

Перевага вебформату полягає в тому, що університет може використовувати його без складної інсталяції на пристроях студентів. Доступ здійснюється через браузер, а це спрощує поширення сервісу та зменшує витрати на навчання користувачів. Водночас адміністрації потрібно забезпечити регулярне оновлення каталогу, оскільки неактуальна інформація нівелює переваги цифрового рішення.

4.2. Оцінка ефективності та ризики впровадження

Ефективність прототипу доцільно оцінювати не лише технічно, а й організаційно. У цифровізації вищої освіти цінність системи визначається тим, наскільки вона скорочує час виконання операцій, зменшує кількість помилок і робить процес прозорішим для всіх учасників.

Порівняно з ручним або напів ручним способом вибору дисциплін, запропонований веб застосунок може дати такі якісні переваги:

- студент швидше знаходить потрібну дисципліну;
- інформація про курси подається в уніфікованому форматі;
- вибір зберігається в системі без додаткового дублювання;
- адміністратор отримує централізований доступ до результатів;
- скорочується кількість уточнень і повторних звернень.

Для орієнтовної оцінки ефективності можна використати просту модель показників:

- **час вибору дисциплін** — зменшення часу, необхідного студенту для перегляду і фіксації вибору;
- **кількість помилок** — зменшення кількості дублювань, неправильних заявок і ручних виправлень;
- **прозорість процесу** — можливість перегляду обраних дисциплін у будь-який момент;
- **навантаження на адміністрацію** — скорочення ручної роботи з обліком заявок.

Наприклад, якщо раніше студент змушений був шукати відомості у кількох джерелах, а потім окремо подавати заявку, то у вебзастосунку ці дії об’єднуються в один сценарій. Це не лише зручніше, а й зменшує ризик втрати даних. На рівні

університету це може проявитися у меншій кількості звернень до деканату щодо уточнення списків або перевірки статусу вибору.

Будь-яка інформаційна система, навіть прототип, має низку ризиків. Для освітнього вебзастосунку основні ризики пов’язані з організацією процесу, технічним супроводом і прийняттям користувачами.

Найбільш ймовірні ризики такі:

- несвоєчасне оновлення переліку дисциплін;
- відсутність відповідального за супровід системи;
- низька цифрова дисципліна користувачів;
- технічні помилки при авторизації;
- некоректне налаштування доступу до корпоративних акаунтів;
- відсутність інтеграції з іншими сервісами університету;
- недостатній рівень тестування перед запуском.

Для мінімізації цих ризиків доцільно:

- визначити відповідальних осіб;
- запровадити регламент оновлення дисциплін;
- протестувати систему на реальних сценаріях;
- передбачити резервне копіювання бази даних;
- обмежити доступ до адміністративних функцій;
- проводити коротке інструктажне ознайомлення користувачів.

Окремим ризиком є сприйняття системи як «ще одного сервісу», який не інтегрований з іншими процесами університету. Щоб цього уникнути, важливо підкреслити її роль як допоміжного інструмента освітнього управління, а не як автономного й ізольованого ресурсу. Саме інтеграція з академічною екосистемою робить сервіс по-справжньому корисним.

4.3. Дорожня карта розвитку

Прототип може бути поступово розширений до повноцінного цифрового сервісу. Найбільш логічним є поетапний розвиток, у межах якого кожне нове оновлення додає практичну цінність.

Можливі напрями розвитку:

- інтеграція з внутрішніми системами РДГУ;
- додавання аналітичного модуля для адміністрації;
- створення рекомендаційного механізму;
- впровадження мобільної версії або PWA;
- підтримка багатомовності;
- повідомлення про статус вибору;
- фільтрація дисциплін за кафедрою, семестром, викладачем, кредитністю;
- збереження історії виборів.

План подальшого розвитку доцільно подати у вигляді таблиці.

Таблиця 4.1

Етап	Функція	Очікуваний ефект
1	Апробація прототипу	Перевірка базового сценарію
2	Інтеграція з системами РДГУ	Автоматизація обміну даними
3	Аналітика вибору	Підтримка рішень адміністрації
4	Рекомендації для студента	Покращення користувацького досвіду
5	Мобільна адаптація / PWA	Зручність використання на різних пристроях
6	i18n	Підтримка іноземних студентів

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було досягнуто поставленої мети — розроблено прототип вебзастосунку для перегляду та попереднього вибору вибіркових дисциплін студентами.

Проаналізовано нормативно-правові засади реалізації права студентів на вибір дисциплін у закладах вищої освіти України. Встановлено, що вибіркові освітні компоненти є важливою складовою індивідуальної освітньої траєкторії студента та мають забезпечуватися відповідно до вимог Закону України «Про вищу освіту» й принципів академічної свободи.

Сформовано функціональні та нефункціональні вимоги до вебзастосунку, які охоплюють авторизацію користувачів, перегляд каталогу дисциплін, попередній вибір і збереження результатів, а також вимоги до безпеки, адаптивності, продуктивності та зручності використання системи.

Обґрунтовано архітектуру та технологічний стек вебзастосунку. Для реалізації системи обрано трьохшарову архітектуру з використанням React для frontend-частини, Node.js та Express для backend-частини, а також MongoDB як системи керування базою даних. Визначено, що обрані технології забезпечують гнучкість, масштабованість і зручність подальшого розвитку системи.

Спроєктовано структуру даних і основні API-інтерфейси вебзастосунку. Розроблено моделі користувачів, дисциплін і результатів вибору, а також визначено REST API-маршрути для авторизації, роботи з каталогом дисциплін, збереження вибору та адміністрування системи.

Реалізовано та протестовано прототип системи, який забезпечує перегляд вибіркових дисциплін, авторизацію через корпоративний акаунт, попередній вибір дисциплін і збереження результатів у базі даних. У процесі тестування перевірено коректність роботи основних користувацьких сценаріїв, взаємодію між frontend- і backend-частинами та базові механізми захисту даних.

Оцінено можливості практичного застосування розробленого вебзастосунку в освітньому процесі РДГУ. Визначено, що система може бути використана як основа для подальшого впровадження цифрового сервісу вибору дисциплін, що сприятиме підвищенню зручності для студентів, прозорості процесу вибору та зменшенню адміністративного навантаження.

Таким чином, кваліфікаційна робота має практичне значення та демонструє можливість використання сучасних вебтехнологій для цифровізації освітніх процесів у закладах вищої освіти.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Закон України «Про вищу освіту» від 01.07.2014 № 1556-VII. URL: https://zakon.rada.gov.ua/laws/show/1556-18#Text (дата звернення: 02.05.2026).
2. Закон України «Про захист персональних даних» від 01.06.2010 № 2297-VI. URL: https://zakon.rada.gov.ua/laws/show/2297-17#Text (дата звернення: 02.05.2026).
3. Закон України «Про освіту» від 05.09.2017 № 2145-VIII. URL: https://zakon.rada.gov.ua/laws/show/2145-19#Text (дата звернення: 02.05.2026).
4. React. URL: https://react.dev (дата звернення: 02.05.2026).
5. React: Quick Start. URL: https://react.dev/learn (дата звернення: 02.05.2026).
6. React: Reference Overview. URL: https://react.dev/reference/react (дата звернення: 02.05.2026).
7. Express - Node.js web application framework. URL: https://expressjs.com (дата звернення: 02.05.2026).
8. MDN Web Docs. Express/Node.js introduction. URL: https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Server-side/Express_Nodejs (дата звернення: 02.05.2026).
9. MongoDB Documentation. Database Scaling. URL: https://www.mongodb.com/resources/basics/scaling (дата звернення: 02.05.2026).
10. MongoDB Atlas. Scalability With MongoDB Atlas. URL: https://www.mongodb.com/resources/products/capabilities/scalability-with-mongodb-atlas (дата звернення: 02.05.2026).
11. Mongoose Documentation. URL: https://mongoosejs.com/docs/ (дата звернення: 02.05.2026).
12. Mongoose. URL: https://mongoosejs.com (дата звернення: 02.05.2026).
13. Tutorial: Mongoose Get Started. URL:https://www.mongodb.com/docs/drivers/node/current/integrations/mongoose/mongoose-get-started/ (дата звернення: 02.05.2026).
14. Mongoose: Documents. URL: https://mongoosejs.com/docs/documents.html (дата звернення: 02.05.2026).

15. Google. Using OAuth 2.0 to Access Google APIs. URL: https://developers.google.com/identity/protocols/oauth2 (дата звернення: 02.05.2026).
16. Google. Using OAuth 2.0 for Web Server Applications. URL: https://developers.google.com/identity/protocols/oauth2/web-server (дата звернення: 02.05.2026).
17. Passport. URL: https://www.passportjs.org (дата звернення: 02.05.2026).
18. passport-google-oauth2. URL: https://www.passportjs.org/packages/passport-google-oauth2/ (дата звернення: 02.05.2026).
19. JWT.io. Introduction. URL: https://jwt.io/introduction (дата звернення: 02.05.2026).
20. Auth0. JSON Web Tokens. URL: https://auth0.com/docs/secure/tokens/json-web-tokens (дата звернення: 02.05.2026).
21. MDN Web Docs. Cross-Origin Resource Sharing (CORS). URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/CORS (дата звернення: 02.05.2026).
22. MDN Web Docs. CORS configuration. URL: https://developer.mozilla.org/en-US/docs/Web/Security/Practical_implementation_guides/CORS (дата звернення: 02.05.2026).
23. Axios. URL: https://axios-http.com (дата звернення: 02.05.2026).
24. cors. URL: https://www.npmjs.com/package/cors (дата звернення: 02.05.2026).
25. dotenv. URL: https://www.npmjs.com/package/dotenv (дата звернення: 02.05.2026).
26. Node.js Documentation. URL: https://nodejs.org/en/docs (дата звернення: 02.05.2026).
27. React Router. URL: https://reactrouter.com (дата звернення: 02.05.2026).
28. JavaScript Info. URL: https://uk.javascript.info/ (дата звернення: 02.05.2026).
29. React Component Based Architecture. URL: https://www.geeksforgeeks.org/reactjs/react-component-based-architecture/ (дата звернення: 02.05.2026).
30. Optimizing Performance. URL: https://legacy.reactjs.org/docs/optimizing-performance.html (дата звернення: 02.05.2026).

31. Публікація про цифровізацію університетської освіти. URL: https://dspace.hnpu.edu.ua/items/7d7ba087-8f20-43ad-9a20-6e5c2d3a2050 (дата звернення: 02.05.2026).
32. Benefits of digital learning application at higher education. URL: https://eprints.zu.edu.ua/32230/1/9.pdf (дата звернення: 02.05.2026).
33. Impact of digital technologies on the quality of higher education. URL: https://revistaeduweb.org/index.php/eduweb/article/download/673/1048?inline=1 (дата звернення: 02.05.2026).
34. Why Universities Must Embrace Digital Transformation. URL: https://www.timeshighereducation.com/hub/intersystems/p/why-universities-must-embrace-digital-transformation (дата звернення: 02.05.2026).
35. Understanding Digital Transformation in Higher Education. URL: https://moderncampus.com/blog/digital-transformation-in-higher-education.html (дата звернення: 02.05.2026).
36. How Digital Transformation Benefits Higher Education. URL: https://gammagroup.co/resources/blog/digital-transformation-higher-education/ (дата звернення: 02.05.2026).
37. React repository on GitHub. URL: https://github.com/reactjs/react.dev (дата звернення: 02.05.2026).
38. Mongoose documentation repository. URL: https://github.com/Automattic/mongoose/blob/master/docs/documents.md (дата звернення: 02.05.2026).
39. Новий Закон України «Про вищу освіту» — Еразмус+. URL: https://erasmusplus.org.ua/news/novyj-zakon-ukrayiny-pro-vyshhu-osvitu/ (дата звернення: 02.05.2026).
40. Закон України «Про вищу освіту» — протокол. URL: https://protocol.ua/ua/pro_vishchu_osvitu/ (дата звернення: 02.05.2026).