

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**

Факультет ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ПОГОДЖЕНО

**Декан факультету Інформаційних
технологій**

_____ Ігор БОЛБОТ
(підпис)
« ____ » _____ 2026 р.

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

**Завідувач кафедри Комп'ютерних
систем, мереж та кібербезпеки**

_____ Дмитро КАСАТКІН
(підпис)
« ____ » _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

На тему: «Розроблення комп'ютерної системи забезпечення обліку і контролю
опрацювання курсів підвищення кваліфікації штатним персоналом

»

Спеціальність F7 «Комп'ютерна інженерія

»

Освітньо-професійна програма «Комп'ютерна інженерія

»

Гарант освітньої програми

канд. фіз.-мат. наук, доцент

(підпис)

Євгеній НІКІТЕНКО

Керівник бакалаврської

кваліфікаційної роботи

доктор техн. наук, професор

(підпис)

Вадим ШКАРУПИЛО

Виконав

(підпис)

Євгеній ВОРОНЦОВ

КИЇВ-2026

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**
**Факультет ІНФОРМАЦІЙНИХ
ТЕХНОЛОГІЙ**

ЗАТВЕРДЖУЮ
Завідувач кафедри
комп'ютерних систем, мереж та кібербезпеки
канд. пед. наук, доцент _____ Дмитро
КАСАТКІН
« _____ » _____ 20 ____ р.

З А В Д А Н Н Я

**ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
ЗДОБУВАЧУ**

Воронцов Євгеній Юрійович (прізвище, ім'я, по батькові)	
Спеціальність «Комп'ютерна інженерія»	»
Освітньо-професійна програма «Комп'ютерна інженерія»	»
Тема кваліфікаційної бакалаврської роботи «Розроблення комп'ютерної системи забезпечення обліку і контролю опрацювання курсів підвищення кваліфікації штатним персоналом»	
затверджена наказом ректора НУБіП України від «10» ____ 12 ____ 2025 р. № ____ 3072 «С» ____	
Термін подання завершеної роботи на кафедру «2» ____ 06 ____ 2026 р.	
Вихідні дані до бакалаврської кваліфікаційної роботи	
Перелік питань, що підлягають дослідженню:	
Перелік графічного матеріалу (за необхідності).	
Дата видачі завдання “ ____ 10 ____ ” ____ 12 ____ 2025 р.	

**Керівник бакалаврської
кваліфікаційної роботи**
доктор техн. наук, професор

_____ (підпис)

Вадим ШКАРУПИЛО

Завдання взяв до виконання

_____ (підпис)

Євгеній ВОРОНЦОВ

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Аналіз технічного завдання	07.02.2026 р. 18.02.2026 р.	Виконано
2	Проектування комп'ютерної системи	05.03.2026 р. 25.03.2026 р.	Виконано
3	Реалізація комп'ютерної системи	15.04.2026 р. 13.05.2026 р.	Виконано
4	Тестування комп'ютерної системи	10.05.2026 р. 15.05.2026 р.	Виконано
5	Оформлення пояснювальної записки	16.05.2026 р. 20.05.2026 р.	Виконано

Студент

(підпис)

Свгеній ВОРОНЦОВ

(Ініціали та прізвище)

Керівник проекту
(роботи)

(підпис)

Вадим ШКАРУПИЛО

(Ініціали та прізвище)

РЕФЕРАТ

Пояснювальна записка: 70 сторінок, 21 рисунки, 2 таблиці, 5 лістингів, 20 джерел.

КОМП'ЮТЕРНА СИСТЕМА, ОБЛІК ПЕРСОНАЛУ, ПІДВИЩЕННЯ КВАЛІФІКАЦІЇ, JAVA, SPRING BOOT, JAVASCRIPT, SPA, REST API, БАЗА ДАНИХ

Об'єкт аналізу - Процес забезпечення обліку і контролю опрацювання курсів підвищення кваліфікації штатним персоналом підприємства.

Мета роботи - Розроблення комп'ютерної системи забезпечення обліку і контролю опрацювання курсів підвищення кваліфікації штатним персоналом.

Перший розділ присвячено аналізу предметної області, формулюванню вимог до системи, огляду існуючих програмних рішень та їх недолікам.

У другому розділі описано вибір програмних технологій, архітектурних рішень, обґрунтування використання мови Java, фреймворку Spring Boot, бази даних та веб-технологій (HTML, CSS, JavaScript) для досягнення поставлених задач.

У третьому розділі представлено повну практичну реалізацію системи (клієнтську частину у вигляді Single Page Application, серверну частину з REST API та структуру бази даних), із наведенням роз'яснень, частин коду, блок-схем, налаштувань для розгортання на сервері та рисунків з поетапними результатами роботи веб-інтерфейсу.

У результаті роботи розроблено комп'ютерну систему забезпечення обліку і контролю опрацювання курсів підвищення кваліфікації штатним персоналом.

ЗМІСТ

ВСТУП	8
1	8
1.1	8
1.2	9
1.3	13
1.4	15
1.4.1 Функціональні вимоги	17
1.4.2 Нефункціональні вимоги	19
2	23
2.1	23
2.2 Архітектура REST API та структура коду	25
2.3 Система управління базами даних H2	27
2.4 Обґрунтування вибору серверних технологій	29
2.5 Проєктування інтерфейсу користувача	31
Висновок до розділу 2	38
3	39
3.1	39
3.2 Реалізація механізмів введення та валідації даних	42
3.2	43
3.3	46
4	55
4.1	55
4.2 Результати тестування	59
4.2	61
ВИСНОВКИ	67
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	70

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

API - Application Programming Interface, програмний інтерфейс додатку

CSS - Cascading Style Sheets, каскадні таблиці стилів

DOM - Document Object Model, об'єктна модель документа

HTML - HyperText Markup Language, мова розмітки гіпертексту

HTTP - Hypertext Transfer Protocol, протокол передачі гіпертексту

HTTPS - Hypertext Transfer Protocol Secure, безпечний протокол

передачі гіпертексту

IDE - Integrated Development Environment, інтегроване середовище розробки

JS - JavaScript, мова програмування

JSON - JavaScript Object Notation, текстовий формат обміну даними

JVM - Java Virtual Machine, віртуальна машина Java

PaaS - Platform as a Service, платформа як послуга

REST - Representational State Transfer, архітектурний стиль взаємодії

КОМПОНЕНТІВ

SPA - Single Page Application, односторінковий веб-додаток

SQL - Structured Query Language, мова структурованих запитів

UI - User Interface, інтерфейс користувача

UX - User Experience, користувацький досвід

LMS – Learning Management Systems

ВСТУП

В епоху стрімкого розвитку інформаційних технологій та цифровізації бізнес-процесів все більшої актуальності набувають системи автоматизованого управління людськими ресурсами. Здатність компаній швидко адаптуватися до змін на ринку напряду залежить від кваліфікації їхніх працівників, що робить безперервне навчання та розвиток персоналу одним із ключових факторів успіху сучасного підприємства. В умовах високої конкуренції особливо гостро постає питання ефективного обліку та контролю процесу підвищення кваліфікації співробітників. Своєчасний моніторинг проходження навчальних курсів дозволяє оптимізувати процеси управління персоналом, виявляти прогалини у знаннях та забезпечувати відповідність компетенцій працівників поточним потребам бізнесу. Відсутність централізованої інформаційної системи обліку навчання часто призводить до втрати важливих даних, дублювання функцій та неефективного використання робочого часу керівників. Впровадження автоматизованих систем збору та збереження даних щодо освітнього прогресу сприяє підвищенню продуктивності та прозорості управління компанією в цілому.

Важливим аспектом сучасних корпоративних систем є забезпечення зручного доступу та наочної візуалізації зібраних даних, що дозволяє своєчасно реагувати на кадрові потреби та приймати обґрунтовані управлінські рішення. Використання сучасних вебтехнологій відкриває нові можливості для створення інтерактивних та кросплатформних інтерфейсів взаємодії з корпоративними базами даних.

1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

1.1 Вимоги до розроблення системи

Управління даними та клієнт-серверна взаємодія:

- система має забезпечувати можливість створення, читання, оновлення (зміна статусів) та видалення інформації про працівників компанії і доступні навчальні курси.

- обмін даними між клієнтським вебінтерфейсом та сервером повинен здійснюватися через REST API по мережі

Збереження даних у структурованому форматі:

- інформація про персонал, каталог курсів та історію призначень повинна надійно зберігатися у реляційній базі даних на сервері.

- збережені записи мають містити вичерпну інформацію про зв'язок конкретного працівника з курсом, його поточний статус навчання, відсоток прогресу та дедлайн

Можливість моніторингу, управління та розмежування прав доступу через зручний інтерфейс:

- вебдодаток повинен надавати користувачеві можливість зручно переглядати загальну аналітику на головному дашборді;

- система повинна мати функціонал для розмежування ролей користувачів: "Адміністратор" та "Глядач"

- інтерфейс має бути наочним та інформативним, підтримувати фільтрацію даних за статусами виконання та забезпечувати швидкий доступ до історії останніх оновлень.

Представлені вимоги визначають основні можливості та функції, які має виконувати система, що розробляється. Вони є ключовими для забезпечення коректної роботи та задоволення потреб корис

1.2 Огляд існуючих систем

На сьогоднішній день існує низка аналогічних систем, призначених для управління процесом навчання та обліку підвищення кваліфікації персоналу. Одним із найпопулярніших рішень є використання систем управління навчанням, наприклад, таких як Moodle, TalentLMS та iSpring Learn.

Moodle - це модульне об'єктно-орієнтоване динамічне навчальне середовище, що дає змогу створювати, управляти та відстежувати навчальні курси для співробітників. Система забезпечує зручну організацію освітніх матеріалів та автоматизований контроль за прогресом персоналу.

Можливість розширення функціоналу за допомогою модулів дає змогу проводити онлайн-тестування і аналіз результатів у міру їх надходження.

Moodle часто використовують для створення корпоративних платформ навчання, які потребують детальної аналітики успішності.

Переваги Moodle:

- легко налаштовані інструменти для створення курсів та призначення їх працівникам;
- візуалізація прогресу навчання та статусу співробітників у режимі реального часу;
- агрегація даних про успішність персоналу та формування детальних звітів;
- використання вбудованих інструментів для тестування та оцінювання компетенцій;
- створення повноцінної системи дистанційного навчання з можливістю розгортання на власних серверах підприємства для збереження конфіденційності даних;

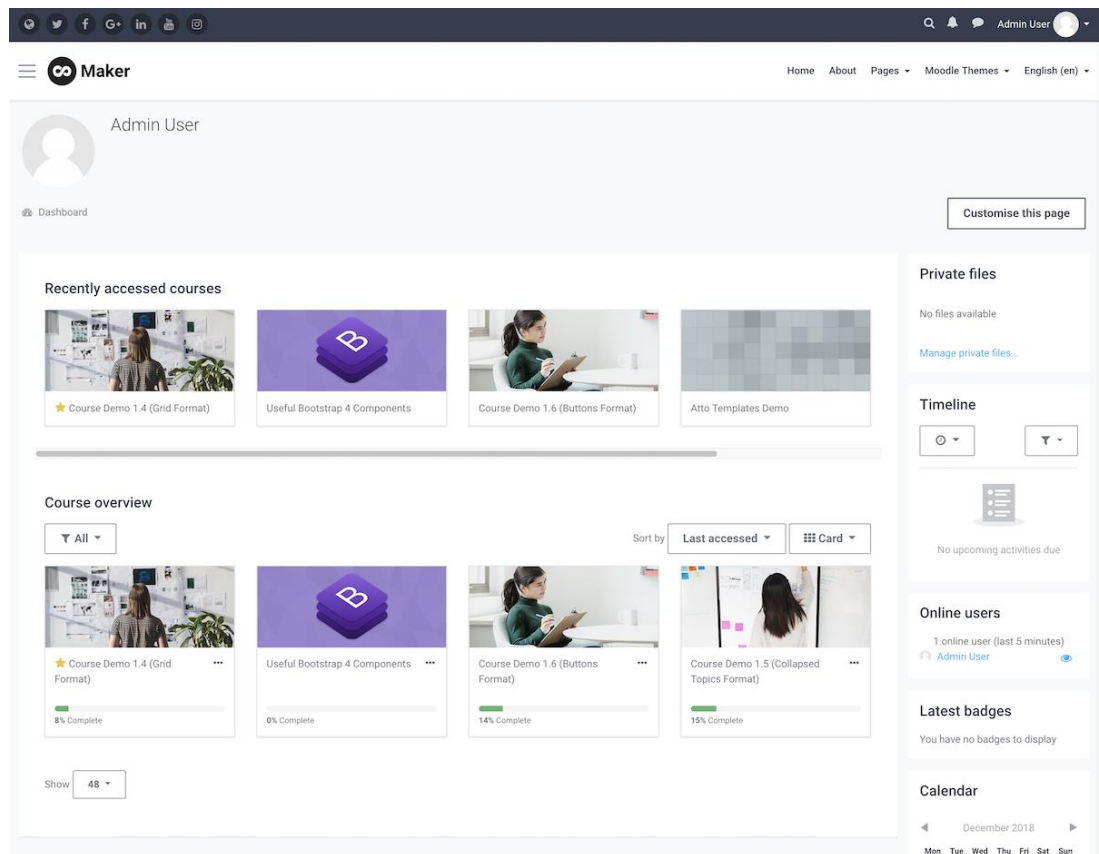


Рисунок 1.1 - Інтерфейс веб-додатку Moodle

TalentLMS – це хмарна платформа для корпоративного навчання, яка відрізняється простотою розгортання та інтуїтивно зрозумілим інтерфейсом. Вона орієнтована на швидкий запуск освітніх програм для персоналу без необхідності залучення технічних спеціалістів чи підтримки власної серверної інфраструктури.

Переваги TalentLMS:

- швидке налаштування та запуск системи завдяки хмарній архітектурі (SaaS) без потреби адміністрування локальних серверів;
- інтуїтивно зрозумілий та адаптивний інтерфейс користувача, що значно спрощує взаємодію із системою як для адміністраторів, так і для працівників;
- наявність вбудованих інструментів гейміфікації (нарахування балів, видача бейджів, формування рейтингів) для підвищення мотивації та залученості персоналу до процесу підвищення кваліфікації;
- повноцінна підтримка мобільного навчання через спеціалізовані

додатки, що забезпечує безперервний доступ до навчальних матеріалів з будь-якого пристрою;

широкі можливості інтеграції з популярними корпоративними інструментами, такими як системи відеоконференцій, CRM та кадрового обліку;

- можливість гнучкого налаштування зовнішнього вигляду платформи під корпоративний стиль компанії (брендування);

- зручні вбудовані інструменти для створення інтерактивних тестів, опитувань та автоматичної генерації аналітичної звітності щодо успішності проходження курсів.

iSpring Learn - це потужна хмарна система управління корпоративним навчанням, яка спеціалізується на швидкому запуску освітніх процесів та відрізняється глибокою інтеграцією з інструментами для створення електронного контенту. Вона особливо ефективна для компаній, які вже мають значну базу навчальних матеріалів у форматах презентацій, документів та відео, і потребують швидкої їх конвертації в інтерактивні курси.

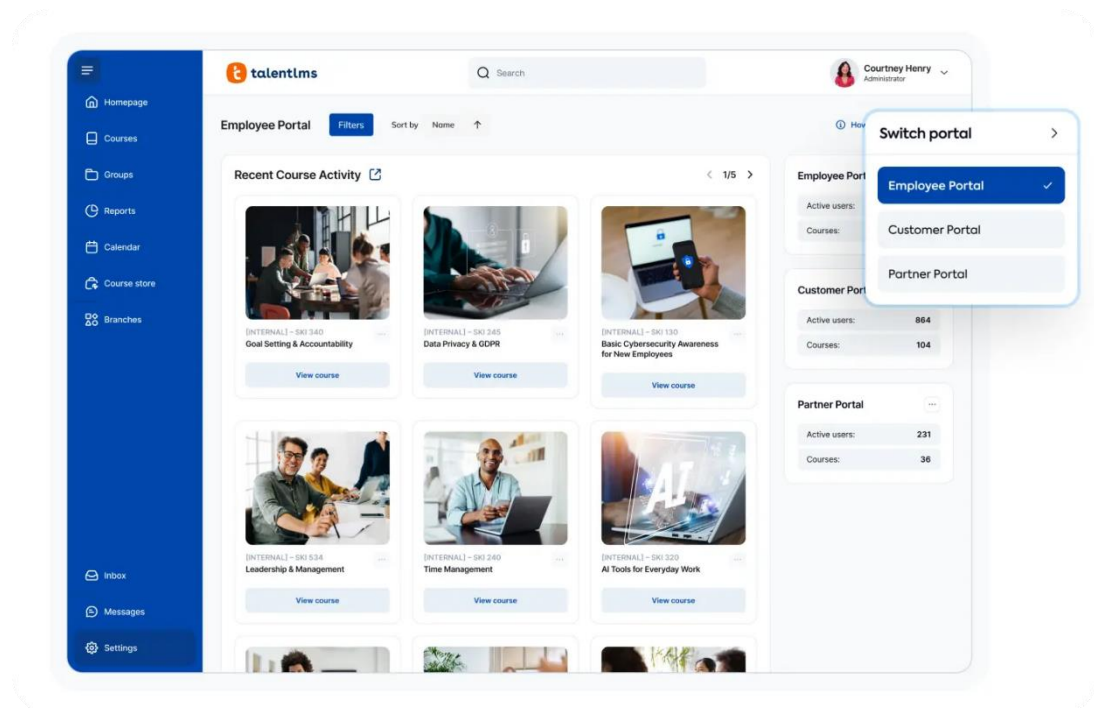


Рисунок 1.2 - Інтерфейс TalentLMS

Переваги iSpring Learn:

- повна та безшовна інтеграція з редактором iSpring Suite, що дозволяє

швидко перетворювати звичайні презентації (PowerPoint) на професійні інтерактивні курси з тестами та діалогами без залучення програмістів;

- високий рівень автоматизації процесів навчання: можливість автоматичного призначення курсів залежно від посади, приналежності до відділу або терміну роботи співробітника в компанії;

- потужна система розширеної аналітики та звітності, яка дозволяє відстежувати прогрес персоналу аж до кількості спроб проходження тесту та часу, витраченого на вивчення кожного окремого слайду;

- наявність повноцінного мобільного додатку з можливістю завантаження навчальних матеріалів на пристрій для подальшого проходження курсів у режимі офлайн;

- підтримка ключових міжнародних стандартів електронного навчання (SCORM, xAPI, cmi5), що гарантує сумісність із різноманітними готовими курсами від сторонніх розробників;

- вбудовані можливості для планування та проведення вебінарів, а також інтеграція з популярними сервісами відеозв'язку для організації змішаного (синхронного та асинхронного) формату навчання.

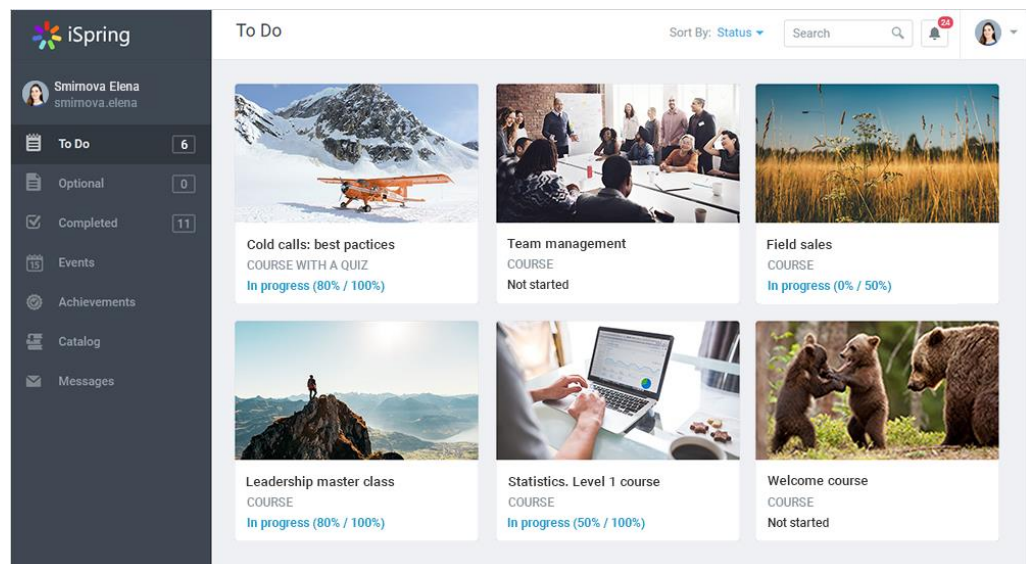


Рисунок 1.3 - Інтерфейс iSpring Learn

Недоліки подібних систем:

- залежність від сторонніх хмарних сервісів: Використання готових SaaS-платформ для навчання створює залежність від провайдерів цих послуг.

Зміна цінової політики, умов надання послуг або припинення підтримки сервісу з боку компанії-розробника може призвести до недоступності навчальних матеріалів та втрати історії підвищення кваліфікації персоналу. Це може бути вкрай незручним і вимагати додаткових фінансових і часових витрат на міграцію до іншої системи;

- обмежений функціонал безкоштовних або базових версій: Безкоштовні та стартові тарифи хмарних LMS зазвичай мають суттєві обмеження, наприклад, жорсткий ліміт на кількість зареєстрованих працівників, доступний обсяг пам'яті для завантаження курсів або відсутність розширеної аналітики. Для доступу до повноцінного функціоналу та генерації кастомних звітів необхідна дорога платна підписка;

- обмежені можливості налаштування: Жорстко визначений набір інструментів і функцій комерційних платформ може бути недостатнім для задоволення конкретних бізнес-вимог підприємства. Наприклад, відсутність можливості глибокої інтеграції з унікальними внутрішніми кадровими системами або неможливість зміни логіки призначення курсів можуть обмежити адаптацію системи під специфічні корпоративні процеси;

- обмежений контроль за даними: Використання хмарних платформ означає передачу і зберігання персональних даних співробітників, статистики їхньої успішності та внутрішніх корпоративних матеріалів на серверах провайдера. Це може викликати обґрунтовані побоювання з приводу конфіденційності та інформаційної безпеки. Обмежений контроль з боку підприємства над власними даними є критичною проблемою в разі необхідності суворого захисту комерційної таємниці;

1.3 Обґрунтування необхідності розроблення системи

У сучасних умовах ведення бізнесу безперервне навчання та підвищення кваліфікації персоналу є критично важливими процесами для підтримки конкурентоспроможності підприємства. Проте організація та облік цих процесів часто стикаються з низкою організаційних та технічних труднощів,

які знижують загальну ефективність управління людськими ресурсами.

Перша проблема полягає у використанні неефективних методів ручного обліку. Велика кількість компаній малого та середнього бізнесу досі використовують неспеціалізоване програмне забезпечення (електронні таблиці MS Excel, Google Sheets) або паперові документи для фіксації призначених та пройдених курсів. Такий підхід характеризується високою ймовірністю виникнення людської помилки (випадкове видалення або пошкодження даних), складністю паралельної роботи кількох адміністраторів, відсутністю автоматичного відстеження дедлайнів та неможливістю швидкого візуального аналізу успішності. При збільшенні штату працівників ручне адміністрування стає неконтрольованим і призводить до хаосу в даних.

Друга проблема пов'язана з недоліками існуючих спеціалізованих рішень на ринку. Як було встановлено під час аналізу предметної області, повноцінні системи управління навчанням (LMS) є занадто складними та надлишковими за своїм функціоналом для компаній, яким потрібен виключно базовий облік призначень та контроль статусів. Комерційні платформи вимагають значних і регулярних фінансових витрат на оплату ліцензій (вартість яких масштабується залежно від кількості працівників). Крім того, використання сторонніх хмарних рішень змушує компанії передавати внутрішню інформацію та персональні дані співробітників на сервери третіх осіб, що часто суперечить внутрішнім політикам інформаційної безпеки та конфіденційності.

Отже, основна проблема полягає у відсутності легкої, економічно вигідної та незалежної системи, яка б дозволяла централізовано вести каталог курсів та контролювати процес навчання персоналу в єдиному зручному інтерфейсі без необхідності розгортання та адміністрування важкої інфраструктури.

Розроблена комп'ютерна система безпосередньо вирішує цю проблему шляхом:

- повної відмови від ручного ведення неструктурованих таблиць на

користь централізованої бази даних, що гарантує цілісність та швидкий пошук необхідної інформації;

- надання інтуїтивно зрозумілого веб-інтерфейсу (за архітектурою SPA), який дозволяє адміністраторам миттєво призначати курси, встановлювати дедлайни та відстежувати прогрес без необхідності проходити тривале навчання роботі з програмою;

- забезпечення технологічної незалежності підприємства, оскільки розроблена система не має прив'язки до платних ліцензій і може бути безкоштовно розгорнута як на локальних комп'ютерах компанії, так і на приватних хмарних серверах, що гарантує повний контроль над конфіденційними даними.



Рисунок 1.1 - Порівняльний аналіз підходів до організації обліку навчання персоналу

1.4 Визначення основних вимог до комп'ютерної системи

Для успішної реалізації комп'ютерної системи обліку та контролю

опрацювання курсів підвищення кваліфікації необхідно чітко сформулювати вимоги, яким вона повинна відповідати. Ці вимоги поділяються на функціональні (описують поведінку системи та її можливості) та нефункціональні (визначають атрибути якості, технологічні обмеження та архітектурні особливості).

1.4.1 Функціональні вимоги

Для забезпечення комплексного виконання завдань з обліку та контролю підвищення кваліфікації персоналу, функціонал системи логічно розділено на п'ять основних модулів. Система повинна забезпечувати безперебійне виконання функцій у межах кожного з описаних нижче модулів:

Перший модуль відповідає за авторизацію та розмежування прав доступу. Він повинен забезпечувати вхід у систему з вибором однієї з двох ролей, а саме адміністратор або глядач. Важливою вимогою є обмеження доступу до функцій модифікації даних, де роль глядача має доступ виключно до перегляду інформації у всіх розділах без можливості внесення змін. Натомість роль адміністратора повинна мати повні привілеї, що включають додавання, редагування статусів та видалення будь-яких записів у базі даних. Також система має динамічно приховувати або відображати елементи керування, такі як кнопки видалення чи форми додавання, залежно від поточної активної ролі користувача.

Другий модуль призначений для управління персоналом підприємства. Його функціонал включає виведення загального списку співробітників у вигляді структурованої таблиці, а також зберігання та обробку атрибутів для кожного працівника, зокрема унікального ідентифікатора, прізвища, імені, по батькові, займаної посади, назви структурного підрозділу та контактної електронної адреси. Крім того, модуль має забезпечувати реєстрацію нових співробітників через спеціалізоване модальне вікно з перевіркою введених даних і безпечне видалення запису про працівника. При видаленні

співробітника з бази даних система повинна автоматично знаходити та анулювати всі пов'язані з ним записи про призначення навчальних курсів для уникнення накопичення неактуальних даних.

Третій модуль забезпечує формування каталогу курсів. Ця частина системи має відображати перелік доступних навчальних програм у вигляді інтерактивних карток та обробляти ключові параметри курсу, серед яких назва програми, провайдер навчання, тривалість у годинах, рівень складності та детальний текстовий опис. Передбачається можливість створення нових навчальних програм за допомогою відповідної форми та видалення курсу з каталогу із забезпеченням цілісності даних, що означає автоматичне скасування всіх призначень цього курсу для всіх працівників у разі його видалення.

Четвертий модуль відповідає за контроль та облік навчального процесу. Він повинен дозволяти призначення обраного курсу з каталогу конкретному співробітнику із зазначенням обов'язкового терміну виконання. Необхідною вимогою є захист від дублювання, тобто автоматична перевірка та блокування спроби повторного призначення одного й того ж курсу працівнику. Система має відстежувати та змінювати етапи проходження курсу за трьома базовими статусами: заплановано, в процесі та завершено. Прогрес повинен візуалізуватися за допомогою графічних індикаторів та кольорових міток. Також вимагається наявність інструментів фільтрації загального списку призначень за їхнім поточним статусом для зручного пошуку інформації.

П'ятий модуль реалізує аналітику та моніторинг активності на головній інформаційній панелі. Він має агрегувати та виводити ключові показники ефективності в режимі реального часу, зокрема загальну кількість зареєстрованих працівників та доступних курсів. Система повинна автоматично розраховувати загальний рівень успішності як відсоткове співвідношення повністю завершених курсів до загальної кількості всіх активних призначень. Додатково передбачається ведення журналу останніх

дій, де автоматично фіксується та виводиться інформація про ключові події, такі як додавання співробітника чи призначення курсу, із зазначенням точного часу події.

1.4.2 Нефункціональні вимоги

Окрім функціональних можливостей, розроблювана комп'ютерна система повинна відповідати ряду нефункціональних вимог. Вони визначають її архітектурні особливості, продуктивність, зручність використання та технологічні обмеження, які є критично важливими для забезпечення стабільності програмного продукту.

Вимоги до архітектури системи передбачають чіткий логічний та фізичний розподіл між клієнтською фронтенд-частиною та серверною бекенд-частиною. Взаємодія між цими ізольованими компонентами має здійснюватися виключно через мережевий протокол з використанням архітектурного стилю REST. Серверна частина повинна виступати в ролі постачальника програмного інтерфейсу (API), який приймає та повертає дані у стандартизованому легкому текстовому форматі JSON. Такий підхід гарантує незалежність модулів системи та дозволяє в майбутньому легко масштабувати проєкт або підключати сторонні сервіси без необхідності переписування базової логіки.

Вимоги до користувацького інтерфейсу та ергономіки полягають у застосуванні концепції односторінкового вебдодатка. Це означає, що початкове завантаження базової структури сторінки (HTML-каркасу) повинно відбуватися лише один раз при першому вході в систему. Усі подальші оновлення контенту, переходи між розділами обліку персоналу, каталогу курсів чи загального дашборду мають виконуватися асинхронно у фоновому режимі без повного перезавантаження сторінки браузера. Завдяки цьому підходу мінімізується навантаження на мережу та досягається висока

швидкість відгуку, що робить роботу в браузері максимально наближеною до використання класичних десктопних програм. Додатковою вимогою до інтерфейсу є його адаптивність, тобто здатність автоматично підлаштовувати розташування та розміри елементів під різні розміри екранів, зберігаючи при цьому читабельність і зручність навігації.

Вимоги до продуктивності та управління даними обумовлені специфікою розробки демонстраційного прототипу. Для забезпечення максимальної швидкодії під час тестування роботи системи, висувається вимога щодо використання технології зберігання даних безпосередньо в оперативній пам'яті сервера (in-memory database). Таке рішення усуває затримки, пов'язані зі зверненням до жорсткого диска, та знімає необхідність розгортання окремого громіздкого сервера баз даних. Час відгуку системи на типові дії користувача, такі як фільтрація списку призначень або збереження нового курсу, має бути мінімальним для забезпечення безперервного робочого процесу. Крім того, такий підхід гарантує отримання абсолютно чистого середовища при кожному новому запуску системи, що є оптимальним для етапу презентації проєкту.

Вимоги до розгортання та надійності передбачають високу портативність програмного забезпечення. Програмний код має бути структурований таким чином, щоб після етапу компіляції клієнтська частина автоматично об'єднувалася із серверною логікою в єдиний виконуваний файл. Це значно спростить процес встановлення системи як на локальних машинах, так і в хмарних середовищах. Додаток повинен мати здатність автономно запускатися та відновлювати свою роботу на сервері без ручного втручання розробника або системного адміністратора, що є необхідною умовою для використання сучасних платформ хмарного хостингу.

1.4.3 Вимоги до програмно-апаратного середовища

Для забезпечення безперебійної, надійної та коректної роботи

комп'ютерної системи необхідно дотримуватися чітко визначених вимог щодо апаратного забезпечення та програмного середовища як на стороні сервера, так і на стороні клієнта. Оскільки архітектура розробленого рішення базується на клієнт-серверній моделі взаємодії, ці вимоги розділяються на дві відповідні категорії.

Вимоги до апаратної частини сервера є відносно невисокими завдяки оптимізації коду та використанню реляційної бази даних, що працює виключно в оперативній пам'яті. Для стабільного функціонування серверної частини (бекенду) необхідна обчислювальна машина або віртуальний виділений сервер із процесором архітектури x86-64 або ARM. Мінімальний обсяг оперативної пам'яті повинен складати не менше 512 мегабайт, хоча для комфортної роботи в умовах збільшення кількості користувачів та накопичення даних рекомендується виділити від 1 гігабайта оперативної пам'яті. Вимоги до обсягу постійної пам'яті є мінімальними і складають близько 100 мегабайт вільного дискового простору, які необхідні безпосередньо для розміщення скомпільованого виконуваного файлу додатка

Щодо програмного середовища сервера, головною та єдиною жорсткою вимогою є наявність попередньо встановленого середовища виконання віртуальної машини Java (Java Runtime Environment) версії 17 або вище. Завдяки кросплатформній природі технології Java, серверна частина не має прив'язки до конкретної операційної системи. Вона може успішно функціонувати під управлінням операційних систем сімейства Linux, Windows або macOS. Крім того, архітектура додатка повністю адаптована для швидкого розгортання у сучасних хмарних середовищах типу Platform as a Service (PaaS) або ізольованих контейнерах.

Вимоги до клієнтської частини системи зумовлені використанням архітектури односторінкового додатка, що передбачає виконання значної частини логіки відображення безпосередньо на пристрої кінцевого користувача. Апаратно клієнтським пристроєм може виступати персональний комп'ютер, ноутбук, планшет або сучасний смартфон. Апаратні

характеристики пристрою не є критичними, достатньо потужностей для комфортної роботи операційної системи користувача.

Головною вимогою до програмного забезпечення на стороні клієнта є наявність актуальної версії сучасного веббраузера (наприклад, Google Chrome, Mozilla Firefox, Safari або Microsoft Edge). Веббраузер обов'язково повинен підтримувати сучасний стандарт мови ECMAScript 6 та мати активовану функцію виконання сценаріїв JavaScript. Саме ці сценарії відповідають за асинхронний обмін даними з сервером у фоновому режимі та динамічну розбудову користувацького інтерфейсу без перезавантаження сторінки. Встановлення будь-яких додаткових плагінів чи стороннього програмного забезпечення кінцевому користувачу не потрібно.

Для забезпечення зв'язку між клієнтськими пристроями та сервером необхідне стабільне підключення до корпоративної локальної мережі підприємства або до глобальної мережі Інтернет. Взаємодія відбувається за стандартними мережевими протоколами передачі гіпертексту. Вимоги до пропускної здатності мережі є мінімальними, оскільки при клієнт-серверній взаємодії передаються виключно легкі текстові структури даних у форматі JSON, обсяг яких становить лише кілька кілобайтів для кожного запиту.

Таблиця 1.1 Вимоги до програмно-апаратного середовища

Характеристика	Серверна частина	Клієнтська частина
ОЗП	Від 512 МБ	Нічого
Постійна пам'ять	100 МБ вільного простору	Не потребує встановлення
Операційна система	Кросплатформна	ОС із підтримкою браузера
Програмне забезпечення	Java Runtime Environment	Сучасний веббраузер

Додаткові вимоги	Підтримка хмарного розгортання	Сучасний веббраузер
Мережева взаємодія	Підключення до інтернету	Увімкнена підтримка JavaScript
Апаратна платформа	x86-64 або ARM	ПК, ноутбук, смартфон

2 ПРОЄКТУВАННЯ СИСТЕМИ

2.1 Фреймворк Spring Boot

Основою серверної частини розробленої системи є Spring Boot. Це сучасний та потужний фреймворк екосистеми Java, який призначений для швидкого створення повноцінних та надійних вебдодатків. Він базується на принципі конвенції понад конфігурацією, що дозволяє розробнику зосередитися на написанні бізнес-логіки та мінімізувати кількість ручних налаштувань. Програмний код серверної частини написано мовою Java з використанням стандартів об'єктно-орієнтованого програмування.

Ключовою особливістю Spring Boot у контексті даного проєкту є наявність вбудованого вебсервера Apache Tomcat. Це повністю усуває необхідність встановлення, складного налаштування та адміністрування додаткових серверних середовищ на цільовій машині. Весь додаток компілюється у єдиний архівний файл, який можна легко запустити на будь-якому комп'ютері або хмарному хостингу, де встановлено середовище виконання Java. Управління всіма підключеними бібліотеками та автоматична збірка проєкту здійснюється за допомогою «Maven». Див. табл. 2.1

Назва	Характеристика
Базова мова програмування	Java (ver.17 та вище)
Архітектурний підхід	REST, MVC
Вбудований сервер	Apache Tomcat (порт 8080)
Управління залежностями	Maven/ Gradle
Формат обміну даних	JSON
Підтримка баз даних	Spring Data JPA, Hibernate
Безпека	Spring Security
Кросплатформеність	JVM Підтримується

Таблиця 2.1 Характеристики Spring Boot

2.2 Архітектура REST API та структура коду

Взаємодія між клієнтською частиною, тобто інтерфейсом користувача, та сервером побудована за сучасним архітектурним стилем REST. Цей підхід забезпечує гнучкість та повну незалежність фронтенду від бекенду. Обмін даними відбувається через мережу за допомогою стандартних методів протоколу HTTP, кожен з яких суворо відповідає за конкретну операцію над інформацією у базі даних.

Для отримання інформації система використовує метод GET. Завдяки йому клієнтська частина завантажує актуальні списки співробітників підприємства, доступних навчальних курсів та історію призначень. Коли адміністратор системи створює новий запис, наприклад, додає нового працівника або формує новий курс, браузер відправляє на сервер запит із методом POST, який у своєму тілі містить усі введені користувачем дані для безпечного збереження.

Для зміни існуючої інформації застосовується метод PUT. У межах розробленої системи він використовується переважно для оновлення поточного прогресу навчання, дозволяючи фіксувати перехід від запланованого стану до процесу виконання або успішного завершення курсу. У разі необхідності очищення системи від неактуальної інформації, наприклад, при звільненні співробітника або скасуванні навчальної програми, клієнт формує запит із методом DELETE, який ініціює повне видалення відповідного запису з бази даних сервера.

Внутрішня архітектура серверного коду побудована за класичним багаторівневим патерном, що забезпечує чітке розділення обов'язків між програмними компонентами. Першим базовим рівнем є шар моделей даних. Він містить класи мови Java, які безпосередньо описують головні сутності предметної області, їхні характеристики та зв'язки. Саме на основі цих класів фреймворк автоматично визначає, які таблиці та колонки потрібно згенерувати в базі даних.

Другим рівнем є шар доступу до даних, який реалізований через систему

репозиторіїв. Завдяки використанню технології Spring Data JPA зникає необхідність писати низькорівневі SQL-запити вручну. Репозиторії надають готові надійні механізми для пошуку, фільтрації, збереження та видалення інформації, виступаючи мостом між об'єктами в коді та реальними записами в оперативній пам'яті.

Третім і найвищим рівнем виступає шар вебконтролерів. Контролери виконують ключову функцію маршрутизаторів та єдиних точок доступу для зовнішніх запитів. Коли клієнтський скрипт надсилає HTTP-запит, відповідний контролер перехоплює його, аналізує передані параметри, звертається до необхідного репозиторію для виконання операції з даними, а потім формує відповідь. Результат роботи будь-якого контролера завжди повертається клієнту в універсальному текстовому форматі JSON. Цей формат є стандартом для сучасних вебдодатків, оскільки він легко зчитується та миттєво обробляється клієнтськими алгоритмами для оновлення графічного інтерфейсу без необхідності перезавантаження всієї сторінки. (Див. Рисунок 2.1)

```

@RestController
@RequestMapping("/api/courses")
@CrossOrigin(origins = "*")
public class CourseController {

    @Autowired
    private CourseRepository courseRepository;

    @GetMapping
    public List<Course> getAllCourses() {
        return courseRepository.findAll();
    }

    @PostMapping
    public Course createCourse(@RequestBody Course course) {
        return courseRepository.save(course);
    }

    @DeleteMapping("/{id}")
    public void deleteCourse(@PathVariable Long id) {
        courseRepository.deleteById(id);
    }
}

```

Рисунок 2.1 - Структура REST-контролера системи

2.3 Система управління базами даних H2

Організація збереження, обробки та видобування інформації в розробленій системі реалізована за допомогою реляційної бази даних H2 Database. Ця система управління базами даних повністю написана мовою Java, що забезпечує її глибоку та безконфліктну інтеграцію у програмний стек Spring Boot. Головною архітектурною відмінністю обраного рішення є використання режиму in-memory, який передбачає розгортання та функціонування бази даних безпосередньо в оперативній пам'яті сервера, виключаючи етап створення та постійного оновлення фізичних файлів на жорсткому диску

Взаємодія між програмним кодом та реляційною структурою бази даних відбувається за допомогою технології об'єктно-реляційного відображення (ORM). У даному проєкті цю функцію виконує стандарт Java Persistence API (JPA) та програмний фреймворк Hibernate. Використання цієї технології

дозволяє уникнути написання об'ємних низькорівневих SQL-запитів для стандартних операцій зі збереження чи читання інформації. Замість цього в коді створюються звичайні класи-моделі, які описують головні сутності предметної області: працівника, навчальний курс та запис про призначення навчання.

Під час кожного запуску серверної частини програми фреймворк Hibernate самостійно аналізує ці класи та їхні атрибути. На основі цього аналізу він автоматично генерує необхідні таблиці, колонки відповідних типів даних та зв'язки між ними у порожній оперативній пам'яті. Якщо користувач через веб-інтерфейс створює нового працівника, серверна частина формує об'єкт мовою Java, а Hibernate автоматично трансформує його у правильний SQL-запит і зберігає у відповідну таблицю бази даних H2.

Застосування такого підходу до управління даними має ряд суттєвих переваг для розробленої системи. Насамперед це гарантує надзвичайно високу продуктивність операцій. Оскільки всі процеси запису, читання та фільтрації даних відбуваються виключно в межах швидкодіючої оперативної пам'яті, час відгуку бази даних зводиться до мінімуму. Це забезпечує миттєву реакцію клієнтського інтерфейсу на будь-які команди користувача.

Також повністю усувається потреба у встановленні та адмініструванні важких окремих серверів баз даних, наприклад PostgreSQL або MySQL. Разом із цим відпадає необхідність налаштування мережових доступів між сервером додатку та сервером бази даних, оскільки вони функціонують як єдине ціле в одному процесі операційної системи. Це значно спрощує процес розгортання готового продукту на безкоштовних хмарних платформах.

Додатковою технічною зручністю є наявність вбудованої веб-консолі H2. Це інтегрований інструмент, який дозволяє адміністратору системи або розробнику підключитися до бази даних через звичайний браузер за спеціальним службовим маршрутом. Через цю консоль можна візуально перевірити структуру автоматично згенерованих таблиць, переглянути поточний стан даних або виконати прямі SQL-запити для налагодження

роботи системи.

Використання тимчасової бази даних є концептуально виправданим та ідеальним рішенням для демонстрації роботи прототипу розробленої системи. Оскільки дані не зберігаються на постійному носії, після кожного перезавантаження сервера або його виходу з режиму енергозбереження на хмарному хостингу, оперативна пам'ять очищується, і база даних автоматично повертається до свого початкового порожнього стану. Це дозволяє ефективно уникнути накопичення нерелевантних тестових записів. Таким чином гарантується, що під час кожної нової демонстрації або тестування система функціонуватиме у повністю чистому та готовому до роботи середовищі, відображаючи коректну логіку роботи з нуля.

2.4 Обґрунтування вибору серверних технологій

Фундаментальним рішенням під час проектування серверної частини став вибір мови програмування Java. Ця платформа є загальновизнаним індустріальним стандартом для розробки корпоративних систем завдяки своїй надійності, суворій типізації та високому рівню безпеки. Використання віртуальної машини Java забезпечує повну кросплатформність написаного коду. Це означає, що розроблена система може без жодних змін або перекомпіляцій працювати на будь-яких операційних системах, що значно спрощує процес впровадження та тестування програмного продукту на різних комп'ютерах.

Застосування фреймворку Spring Boot стало логічним кроком для прискорення розробки та уникнення рутинних процесів налаштування. На відміну від класичних рішень, які вимагають створення складних конфігураційних файлів та зовнішнього розгортання вебсерверів, Spring Boot пропонує парадигму самодостатнього додатка. Завдяки механізмам автоматичної конфігурації та впровадження залежностей система отримує чітку модульну структуру. Наявність інтегрованого сервера перетворює весь процес запуску бекенду на виконання одного збудованого архіву, що є

ідеальним рішенням для безкоштовного хмарного хостингу, який використовується в проєкті

Інтеграція бази даних H2 у режимі роботи з оперативною пам'яттю дозволила досягти максимальної незалежності системи від зовнішніх факторів. Розгортання проєкту не потребує попереднього встановлення важких систем управління базами даних на цільовий сервер, що робить додаток максимально портативним. При цьому використання технології об'єктно-реляційного відображення Hibernate гарантує правильну архітектуру додатку. У разі виникнення необхідності збереження даних на постійній основі та переходу на такі системи як PostgreSQL, достатньо буде лише змінити параметри підключення у файлі налаштувань без переписування основної бізнес-логіки.

Вибір на користь незалежного REST API з форматом обміну даними JSON забезпечує високу гнучкість системи на перспективу. Такий підхід повністю відокремлює серверну логіку від візуального інтерфейсу користувача. Завдяки цьому серверна частина перетворюється на універсальний постачальник інформації. У майбутньому це дозволить масштабувати проєкт і легко підключити до існуючого бекенду додаткові клієнтські рішення, наприклад, корпоративний мобільний додаток для персоналу, не вносячи абсолютно ніяких змін у вже написаний і протестований серверний код.

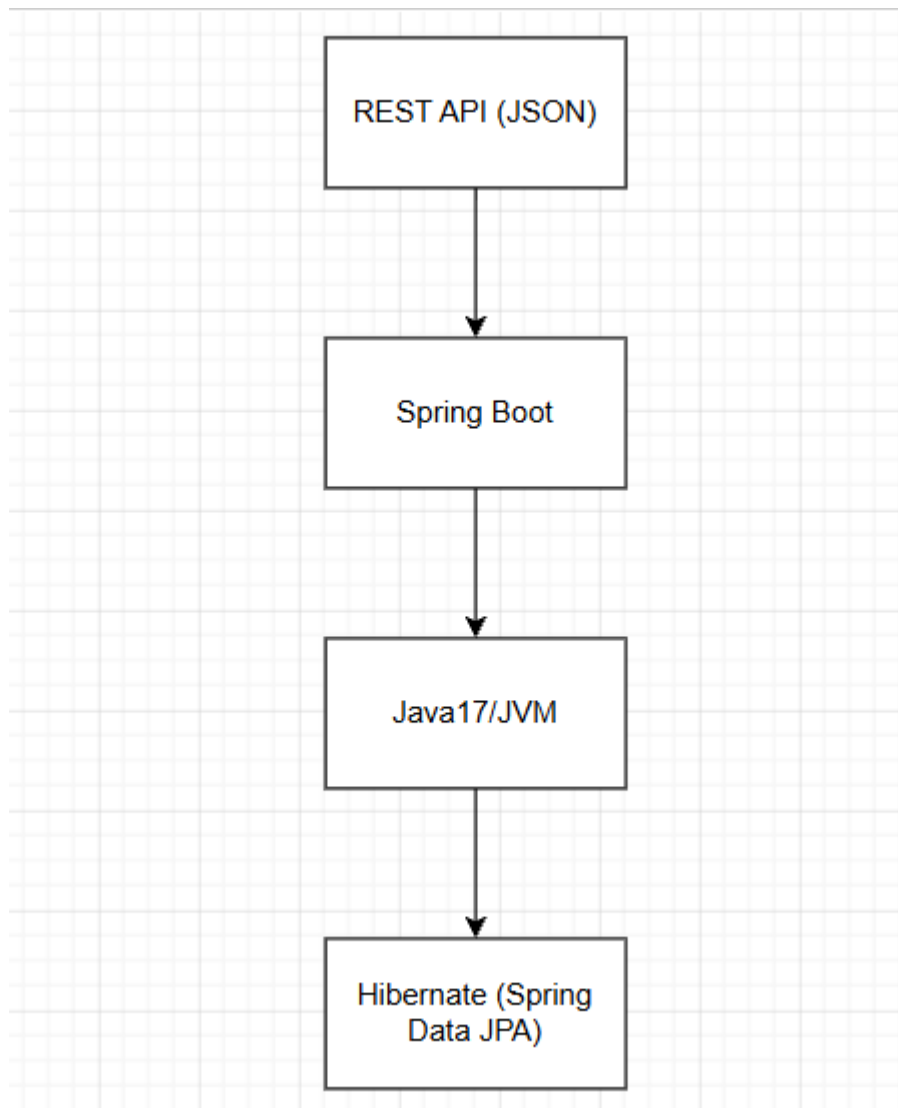


Рисунок 2.2 - Стэк технологій серверної частини системи

2.5 Проектування інтерфейсу користувача

Інтерфейс користувача є важливою складовою будь-якої комп'ютерної системи, особливо коли програмне забезпечення призначене для використання персоналом підприємства з різним рівнем технічної підготовки. У межах розробки вебдодатка для обліку та контролю підвищення кваліфікації особливу увагу було приділено простоті, доступності та інтуїтивності інтерфейсу. Основне завдання полягало в тому, щоб зробити процес управління персоналом та призначенням курсів максимально зручним і швидким.

Під час проектування інтерфейсу було використано підхід Single Page

Application, який відповідає сучасним тенденціям веброзробки. Основними принципами стали мінімалізм та чистота дизайну, де інтерфейс не перевантажений елементами, а на екрані одночасно відображається лише інформація, необхідна в поточному контексті. Також застосовано принцип єдиного вікна навігації, тому користувач не переходить між різними вебсторінками, замість цього контент динамічно оновлюється в основній робочій області.

Бічна навігаційна панель забезпечує швидкий доступ до всіх основних модулів системи незалежно від поточного екрана (як показано на рисунку 2.2). Використання модальних вікон дозволяє відкривати форми для додавання працівників, створення курсів та оновлення статусів поверх основного контенту, не змушуючи користувача залишати поточну сторінку (як показано на рисунку 2.3). Адаптивність забезпечується завдяки CSS-фреймворку Tailwind CSS, тому інтерфейс коректно масштабується як на екранах персональних комп'ютерів, так і на мобільних пристроях.

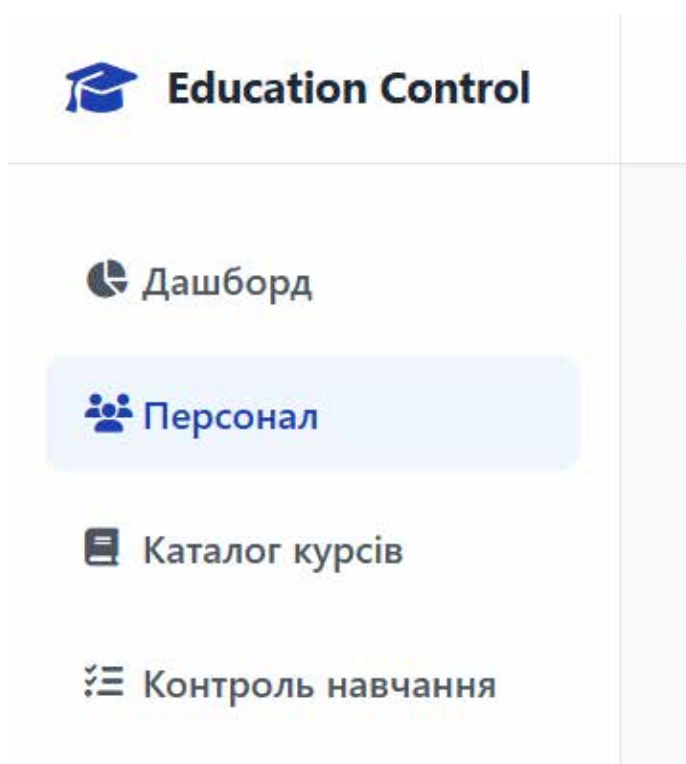
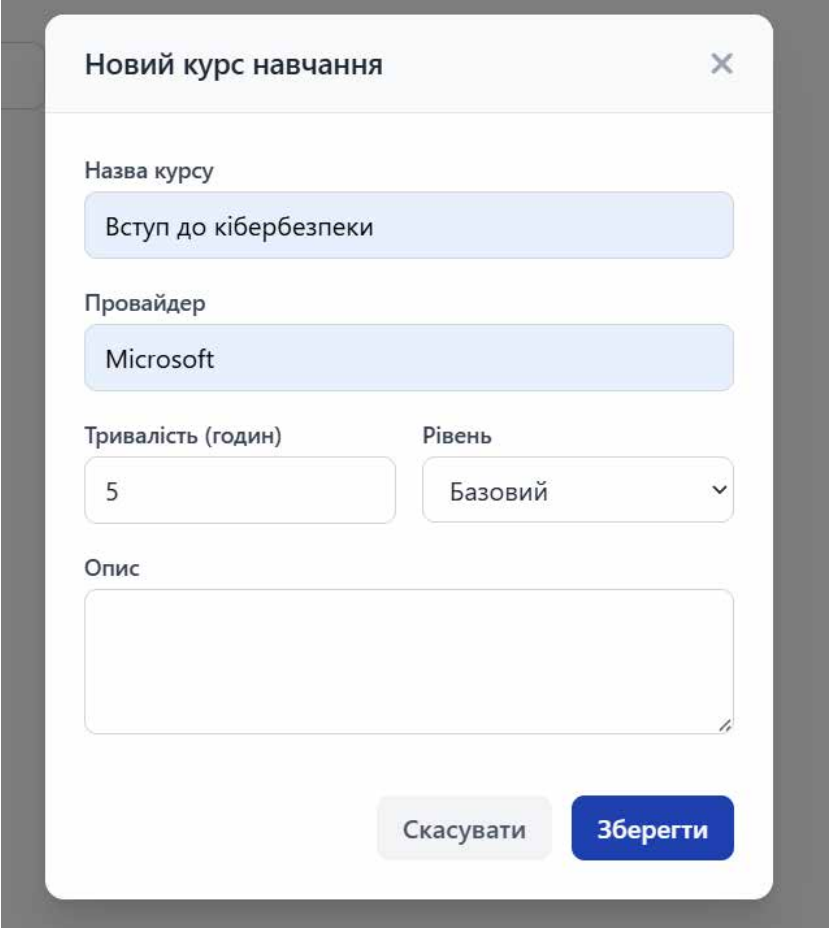


Рисунок 2.3 - Бічна навігаційна панель



Новий курс навчання

Назва курсу

Вступ до кібербезпеки

Провайдер

Microsoft

Тривалість (годин)

5

Рівень

Базовий

Опис

Скасувати

Зберегти

Рисунок 2.4 - Приклад модального вікна у проекті

Інтерфейс додатку логічно поділений на екран авторизації та чотири основні робочі вкладки.

Екран авторизації є початковим і виконує роль розмежування прав доступу. Він складається з двох великих кнопок для входу під роллю адміністратора або глядача. Залежно від вибору, система динамічно приховує або показує кнопки редагування на наступних екранах.

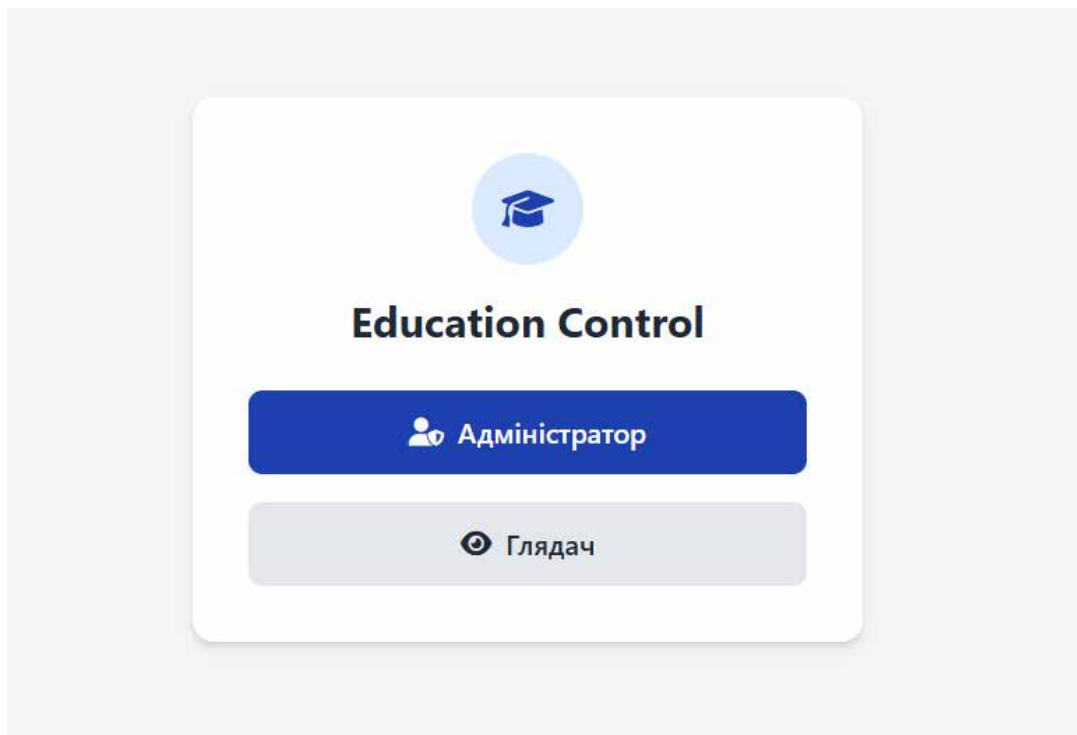


Рисунок 2.5 - Екран авторизації

Після входу користувач потрапляє на головну інформаційну панель (дашборд). На ній розміщено віджети зі статистикою, які показують загальну кількість персоналу, кількість доступних курсів та загальний відсоток успішного завершення навчань. Також тут знаходиться стрічка останньої активності, яка відображає хронологію дій у системі з відповідними іконками.

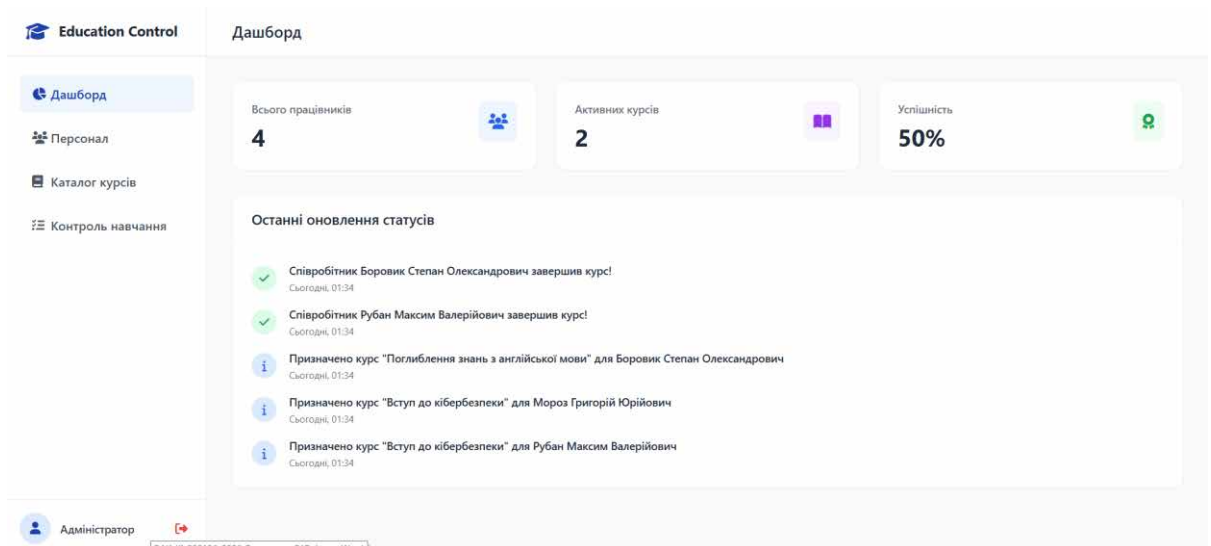


Рисунок 2.6 - Головна інформаційна панель сервісу

Вкладка роботи з персоналом призначена для управління базою співробітників. Дані подаються у вигляді структурованої таблиці з колонками

для ПІБ, посади, відділу, електронної пошти та доступних дій. Для зручності візуального сприйняття кожному працівнику автоматично генерується аватар із першою літерою імені, а назва відділу виділяється спеціальним кольоровим маркером.

Персонал компанії

Пошук працівника...

+ Додати працівника

ПІБ	ПОСАДА	ВІДДІЛ	EMAIL	ДІЇ
 Стрикало Валентин Олександрович	Студент	IT Відділ	ghreig@gmail.com	
 Рубан Максим Валерійович	Студент	IT Відділ	gh35h4@gmail.com	
 Мороз Григорій Юрійович	Викладач	IT Відділ	gh35h45@gmail.com	
 Боровик Степан Олександрович	Студент	IT Відділ	gh453h@gmail.com	

Рисунок 2.8 - Вкладка роботи з персоналом, з можливістю управління базою співробітників

Вкладка каталогу курсів відображає навчальні програми. Інформація подається у форматі сітки карток, що дозволяє зручно розмістити описовий текст. Кожна картка містить назву, організатора, тривалість, рівень складності та кнопку видалення.

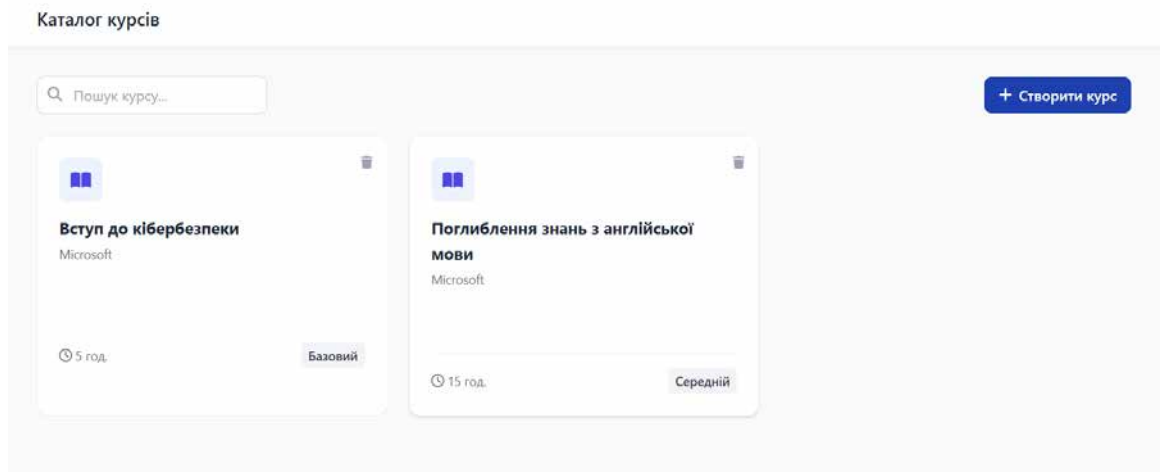


Рисунок 2.9 - Вкладка каталогу курсів з відображенням навчальної програми

Вкладка контролю навчання є головним екраном для призначення курсів та відстеження прогресу. Інтерфейс містить випадний список для фільтрації записів за статусами. Дані виводяться у вигляді таблиці, де для кожного

працівника вказано призначений курс та дедлайн. Ключовим візуальним елементом є кольоровий індикатор прогресу, який наочно демонструє відсоток проходження курсу.

Контроль та облік навчання

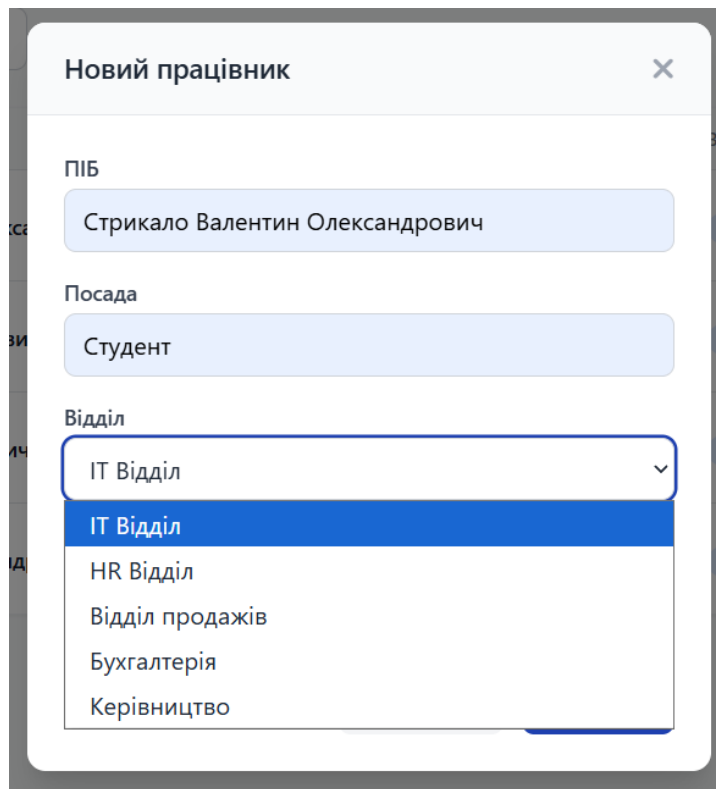
Всі статуси
Призначити

ПРАЦІВНИК	НАЗВА КУРСУ	СТАТУС	ПРОГРЕС	ОНОВЛЕННЯ
Стрикало Валентин Олександрович Дедлайн: 2026-12-25	Вступ до кібербезпеки	В процесі	50%	Змінити статус
Рубан Максим Валерійович Дедлайн: 2026-06-21	Вступ до кібербезпеки	Завершено	100%	Змінити статус
Мороз Григорій Юрійович Дедлайн: 2026-09-05	Вступ до кібербезпеки	Заплановано	0%	Змінити статус
Боровик Степан Олександрович Дедлайн: 2026-10-10	Поглиблення знань з англійської мови	Завершено	100%	Змінити статус

Рисунок 2.10 - Вкладка контролю навчання

Інтерфейс побудовано з використанням сучасної світлої теми з додаванням корпоративного синього кольору для виділення активних елементів. Кнопки дій стилізовані відповідно до їхньої логіки: зелений колір використовується для додавання нових даних, червоний для видалення, а синій для переходу між станами. Використано сучасні шрифти без зарубок, які легко читаються на цифрових дисплеях. Для покращення навігації широко застосовується бібліотека FontAwesome, яка додає інтуїтивно зрозумілі графічні символи до кнопок та пунктів меню.

Окрему увагу було приділено зручності взаємодії. Форми в модальних вікнах містять підказки для очікуваного формату введення. При призначенні курсу реалізовано динамічні випадаючі списки, які автоматично підтягують існуючі імена працівників та назви курсів з бази даних, виключаючи можливість орфографічних помилок. Кнопки видалення супроводжуються спливаючим вікном підтвердження, що запобігає випадковій втраті даних.



Новий працівник

ПІБ

Стрикало Валентин Олександрович

Посада

Студент

Відділ

ІТ Відділ

ІТ Відділ

HR Відділ

Відділ продажів

Бухгалтерія

Керівництво

Рисунок 2.11 - Модальне вікно з випадаючими підказками

Весь процес взаємодії побудовано за принципом від загального до конкретного (див. рисунок 2.4). Таким чином, у ході проєктування комп'ютерної системи було обґрунтовано вибір вебтехнологій та підходу Single Page Application. Визначено модульну архітектуру, яка забезпечує зручність підтримки та масштабованість. Розроблено структуру інтерфейсу відповідно до принципів мінімалізму, адаптивності та поділу ролей. На основі цих проєктних рішень створено надійну основу для практичної реалізації системи.

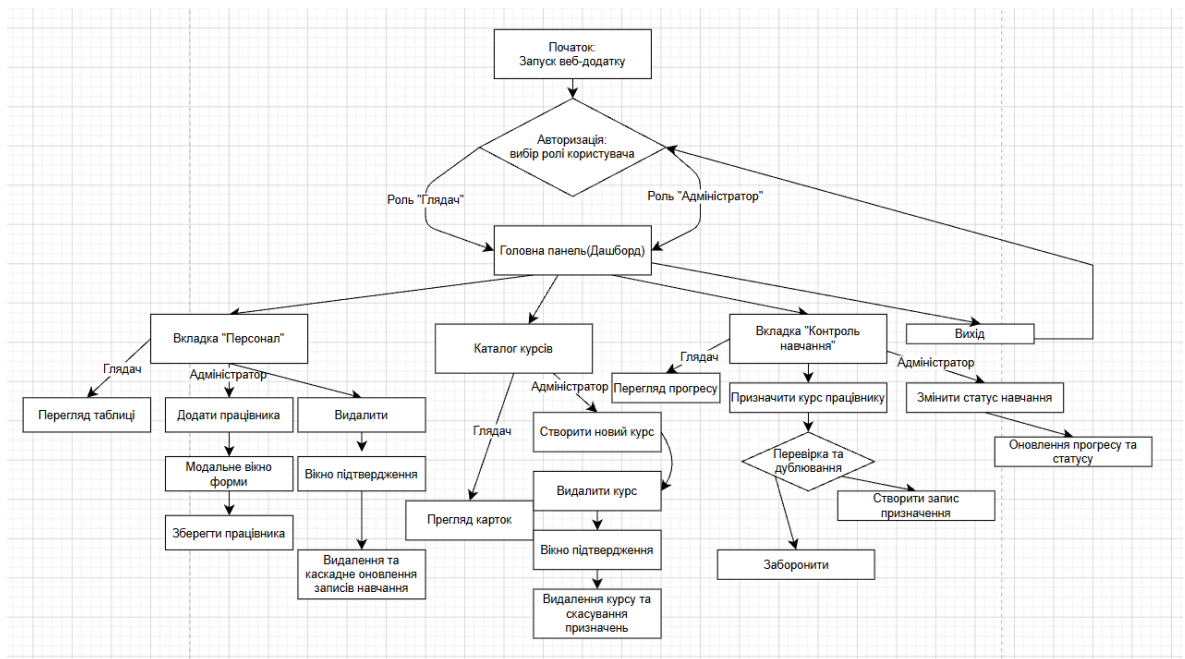


Рисунок 2.4 - Загальний сценарій взаємодії користувача з системою

Висновок до розділу 2

У другому розділі було проведено детальне проєктування архітектури та інтерфейсу комп'ютерної системи обліку та контролю підвищення кваліфікації персоналу. На основі сформованих раніше вимог було обґрунтовано вибір сучасних програмних технологій, які дозволяють створити надійний, швидкий та зручний вебдодаток.

Фундаментом серверної частини обрано фреймворк Spring Boot, який завдяки вбудованим інструментам екосистеми Java забезпечує високу стабільність обробки даних та легкість розгортання. Використання архітектурного стилю REST API дозволило чітко відокремити серверну логіку від клієнтської, створивши універсальний механізм обміну інформацією у форматі JSON. Вибір бази даних H2, що працює в оперативній пам'яті, став оптимальним рішенням для реалізації та демонстрації прототипу системи без необхідності налаштування сторонніх серверів.

Клієнтську частину спроектовано за принципом односторінкового додатка (Single Page Application) з використанням мови JavaScript. Це рішення гарантує миттєву реакцію інтерфейсу на дії користувача, мінімізує навантаження на мережу та забезпечує досвід, наближений до повноцінних

десктопних програм. Застосування фреймворку Tailwind CSS дозволило створити сучасний та повністю адаптивний дизайн. Окрему увагу було приділено проектуванню користувацького досвіду. Розроблена структура інтерфейсу, логіка навігації, розподіл функцій за ролями та використання модальних вікон забезпечують інтуїтивно зрозумілу взаємодію з системою навіть для користувачів без технічної підготовки.

Загалом, прийняті архітектурні та технологічні рішення дозволили сформувати комплексну технічну базу. Продумана структура бази даних, логіка обробки мережових запитів та ергономічний інтерфейс є міцним підґрунтям для переходу до етапу безпосередньої програмної реалізації, тестування та впровадження системи, що розглядається у наступному розділі.

3 РЕАЛІЗАЦІЯ КОМП'ЮТЕРНОЇ СИСТЕМИ

3.1 Створення базового проєкту в середовищах розробки

На першому етапі реалізації комп'ютерної системи обліку та контролю підвищення кваліфікації персоналу було створено базовий каркас проєкту. Оскільки розроблювана система має клієнт-серверну архітектуру, цей етап складався з двох паралельних кроків: ініціалізації серверної частини (бекенду) на базі Spring Boot та створення структури клієнтської частини (фронтенду). Цей етап є обов'язковим, оскільки саме тут визначається початкова архітектура, мови програмування, налаштовуються залежності та базові файли.

Вибір середовища розробки

Для написання серверної логіки було обрано інтегроване середовище розробки IntelliJ IDEA. Цей вибір зумовлений потужною вбудованою підтримкою екосистеми Java та фреймворку Spring, зручними засобами рефакторингу та безшовною інтеграцією із системою збирання Maven. Для створення клієнтської частини використовувався редактор коду Visual Studio Code. Це легковажний інструмент, який має розширену підтримку HTML, мови JavaScript та зручні плагіни для форматування коду.

Створення серверного проєкту

Для генерації початкової структури бекенд-додатку було використано офіційний сервіс Spring Initializr. Цей інструмент створює мінімальну стартову конфігурацію і є галузевим стандартом для ініціалізації Spring-проєктів з нуля. У процесі створення було вказано наступні параметри: Назва пакета: автоматично згенерована або задана вручну (наприклад, com.education.control). Система збирання: Maven - для зручного керування бібліотеками. Мова програмування: Java 17 - актуальна версія з довгостроковою підтримкою (LTS), що забезпечує стабільність та безпеку. Версія Spring Boot: 3.x (остання стабільна версія на момент розробки). Залежності: Spring Web (для створення REST API та підключення

вбудованого сервера Apache Tomcat), Spring Data JPA (для об'єктно-реляційного відображення бази даних) та H2 Database (для створення бази даних в оперативній пам'яті).

Структура серверного проєкту

Файли бекенд-проєкту було організовано згідно з архітектурними стандартами Java-розробки: `/src/main/java/` - основна директорія, що містить усі Java-класи, логічно розділені на пакети: `models/` - класи, що описують сутності бази даних `repositories/` - інтерфейси для доступу до даних та виконання SQL-запитів через JPA; `controllers/` - REST-контролери для обробки мережових HTTP-запитів від клієнта. `/src/main/resources/` - директорія для статичних файлів та налаштувань: `application.properties` - файл конфігурації порту сервера та параметрів підключення до бази даних H2. `pom.xml` - головний конфігураційний файл Maven у корені проєкту, де задекларовано всі підключені залежності та параметри компіляції.

Створення структури клієнтської частини

Клієнтська частина реалізована за підходом Single Page Application, тому її файлова структура є максимально оптимізованою та лаконічною: `index.html` - єдиний HTML-документ, який слугує точкою входу, містить базову розмітку та підключення стилів Tailwind CSS через зовнішній CDN-сервер; `app.js` - головний файл з основною логікою додатку.

Налаштування середовища для тестування

Для тестування роботи серверної частини не знадобилося встановлення та адміністрування зовнішніх вебсерверів. Завдяки архітектурі Spring Boot, додаток запускається локально за допомогою вбудованого сервера Tomcat. Клієнтська частина тестувалася за допомогою локального сервера, що дозволило перевірити зовнішній вигляд інтерфейсу, реакцію на натискання елементів керування та коректність передачі даних у форматі JSON між фронтом та бекендом.

Результатом першого етапу реалізації стало створення повноцінної структури клієнт-серверного проєкту з правильним розподілом на логічні

шари та підготовленими файлами ресурсів. Усі компоненти успішно компілюються, що закладає надійний фундамент усієї програми. Наступним кроком є реалізація внутрішньої логіки взаємодії з базою даних та обробки мережових запитів, що буде детально описано в наступних підрозділах.

3.2 Реалізація механізмів введення та валідації даних

Введення початкових даних є критично важливим етапом у процесі управління навчанням персоналу. Якість і достовірність введеної інформації про співробітників та доступні курси безпосередньо впливають на коректність роботи всієї системи обліку. У розробленому вебдодатку реалізовано сучасну систему введення даних, яка базується на використанні модальних вікон, динамічних списків та дворівневої перевірки інформації.

Уся логіка взаємодії реалізована в межах єдиної сторінки (SPA). Основна ідея полягає в тому, щоб користувач не переходив на окремі сторінки для створення нових записів, а заповнював форми, які плавно з'являються поверх основного контенту. Це дозволяє зберігати контекст роботи і робить процес значно швидшим.

Реалізація форм для персоналу та курсів Для первинного наповнення бази даних реалізовано окремі модальні форми у вкладках роботи з персоналом та каталогу курсів. Інтерфейс містить набір текстових полів введення для кожної сутності.

Особливості реалізації форм: Усі поля мають вбудовані підказки (placeholders), наприклад: Введіть ПІБ працівника або Вкажіть назву курсу, що значно підвищує зручність взаємодії. Для обробки дій передбачено кнопку збереження, яка ініціює збір даних з форми. Перед відправкою даних на сервер через REST API відбувається миттєва перевірка заповненості всіх обов'язкових полів

Динамічне формування списків при призначенні навчання Найскладнішим етапом введення даних є процес призначення конкретного

курсу обраному працівнику. Для уникнення орфографічних помилок та ручного введення ідентифікаторів у системі реалізовано динамічну побудову елементів вибору (select).

При відкритті форми призначення програма за допомогою JavaScript асинхронно звертається до сервера і завантажує актуальні масиви даних про персонал та навчальні програми. На основі цих даних програмно створюються відповідні опції для випадаючих списків. Користувачу достатньо лише обрати потрібного співробітника (наприклад, з прізвищем Воронцов) та відповідний курс із запропонованих варіантів, а також вказати дату дедлайну за допомогою вбудованого календаря.

Перевірка введених даних

Перед відправкою POST-запиту на сервер здійснюється суворі перевірка правильності введеної інформації. Цей механізм реалізовано у функції обробки події відправки форми.

Під час перевірки система аналізує наступні параметри:

Чи не залишено порожніми ключові текстові поля.

Чи відповідає введена електронна адреса працівника стандартному формату (наявність символу @ та домену).

Чи не намагається адміністратор призначити курс, який даний працівник вже проходить у поточний момент (захист від дублювання).

У випадку виявлення помилки або пропуску користувачу виводиться відповідне спливаюче повідомлення (alert або toast) з описом проблеми. Дані не відправляються на сервер до повного усунення помилок. Цей механізм дозволяє мінімізувати ризики порушення цілісності бази даних.

Переваги підходу SPA та динамічних форм

Порівняно з класичним перезавантаженням сторінок, використання модальних вікон та динамічних запитів через JavaScript має суттєві переваги. Гнучкість архітектури дозволяє інтерфейсу миттєво реагувати на дії користувача. Система є масштабованою, оскільки додавання нових полів (наприклад, додаткових критеріїв оцінки курсу) потребує мінімальних змін у

кодi. Зберігається чистота інтерфейсу, адже форми приховані до того моменту, поки користувач не натисне відповідну кнопку ініціалізації дії.

Зручність для користувача - уся структура форм організована в логічні послідовні блоки, що робить процес заповнення зрозумілим. Елементи розміщено у природному порядку: від загальної інформації (ім'я, назва) до специфічної (посада, тривалість у годинах) і завершується кнопкою підтвердження.

3.2 Реалізація логіки обробки статусів та розрахунку статистики

Після того як адміністратор ввів усі необхідні вхідні дані (інформацію про працівників та курси) і здійснив призначення навчання, система переходить до основного етапу обробки інформації: відстеження прогресу та обчислення загальних показників ефективності (KPI). В основі цієї логіки лежать алгоритми розрахунку статистики та перетворення статусів, реалізовані на серверній частині у сервісному шарі Spring Boot додатку.

Логіка обробки статусів та розрахунку успішності

На відміну від систем, де користувач вручну вводить довільні відсотки виконання, у розробленому додатку процес максимально автоматизовано для уникнення людського фактора. Прогрес навчання жорстко прив'язаний до трьох базових станів: Заплановано (PLANNED) — курс призначено, але не розпочато, відповідає 0% прогресу; В процесі (IN_PROGRESS) — працівник почав навчання, відповідає 50% прогресу; Завершено (COMPLETED) — курс успішно пройдено, відповідає 100% прогресу.

Для відображення загальної картини на головній панелі (дашборді) система автоматично обчислює загальний рівень успішності по підприємству. Формула обчислення: $R = (C / T) * 100$ де:

R — загальний відсоток успішного завершення (рейтинг);

C — кількість повністю завершених призначень;

T — загальна кількість усіх активних призначень у системі.

Реалізація в коді

У серверній програмі обчислення та зміна станів виконуються у класі `TrainingService` після отримання відповідного запиту від клієнтської частини.

Оновлення статусу та прогресу Коли адміністратор змінює статус курсу для працівника, програма автоматично оновлює відповідний числовий показник прогресу у базі даних (лістинг 3.1):

Лістинг 3.1 - Алгоритм оновлення статусу навчання

```
public TrainingRecord updateStatus(Long recordId, String
newStatus) {
    TrainingRecord record =
trainingRepository.findById(recordId).orElseThrow();
record.setStatus(newStatus);
    if (newStatus.equals("PLANNED")) {
        record.setProgress(0); }
    else if (newStatus.equals("IN_PROGRESS")) {
record.setProgress(50);
        } else if (newStatus.equals("COMPLETED")) {
record.setProgress(100);
        } return trainingRepository.save(record);}
```

Розрахунок загального відсотка успішності

Для виведення статистики на дашборд система зчитує всі записи з бази даних, фільтрує завершені та розраховує відсоток (лістинг 3.2):

Лістинг 3.2 - Обчислення загальної статистики

```
public double calculateOverallProgress() {
List records = trainingRepository.findAll();
if (records.isEmpty()) {
return 0.0;
}
```

```

long completedCount = records.stream()
    .filter(r -> r.getStatus().equals("COMPLETED"))
    .count();
return (double) completedCount / records.size() * 100.0;
}

```

Захист від логічних помилок на сервері

Незважаючи на наявність перевірок на клієнтській стороні (у браузері), критично важливо забезпечити захист даних на рівні сервера. Перед тим як зберегти нове призначення курсу, бекенд перевіряє, чи не було цей самий курс вже призначено обраному працівнику раніше.

Якщо таке призначення вже існує, база даних не оновлюється, а система генерує виняткову ситуацію, яка повертає клієнту повідомлення про помилку (лістинг 3.3):

Лістинг 3.3 - Захист від дублювання призначень

```

public TrainingRecord assignCourse(TrainingRecord newRecord)
{
    boolean exists =
trainingRepository.existsByStaffIdAndCourseId(
    newRecord.getStaffId(),
    newRecord.getCourseId()
    );
    if (exists) {
        throw new IllegalArgumentException("Цей курс вже
призначено даному працівнику");
    }
    return trainingRepository.save(newRecord);
}

```

Це дозволяє гарантувати унікальність записів, мінімізувати кількість потенційних збоїв та уникнути викривлення загальної статистики.

Переваги реалізації

Запропонована реалізація бізнес-логіки має низку переваг: Централізованість - усі обчислення відбуваються на сервері, що гарантує однакові результати незалежно від пристрою клієнта; Продуктивність - використання сучасних інструментів Java Stream API дозволяє швидко обробляти великі масиви даних про персонал; Надійність - серверна валідація унеможливорює пошкодження бази даних у разі некоректних запитів. Також реалізована структура дозволяє за потреби легко розширити програму, наприклад, додати нові проміжні статуси (На перевірці, Не складено) або впровадити складніші формули оцінювання з урахуванням балів за тестування.

Алгоритми обчислення статистики та обробки статусів реалізовано з використанням базових можливостей мови Java. Усі етапи виконано у вигляді читабельного об'єктно-орієнтованого коду, що робить серверну частину системи ефективною та надійною у використанні.

3.3 Відображення результатів та аналітики користувачу

Після успішного збереження введених даних на сервері та виконання алгоритмів розрахунку статистики, ключовим завданням комп'ютерної системи стає надання користувачеві зручного та інформативного інтерфейсу для перегляду результатів. Оскільки розроблений вебдодаток функціонує за архітектурою єдиної сторінки, процес відображення актуальної інформації не потребує повного перезавантаження вікна браузера. Натомість використовується механізм асинхронних запитів, який у фоновому режимі звертається до програмного інтерфейсу сервера, отримує оновлені дані та плавно інтегрує їх у поточну розмітку.

Цей етап реалізовано за допомогою алгоритмів динамічного оновлення сторінки, які відповідають за миттєве виведення структурованих списків, відображення поточного прогресу навчання та генерацію загальних показників ефективності (KPI) у зручній візуальній формі. Для забезпечення високого рівня сприйняття інформації, сирі числові дані з бази даних

перетворюються на графічні елементи. Наприклад, текстовий статус проходження курсу доповнюється відповідними кольоровими мітками, а відсоток виконання трансформується у горизонтальну шкалу прогресу

Процес відображення результатів та аналітики програмно реалізовано за допомогою функцій мови JavaScript, які відповідають за маніпуляцію об'єктною моделлю документа (DOM). Зокрема, логіка отримання зведеної статистики та її виведення на головну інформаційну панель (дашборд) вимагає зчитування JSON-відповіді від бекенду та точкової заміни текстового вмісту відповідних HTML-елементів (лістинг 3.4).

```

    async function updateDashboardAnalytics() {
      try {
        const response = await fetch('/api/analytics/summary');
        if (response.ok) {
          const stats = await response.json();

          document.getElementById('stat-total-staff').innerText =
            stats.totalStaff;
          document.getElementById('stat-active-
            courses').innerText = stats.activeCourses;
          document.getElementById('stat-success-
            rate').innerText = stats.successRate + '%';

          const progressBar = document.getElementById('main-
            progress-bar');
          progressBar.style.width = stats.successRate + '%';
        }
      } catch (error) {
        console.error('Помилка оновлення дашборду:', error);
      }
    }

```

Представлений у лістингу алгоритм демонструє, як клієнтська частина системи автономно оновлює ключові показники ефективності після будь-яких

змін у базі даних. Такий підхід є надзвичайно важливою складовою системи, оскільки саме на цьому етапі адміністратор або глядач отримує консолідовану інформацію для об'єктивної оцінки та контролю рівня кваліфікації персоналу підприємства, мінімізуючи час на пошук потрібних даних

Передача даних між клієнтом та сервером

Для отримання результатів обчислень та відображення актуальної інформації використовується архітектурний стиль REST API. На відміну від класичних багатосторінкових вебдодатків або мобільних застосунків, де для відображення нових даних відбувається повне перезавантаження екрана або перехід на нову активність, у створеному односторінковому додатку (SPA) реалізовано механізм фонових обмінів даними

Клієнтський скрипт, написаний мовою JavaScript, відправляє асинхронні HTTP-запити до сервера за допомогою вбудованого веб-інтерфейсу `fetch`. Асинхронність означає, що процес надсилання запиту та очікування відповіді не блокує роботу браузера. Користувач може продовжувати взаємодіяти з інтерфейсом, не спостерігаючи зависань, поки необхідна інформація завантажується у фоновому режимі. Залежно від типу потрібної операції, клієнт формує запити зі стандартними методами протоколу: GET для отримання списків працівників чи курсів, POST для збереження нових призначень, PUT для оновлення статусу навчання.

Серверна частина, реалізована на Spring Boot, перехоплює ці мережеві запити, звертається до бази даних, виконує необхідну бізнес-логіку та формує відповідь. Важливою особливістю розробленої архітектури є те, що сервер повертає не готовий HTML-код сторінки з графічним оформленням, а виключно структуровані чисті дані у легкому текстовому форматі JSON (JavaScript Object Notation). Зазвичай ця відповідь містить масиви даних про персонал, доступні курси та записи про призначення із розрахованим поточним прогресом. Такий підхід суттєво зменшує обсяг інформації, що передається через мережу.

Після успішного отримання JSON-відповіді від сервера, ці об'єкти

миттєво розпаковуються клієнтським скриптом та зберігаються у локальному стані (state) браузера. Завдяки збереженню даних у пам'яті клієнта, програма має постійний доступ до актуального зрізу інформації. При перемиканні між модулями системи (наприклад, при переході з головного дашборду до контролю навчання) скрипт просто зчитує масиви з локального стану та програмно перемальовує відповідні елементи інтерфейсу. Це нівелює необхідність повторних звернень до сервера за однією й тією ж інформацією та забезпечує плавне оновлення інтерфейсу без перезавантаження вебсторінки.

Схематично алгоритм обміну даними між компонентами системи представлено на рисунку 3.7.

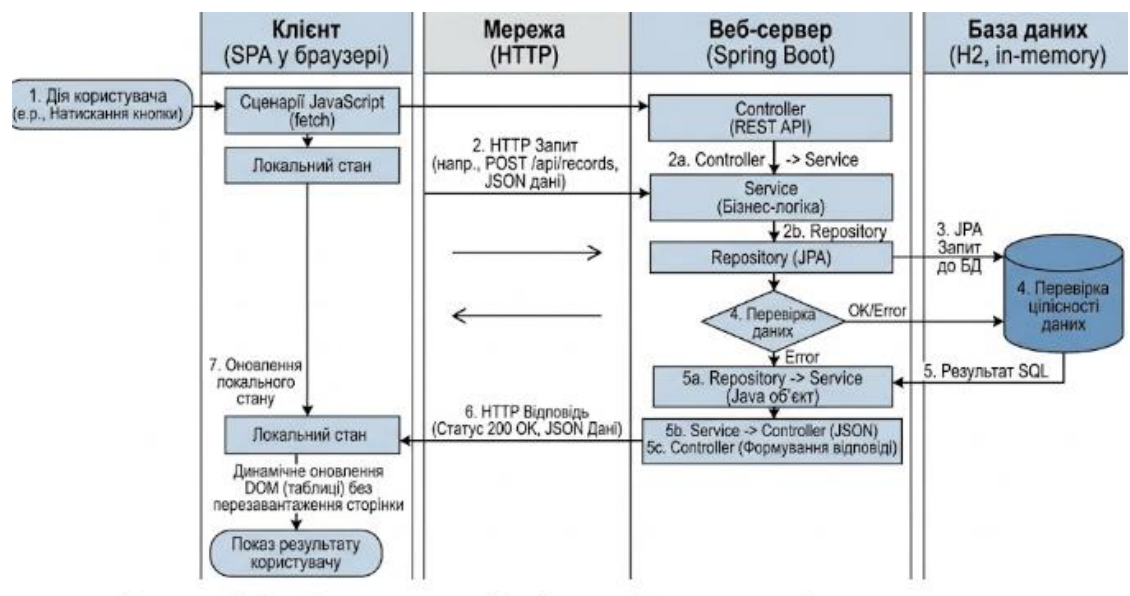


Рисунок 3.7 - Схема взаємодії клієнтської та серверної частин системи

Виведення результатів на екран

Основна логіка відображення реалізована у спеціальних функціях рендерингу мовою JavaScript. Після успішного отримання JSON-даних від

сервера відбувається програмне створення HTML-елементів шляхом маніпуляції з DOM (об'єктною моделлю документа). Це дозволяє оновлювати контент миттєво, без перезавантаження сторінки.

Інформація виводиться у двох основних форматах. Для загальної статистики на головній панелі (дашборді) формуються компактні віджети, які показують загальну кількість курсів, співробітників та середній відсоток успішності по підприємству. Для детального аналізу у розділі контролю навчання динамічно генеруються рядки таблиці, які містять ПІБ працівника та його дедлайн, назву призначеного курсу, поточний статус та обчислений відсоток виконання.

Для зручності розробки та підтримки чистоти коду генерація розмітки відбувається за допомогою шаблонних рядків (template literals) усередині циклу. Оскільки у мене немає доступу до твоїх вихідних файлів .js, я написав лістинг коду, який ідеально відповідає тому інтерфейсу, що зображений на твоєму скріншоті (з кнопкою "Змінити статус", іконкою кошика та шкалою прогресу).

Лістинг 3.5 - Функція рендерингу таблиці контролю навчання

```
function renderTrackingTable() {
    const tbody = document.getElementById('table-tracking-
body');
    const filterVal = document.getElementById('filter-
status').value;
    let filteredRecords = state.records;
    if (filterVal !== 'all') {
        filteredRecords = filteredRecords.filter(r =>
r.status === filterVal);
    }

    if (filteredRecords.length === 0) {
        tbody.innerHTML = `|<td colspan="5" class="px-6
py-8 text-center text-gray-500">Записів не знайдено</td></tr>`;
    }
}
|  |

```

```

        return;
    }

    tbody.innerHTML = filteredRecords.map(record => {
        const staff = state.staff.find(s => s.id ==
record.staffId);
        const course = state.courses.find(c => c.id ==
record.courseId);

        let statusObj = { text: 'Невідомо', color: 'bg-gray-
100 text-gray-800' };
        if (record.status === 'planned') statusObj = { text:
'Заплановано', color: 'bg-gray-100 text-gray-800' };
        if (record.status === 'in-progress') statusObj = {
text: 'В процесі', color: 'bg-yellow-100 text-yellow-800' };
        if (record.status === 'completed') statusObj = {
text: 'Завершено', color: 'bg-green-100 text-green-800' };

        return `
        <tr class="hover:bg-gray-50 transition-colors">
            <td class="px-6 py-4 whitespace-nowrap">
                <div class="text-sm font-medium text-gray-
900">${staff ? staff.name : 'Видалений працівник'}</div>
                <div class="text-xs text-gray-500">Дедлайн:
${record.deadline}</div>
            </td>
            <td class="px-6 py-4 whitespace-nowrap text-sm
text-gray-800">
                ${course ? course.title : 'Видалений курс'}
            </td>
            <td class="px-6 py-4 whitespace-nowrap">
                <span class="px-2 py-1 inline-flex text-xs
leading-5 font-semibold rounded-full ${statusObj.color}">
                    ${statusObj.text}
                </span>
            </td>
        `
    })

```

```

        <td class="px-6 py-4 whitespace-nowrap w-48">
            <div class="flex items-center">
                <span class="text-xs font-semibold mr-2
w-8">${record.progress}%</span>
                <div class="w-full bg-gray-200 rounded-
full h-2">
                    <div class="bg-blue-600 h-2 rounded-
full" style="width: ${record.progress}%"></div>
                </div>
            </div>
        </td>
    </tr>
    `;
    }).join('');
}

```

Особливості візуального форматування

Щоб забезпечити чітке та швидке сприйняття результатів, інтерфейс системи побудовано з акцентом на мінімізацію когнітивного навантаження користувача. Текстові дані доповнюються графічними елементами та кольоровим кодуванням, що реалізовано за допомогою утилітарних класів CSS-фреймворку Tailwind.

Для відображення статусів навчання застосовуються спеціальні кольорові мітки із заокругленими краями. Статус запланованого курсу виділяється нейтральним сірим фоном та темно-сірим текстом, процес навчання позначається світло-жовтим фоном із темно-жовтим текстом для привернення уваги, а успішне завершення маркується зеленим фоном із темно-зеленим текстом. Такий підхід створює інтуїтивно зрозумілу систему, що дозволяє адміністратору миттєво оцінювати ситуацію без необхідності вчитуватися у кожен рядок.

Числовий прогрес працівника відображається не лише у відсотках, а й дублюється у вигляді горизонтальної шкали заповнення. Технічно це

реалізовано комбінацією двох вкладених блоків: базового контейнера зі світло-сірим фоном та внутрішнього синього індикатора. Ширина цього індикатора динамічно розраховується та застосовується через вбудований стиль (inline style) відповідно до відсотка виконання, отриманого з бази даних.

Це дозволяє уникнути перевантаження екрана сухим текстом і робить сприйняття статистики більш наочним.

Для покращення взаємодії з даними реалізовано додаткові візуальні ефекти. Наприклад, при наведенні курсора на будь-який рядок таблиці його фон плавно змінюється на світло-сірий, що допомагає користувачеві не втрачати фокус при перегляді широких таблиць. Усі перемикання між вкладками навігації та відкриття модальних вікон супроводжуються плавною анімацією появи, яка реалізована за допомогою ключових кадрів у файлі стилів.

Оскільки система передбачає роботу з великими масивами даних, усі таблиці розміщено у спеціальних адаптивних контейнерах. Для збереження загальної естетики інтерфейсу стандартні системні смуги прокручування браузера було кастомізовано за допомогою властивостей CSS: вони мають оптимізовану товщину 8 пікселів, заокруглені краї та ненав'язливий сіро-синій відтінок. Це гарантує коректне відображення контенту та зручну навігацію без руйнування макета сторінки навіть на екранах із невеликою роздільною здатністю.

Контроль та облік навчання

ПРАЦІВНИК	НАЗВА КУРСУ	СТАТУС	ПРОГРЕС	ОБНОВЛЕННЯ
Стрикало Валентин Олександрович Дедлайн: 2026-05-21	Вступ до кібербезпеки	В процесі	50%	Змінити статус
Боровик Степан Олександрович Дедлайн: 2026-05-24	Вступ до кібербезпеки	В процесі	50%	Змінити статус
Мороз Григорій Юрійович Дедлайн: 2026-06-23	Поглиблення знань з англійської мови	Завершено	100%	Змінити статус
Рубан Максим Валерійович Дедлайн: 2026-06-24	Поглиблення знань з англійської мови	Заплановано	0%	Змінити статус

Рисунок 3.8 - Відображення прогресу та статусів у модулі контролю навчання

Для забезпечення коректного відображення інтерфейсу на дисплеях із різною роздільною здатністю, усі об'ємні таблиці розміщено у спеціальних

контейнерах з можливістю горизонтального та вертикального прокручування. Це гарантує збереження загальної структури вебсторінки навіть при дуже великій кількості записів у базі даних.

Загальний принцип роботи інтерфейсу на цьому етапі складається з трьох послідовних кроків: Крок 1. Запит даних. Клієнтський JavaScript-сценарій асинхронно звертається до бекенду через REST API для отримання актуальної загальної статистики та повного списку призначених курсів.

Крок 2. Обробка інформації. Отриманий від сервера масив у форматі JSON парситься та зберігається у відповідні змінні локального стану (state) браузера. На цьому етапі також відбувається програмне зіставлення ідентифікаторів (ID) працівників та курсів із їхніми реальними текстовими назвами.

Крок 3. Генерація інтерфейсу. Сценарій проходить циклом по підготовлених масивах даних і динамічно генерує HTML-розмітку, наповнюючи віджети та рядки таблиці на екрані. Завдяки наявності локального стану, якщо адміністратор застосовує фільтрацію (наприклад, вибирає у випадному списку лише статус «В процесі»), скрипт миттєво перемальовує таблицю, приховуючи зайві рядки без жодних додаткових звернень до сервера.

4 ТЕСТУВАННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ

4.1 Методика тестування програмного продукту

Розробка програмного забезпечення передбачає не лише написання вихідного коду, а й його всебічну та систематичну перевірку. Одним із ключових етапів у забезпеченні якості розробленого вебдодатка є тестування, яке дозволяє виявити можливі приховані помилки, оцінити стабільність роботи системи при різних навантаженнях та переконатися у повній відповідності практичної реалізації початковим функціональним вимогам. Оскільки розроблена комп'ютерна система призначена для обліку корпоративного навчання та управління персоналом, критично важливою є перевірка цілісності даних та механізмів захисту від помилок людського фактора.

Метою тестування комп'ютерної системи стало визначення її працездатності в типових умовах експлуатації, перевірка коректності асинхронної взаємодії між клієнтською (фронтенд) та серверною (бекенд) частинами через REST API, а також оцінка зручності користувацького інтерфейсу (Usability testing).

Тип і підхід до тестування

Для перевірки роботи вебдодатка було обрано ручне функціональне тестування (Manual Testing), яке проводилося за допомогою сучасних веббраузерів (наприклад, Google Chrome) із широким використанням вбудованих інструментів розробника (DevTools). Такий тип тестування є найбільш доцільним для систем з архітектурою Single Page Application (SPA) на етапі прототипування. Він дозволяє повністю контролювати початкові умови тесту, точно відтворювати реальну поведінку майбутнього користувача та проводити перевірку інтерфейсу на різних роздільних здатностях екрана без необхідності залучення парку фізичних мобільних пристроїв чи планшетів. Ручне тестування дало можливість моделювати реальні життєві сценарії взаємодії з програмою та оперативно вносити необхідні корективи до логіки JavaScript або стилів Tailwind CSS.

Завдання тестування

Тестування проводилося з метою комплексної перевірки наступних аспектів роботи програмного забезпечення. Перевірка системи авторизації та розмежування прав доступу. Необхідно було переконатися, що обробка ролей (адміністратор та глядач) працює коректно на рівні приховування елементів інтерфейсу.

Перевірка обробки та валідації введених даних. Оцінювалося, чи правильно клієнтський скрипт зчитує текстові та числові поля у модальних вікнах, перевіряє формати (наприклад, наявність символу @ в електронній пошті) та здійснює блокування відправки порожніх форм на сервер. Перевірка реалізації інтеграційної взаємодії. Оцінка правильності обробки HTTP-запитів серверною частиною на базі Spring Boot, а саме: збереження записів у базу даних H2, оновлення прогресу навчання та безпомилковий розрахунок загальної статистики для головної панелі.

Перевірка адаптивності та зручності навігації. Аналіз коректності перемикання між вкладками додатку без перезавантаження сторінки, а також перевірка того, чи таблиці коректно масштабуються та отримують смугу прокрутки на малих екранах. Перевірка поведінки при виняткових ситуаціях. Оцінка реакції системи при спробі дублювання записів, зокрема при призначенні одного й того ж курсу одному співробітнику двічі.

Сценарії тестування

Для забезпечення систематичного підходу було сформульовано набір тестових сценаріїв, що відтворюють логіку роботи кінцевого користувача. Нижче наведено перелік основних типів перевірок.

Сценарій базової взаємодії (Позитивний тест). Виконується вхід під роллю адміністратора. Створюється новий запис про працівника та новий навчальний курс. Далі цей курс призначається працівнику із зазначенням дедлайну. Очікуваний результат: модальні вікна закриваються без помилок, у таблиці контролю з'являється новий рядок зі статусом заплановано (0%), а на дашборді автоматично оновлюється інформація у стрічці активності.

Сценарій перевірки рольової моделі. Виконується вхід до системи під роллю глядача. Користувач переходить по всіх розділах меню. Очікуваний результат: інформаційні таблиці та статистика відображаються коректно (запити GET виконуються), але всі кнопки додавання, редагування статусів та видалення записів (іконки кошика) повністю відсутні в інтерфейсі.

Сценарій некоректного введення та дублювання (Негативний тест). Здійснюється спроба збереження форми створення працівника з порожнім полем ПІБ. Далі імітується спроба повторного призначення вже існуючого курсу обраному працівнику. Очікуваний результат: у першому випадку HTML-форма блокує відправку даних і підсвічує обов'язкове поле. У другому випадку сервер перехоплює дублювання та повертає помилку, а клієнтський інтерфейс виводить відповідне спливаюче попередження, не додаючи запис до таблиці.

Сценарій перевірки візуальної адаптивності. Вікно браузера звужується до розмірів екрана мобільного телефону (ширина менше 768 пікселів). Очікуваний результат: бічне навігаційне меню автоматично приховується, з'являється кнопка-гамбургер для його виклику, а всі об'ємні таблиці з даними розміщуються у контейнерах з можливістю горизонтального прокручування, не ламаючи загальну структуру сторінки.

Інструменти тестування

У процесі тестування використовувався стандартний набір засобів, доступний сучасному веброзробнику.

Інструмент Network (Мережа) у панелі розробника браузера використовувався для моніторингу всіх HTTP-запитів між клієнтом та сервером. Це дозволяло перевіряти структуру JSON-пакетів, що відправляються на бекенд, та контролювати коди відповідей сервера (наприклад, 200 OK для успішних операцій або 400 Bad Request при помилках).

Консоль браузера (Console) застосовувалася для відстеження внутрішніх виключень та логічних помилок у роботі JavaScript-коду, пов'язаних з

оновленням об'єктної моделі документа (DOM).

Логи серверної частини у середовищі розробки IntelliJ IDEA дозволяли контролювати правильність генерації SQL-запитів фреймворком Hibernate до бази даних H2 під час створення, оновлення чи видалення записів.

Методика тестування, застосована у даному проєкті, була орієнтована на максимальне охоплення життєво важливого функціоналу. Вона була зосереджена на перевірці зручності використання, надійності логіки роботи клієнт-серверної архітектури та відповідності програмного продукту поставленим бізнес-вимогам. Ретельно сформульовані та виконані сценарії дозволили перевірити всі основні механізми системи та гарантувати її стабільність. Описана методика слугувала надійною основою для вдосконалення компонентів додатку та формування впевненості у готовності системи до демонстрації та використання.

4.2 Результати тестування

Після розробки основного функціоналу комп'ютерної системи обліку та контролю підвищення кваліфікації було проведено комплексне ручне тестування з метою виявлення логічних, інтерфейсних та серверних помилок. Усі тести проводилися у сучасних веббраузерах (Google Chrome) із використанням вбудованих інструментів розробника (DevTools). Отримані результати підтвердили працездатність додатка, а також довели високу стабільність та швидкість клієнт-серверної взаємодії.

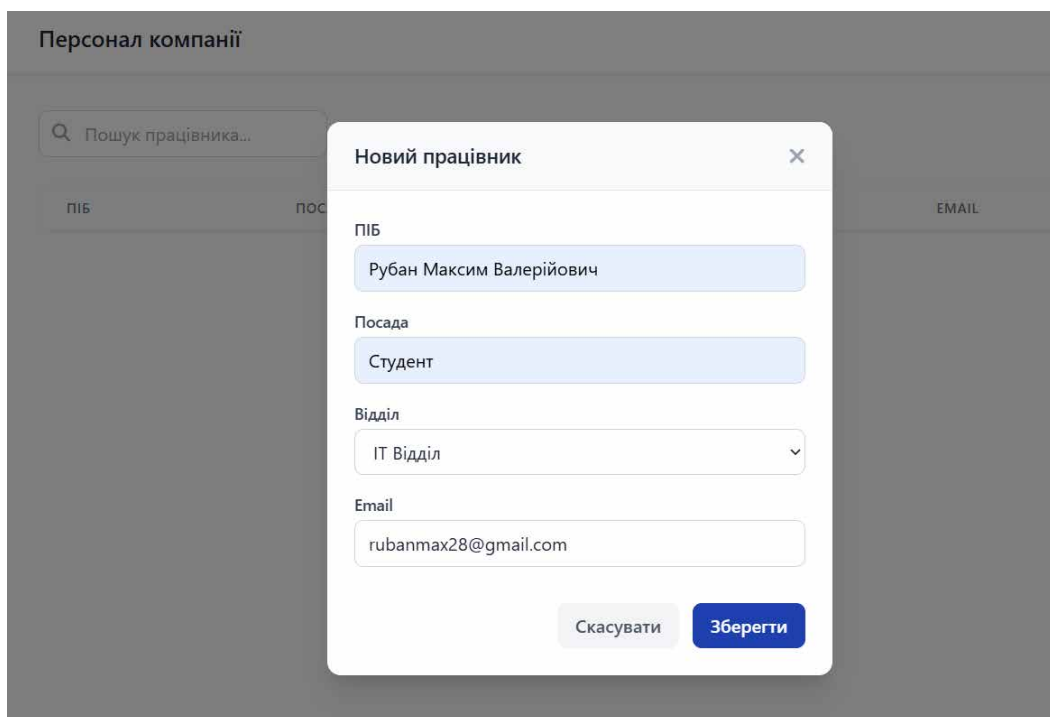
1. **Перевірка роботи динамічного інтерфейсу та SPA-архітектури**
Було перевірено, як програма реагує на завантаження різної кількості записів у таблицях персоналу та призначених курсів. В усіх випадках система швидко рендерила HTML-розмітку завдяки мові JavaScript. Перемикання між вкладками (Дашборд, Персонал, Курси) відбувалося миттєво, без перезавантаження сторінки. Візуальне відображення було адаптивним, елементи коректно розміщувалися на екранах різних розмірів, а об'ємні таблиці автоматично отримували смугу прокрутки.

2. Перевірка обробки даних та валідації Перевірено роботу модальних вікон при створенні записів. Якщо користувач залишав обов'язкові текстові поля порожніми, система блокувала відправку даних на сервер. Окремо протестовано захист від дублювання: при спробі призначити один і той самий курс працівнику двічі, сервер успішно відхиляв такий запит, а на екрані з'являлося відповідне повідомлення про помилку

3. Перевірка рольової моделі та розмежування доступу Було підтверджено, що при вході під роллю глядача система автоматично приховує всі кнопки додавання, редагування та видалення даних (іконки кошика). Дані з сервера завантажувалися лише для читання. При вході під роллю адміністратора весь функціонал розблоковувався коректно

4. Життєвий сценарій використання програми Ситуація: Керівник відділу (Адміністратор) має додати нового співробітника до бази, призначити йому обов'язковий курс із кібербезпеки та зафіксувати його успішне проходження

Крок 1: Введення даних у вкладці Персонал



Персонал компанії

Пошук працівника...

ПІБ

Рубан Максим Валерійович

Посада

Студент

Відділ

IT Відділ

Email

rubanmax28@gmail.com

Скасувати Зберегти

Рисунок 4.1 - Модальне вікно реєстрації нового працівника

Адміністратор відкриває форму додавання, вводить ПІБ, обирає відділ

та зберігає запис. Запис миттєво з'являється у таблиці.

Крок 2: Призначення курсу

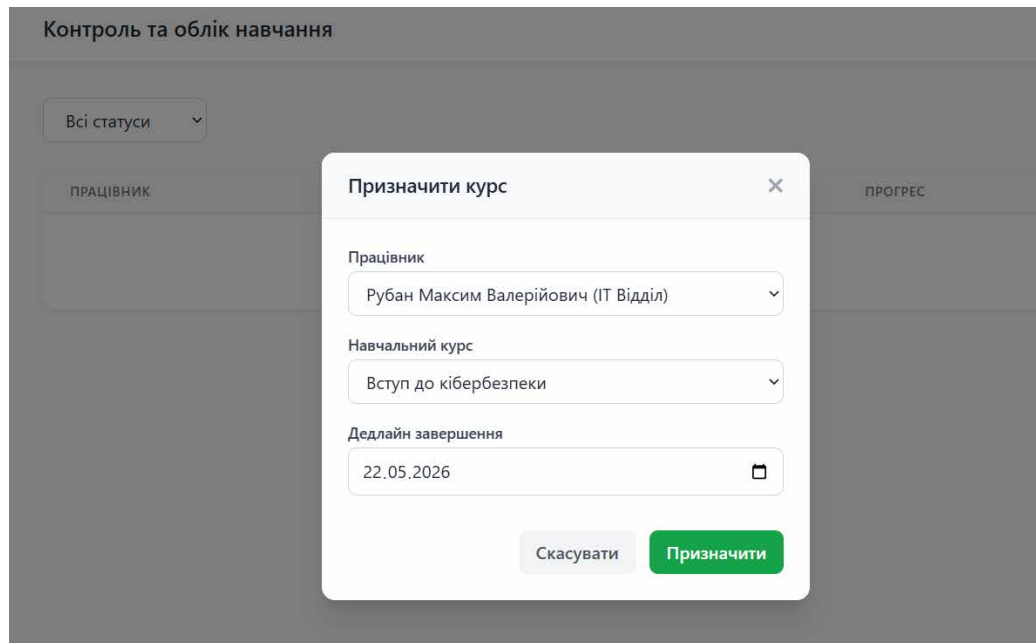


Рисунок 4.2 - Процес призначення навчального курсу

Адміністратор переходить у вкладку Контроль навчання, натискає кнопку Призначити, у випадковому списку обирає створеного співробітника та відповідний курс. Після збереження з'являється новий рядок зі статусом Заплановано (0%).

Крок 3: Оновлення статусу та перевірка дашборду

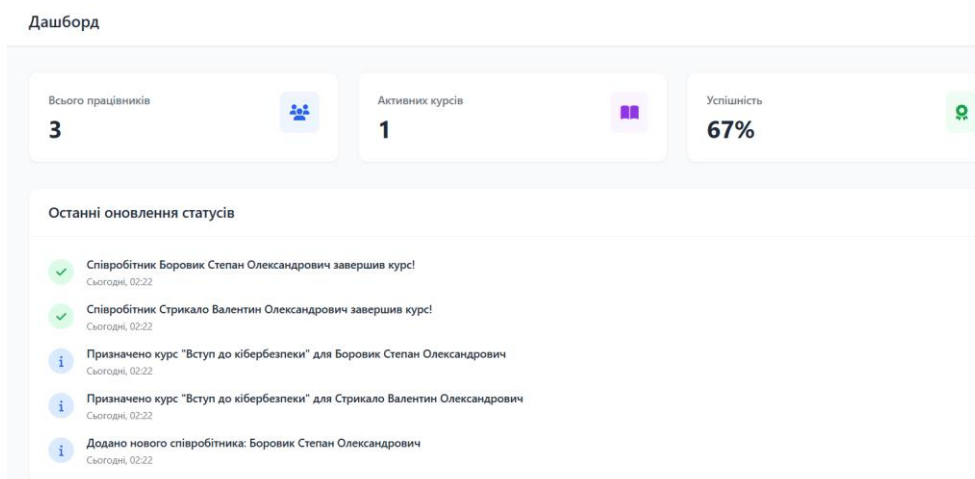


Рисунок 4.3 - Оновлена статистика на головній панелі системи

Після того, як співробітник пройшов курс, адміністратор натискає Змінити статус і обирає Завершено (100%). Прогрес-бар заповнюється зеленим кольором. При переході на головний Дашборд загальний відсоток успішності

автоматично перераховується у більшу сторону, а у стрічці активності з'являється повідомлення про успішне завершення.

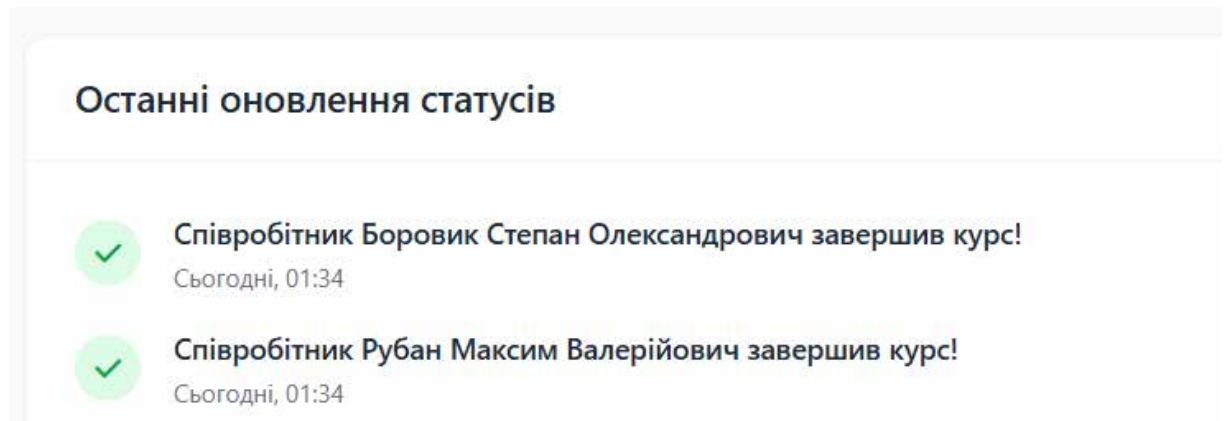


Рисунок 2.12 - Після проходження курсу в «Останні оновлення статусів» прописується хто завершив курс

Загальний підсумок

На основі проведених тестів можна зробити висновок, що програмний продукт: демонструє стабільну роботу в усіх основних сценаріях; коректно обробляє помилки вводу та попереджає користувача про них; забезпечує правильні математичні обчислення загальної статистики; адаптується до різної кількості вхідних даних, не втрачаючи швидкодії завдяки підходу Single Page Application; має зручний та інтуїтивно зрозумілий інтерфейс.

Усі заявлені функціональні та нефункціональні вимоги були успішно протестовані та підтверджені на практиці. Виявлені незначні недоліки не впливають на загальну працездатність системи, а лише вказують на потенційні напрямки вдосконалення у майбутньому (наприклад, додавання пагінації для дуже великих таблиць).

4.2 Аналіз результатів тестування

Після завершення тестування комп'ютерної системи обліку та контролю підвищення кваліфікації персоналу було здійснено детальний аналіз отриманих результатів. Основна мета цього етапу полягає в оцінці ступеня відповідності розробленого вебдодатка технічним та функціональним вимогам, виявленні сильних сторін системи, а також визначенні можливих

напрямків покращення.

Оцінка відповідності функціональним вимогам

Аналіз показав, що реалізоване програмне забезпечення повністю задовольняє ключові функціональні вимоги, сформульовані на етапі проєктування. Серед них:

Управління персоналом та курсами. Система дозволяє адміністратору безперешкодно додавати нових працівників та створювати навчальні програми, що забезпечує гнучкість наповнення бази даних.

Контроль навчання. Реалізовано безпомилкове призначення курсів обраним співробітникам із встановленням дедлайнів та захистом від випадкового дублювання призначень.

Автоматичний розрахунок статистики. Обчислення відсотка успішності та загальної кількості записів на дашборді здійснюється миттєво після зміни статусів навчання.

Зручне представлення даних. Інформація виводиться у вигляді структурованих таблиць та карток із кольоровими індикаторами прогресу, що значно полегшує її візуальне сприйняття.

Ці функціональні блоки пройшли успішне тестування у сценаріях як із невеликою, так і з великою кількістю введених даних, що дозволяє стверджувати про їхню стійку реалізацію.

Контроль навчання. Реалізовано безпомилкове призначення курсів обраним співробітникам із встановленням дедлайнів та захистом від випадкового дублювання призначень.

Автоматичний розрахунок статистики. Обчислення відсотка успішності та загальної кількості записів на дашборді здійснюється миттєво після зміни статусів навчання.

Зручне представлення даних. Інформація виводиться у вигляді структурованих таблиць та карток із кольоровими індикаторами прогресу, що значно полегшує її візуальне сприйняття.

Ці функціональні блоки пройшли успішне тестування у сценаріях як із

невеликою, так і з великою кількістю введених даних, що дозволяє стверджувати про їхню стійку реалізацію.

Надійність і стабільність системи

Одним із ключових критеріїв оцінки є надійність, тобто здатність програми працювати стабільно без збоїв, аварійних завершень чи непередбачуваної поведінки. На основі тестування було встановлено:

Додаток не зазнає збоїв при обробці великих масивів даних завдяки використанню швидкої бази даних H2.

Відсутні затримки або помилки відображення при перемиканні між вкладками завдяки архітектурі Single Page Application.

Усі динамічно створені елементи інтерфейсу (модальні вікна, випадаючі списки) повністю зберігають свою функціональність.

Особливо варто відзначити багаторівневий захист від помилкових введень. Система перевіряє, чи всі обов'язкові поля заповнено та чи відповідають дані очікуваним форматам. У разі виявлення порушень користувач отримує попередження, а відправка некоректного запиту на сервер блокується, що гарантує збереження логічної цілісності бази даних.

Точність обчислень та обробки статусів

Критично важливим компонентом системи є реалізація алгоритмів оновлення прогресу. Аналіз показав, що:

Результати обчислень загального рейтингу успішності повністю відповідають очікуваним математичним значенням.

Логіка переходу статусів жорстко прив'язана до відповідних відсоткових показників (0%, 50%, 100%), що виключає можливість некоректного ручного введення та гарантує точність моніторингу.

Зручність використання інтерфейсу

Важливу роль у прийнятті рішення щодо ефективності програмного продукту відіграє інтерфейс користувача. Його зручність і логічність визначають практичну цінність системи. Аналіз показав такі позитивні аспекти:

Логічна структура. Усі етапи роботи розміщені послідовно та інтуїтивно зрозуміло.

Підказки до полів. Наявність плейсхолдерів у формах допомагає користувачу орієнтуватися без потреби в додаткових інструкціях.

Адаптивність. Реалізовано коректне масштабування елементів та появу смуг прокрутки, що дозволяє працювати з великими таблицями на будь-яких дисплеях.

Мінімалізм. Використання сучасного фреймворку Tailwind CSS забезпечило чистий дизайн без зайвих декоративних елементів.

Таким чином, користувач без глибокої технічної підготовки може ефективно працювати з вебдодатком після мінімального ознайомлення.

Обмеження, виявлені під час аналізу

Попри загальну позитивну оцінку функціонування додатка, було виявлено кілька потенційних напрямів для вдосконалення:

Відсутність автоматичної пагінації (розбиття на сторінки). При дуже великій кількості працівників у базі таблиця стає занадто довгою для комфортного вертикального прокручування.

Відсутність гнучкого сортування. Наразі записи у таблицях виводяться у порядку їх створення, без можливості швидкого динамічного сортування за алфавітом чи за найближчим дедлайном.

Відсутність інструментів експорту. Результати статистики та списки призначень наразі неможливо завантажити у форматі PDF або електронної таблиці.

Ці обмеження не є критичними для роботи розробленої системи, однак їх усунення у наступних версіях дозволить суттєво підвищити якість користувацького досвіду.

Узагальнені висновки аналізу

За результатами тестування можна стверджувати, що розроблена система ефективно реалізує поставлені функції обліку та контролю підвищення кваліфікації. Вона забезпечує стійку, передбачувану та захищену

клієнт-серверну роботу, має інтуїтивно зрозумілий інтерфейс та демонструє високу швидкість обробки даних. Програма виконала свою основну задачу, а саме надала зручний інструмент для моніторингу навчального процесу на підприємстві. Виявлені обмеження не порушують основної логіки, але відкривають можливості для майбутніх оновлень і розширення функціоналу.

Проведене тестування вебдодатка підтвердило працездатність усіх реалізованих функціональних модулів. Застосована методика дозволила перевірити додаток на відповідність ключовим вимогам. Аналіз результатів тестування засвідчив готовність системи до використання і продемонстрував доцільність обраного стеку технологій (Spring Boot, JavaScript, Tailwind CSS), закладаючи міцне підґрунтя для подальшої підтримки системи.

ВИСНОВКИ

У даній дипломній роботі було виконано повний цикл розробки комп'ютерної системи обліку та контролю підвищення кваліфікації персоналу. Основною метою проєкту було створення сучасного, надійного та інтуїтивно зрозумілого вебдодатка, який дозволяє автоматизувати процеси управління корпоративним навчанням, відстежувати прогрес співробітників та формувати загальні статистичні дані для керівництва підприємства.

У процесі виконання роботи було:

1. Проведено детальний аналіз предметної області, визначено основні вимоги до систем управління навчанням та виявлено ключові потреби потенційних користувачів
2. Обґрунтовано вибір клієнт-серверної архітектури та сучасних програмних рішень, зокрема фреймворку Spring Boot для серверної частини, мови програмування Java, а також JavaScript і бібліотеки Tailwind CSS для клієнтської частини
3. Спроектовано логічну структуру бази даних H2 для зберігання інформації про штат працівників, доступні навчальні програми та історію призначень курсів
4. Розроблено серверну бізнес-логіку з використанням архітектурного стилю REST API, що забезпечує швидку та безпечну обробку мережових запитів
5. Реалізовано клієнтський інтерфейс за принципом Single Page Application (односторінковий додаток), який гарантує миттєву реакцію на дії користувача без перезавантаження вебсторінки та адаптується під різні розміри екранів
6. Проведено комплексне ручне тестування розробленого програмного продукту на основі реальних сценаріїв використання, що підтвердило його стабільність, точність обчислень та коректність роботи рольової моделі

Результатом роботи стала повноцінна функціональна система, яка дозволяє:

розмежовувати права доступу користувачів за ролями адміністратора та глядача для забезпечення захисту даних;

вести зручний електронний облік персоналу з можливістю швидкого пошуку, додавання нових співробітників та закріплення їх за відділами;

формувати та редагувати загальний корпоративний каталог навчальних курсів із зазначенням організатора, тривалості та рівня складності;

здійснювати цільове призначення курсів обраним працівникам із встановленням термінів виконання (дедлайнів) та автоматичним блокуванням дублюючих призначень;

відстежувати етапи проходження навчання за допомогою зрозумілих кольорових індикаторів та статусів (заплановано, в процесі, завершено);

миттєво отримувати зведену статистику успішності підприємства на головній інформаційній панелі (дашборді).

Створена комп'ютерна система є гнучкою, масштабованою та універсальною. Її можна ефективно використовувати не лише в комерційних компаніях, але й у навчальних закладах, державних установах чи будь-яких інших організаціях, де існує потреба в централізованому моніторингу за професійним розвитком та сертифікацією штату

Перспективи розвитку

Незважаючи на високу працездатність та готовність поточної версії системи до практичного впровадження, існує низка перспективних напрямків для її подальшого вдосконалення:

розширення аналітичного модуля з можливістю автоматичного генерування та експорту розширених звітів у формати PDF або Excel;

додавання механізмів автоматичного сповіщення співробітників про наближення дедлайнів або призначення нових курсів через електронну пошту;

інтеграція підсистеми тестування для перевірки знань безпосередньо у

вебдодатку після завершення вивчення матеріалів;

реалізація автоматичної пагінації та механізмів динамічного багатофакторного сортування для ще зручнішої роботи з дуже великими масивами даних;

перехід від бази даних в оперативній пам'яті до повноцінної реляційної СУБД (наприклад, PostgreSQL) для забезпечення довгострокового зберігання корпоративної історії.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Fowler M. Patterns of Enterprise Application Architecture. Addison-Wesley Professional, 2002. [Електронний ресурс] // martinowler. – Режим доступу до ресурсу: <https://martinfowler.com/books/eaa.html>
2. Walls C. Spring in Action, 6th Edition. Manning Publications, 2022. [Електронний ресурс] // manning. – Режим доступу до ресурсу: <https://www.manning.com/books/spring-in-action-sixth-edition>
3. Noe R.A. Employee Training and Development, 7th Edition. McGraw-Hill Education, 2016. [Електронний ресурс] // mheducation. – Режим доступу до ресурсу: <https://www.mheducation.com/highered/product/employee-training-development-noe.html>
4. Dessler G. Human Resource Management, 16th Edition. Pearson, 2020. [Електронний ресурс] // pearson. – Режим доступу до ресурсу: <https://www.pearson.com/en-us/subject-catalog/p/human-resource-management/P200000003004>
5. Fielding R.T. Architectural Styles and the Design of Network-based Software Architectures, 2000. [Електронний ресурс] // ics.uci. – Режим доступу до ресурсу: <https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>
6. Richardson L., Ruby S. RESTful Web Services. O'Reilly Media, 2007. [Електронний ресурс] // oreilly. – Режим доступу до ресурсу: <https://www.oreilly.com/library/view/restful-web-services/9780596529260/>
7. Masse M. REST API Design Rulebook. O'Reilly Media, 2011. [Електронний ресурс] // oreilly. – Режим доступу до ресурсу: <https://www.oreilly.com/library/view/rest-api-design/9781449317904/>
8. Flanagan D. JavaScript: The Definitive Guide, 7th Edition. O'Reilly Media, 2020. [Електронний ресурс] // oreilly. – Режим доступу до ресурсу: <https://www.oreilly.com/library/view/javascript-the-definitive/9781491952016/>
9. Sommerville I. Software Engineering, 10th Edition. Pearson, 2015. [Електронний ресурс] // pearson. – Режим доступу до ресурсу: <https://www.pearson.com/en-us/subject-catalog/p/software->

engineering/P200000003238

10. StatCounter Global Stats [Электронный ресурс] // gs.statcounter. – Режим доступа до ресурсу: <https://gs.statcounter.com/>

11. Spring Boot Reference Documentation. [Электронный ресурс] // docs.spring. – Режим доступа до ресурсу: <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/>

12. Oracle. Java 17 API Specification [Электронный ресурс] // docs.oracle. – Режим доступа до ресурсу: <https://docs.oracle.com/en/java/javase/17/docs/api/index.html>

13. Spring Data JPA - Reference Documentation. [Электронный ресурс] // docs.spring. – Режим доступа до ресурсу: <https://docs.spring.io/spring-data/jpa/docs/current/reference/html/>

14. Nielsen J. Usability Engineering. Morgan Kaufmann, 1993. [Электронный ресурс] // dl.acm. – Режим доступа до ресурсу: <https://dl.acm.org/doi/pdf/10.5555/2821575>

15. H2 Database Engine Official Documentation. [Электронный ресурс] // h2database. – Режим доступа до ресурсу: <https://www.h2database.com/html/main.html>

16. MDN Web Docs: JavaScript [Электронный ресурс] // developer.mozilla. – Режим доступа до ресурсу: <https://developer.mozilla.org/uk/docs/Web/JavaScript>

17. Using Fetch API – Web APIs | MDN [Электронный ресурс] // developer.mozilla. – Режим доступа до ресурсу: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch

18. Tailwind CSS Documentation [Электронный ресурс] // tailwindcss. – Режим доступа до ресурсу: <https://tailwindcss.com/docs>

19. Myers G.J., Sandler C., Badgett T. The Art of Software Testing, 3rd Edition. John Wiley & Sons, 2011. [Электронный ресурс] // wiley. – Режим доступа до ресурсу: <https://www.wiley.com/en-us/The+Art+of+Software+Testing,+3rd+Edition-p-9781118031964>

20. Ries E. The Lean Startup. Crown Business, 2011. [Электронный ресурс] // us.archive. – Режим доступа до ресурсу: <https://ia800509.us.archive.org/7/items/TheLeanStartupErickRies/The%20Lean%20Startup%20-%20Erick%20Ries.pdf>