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

БАКАЛАВРСЬКА ДИПЛОМНА РОБОТА
на тему:

Веб-застосунок для організації та оцінювання навчального процесу

ПОЯСНЮВАЛЬНА ЗАПИСКА

Виконав студент 4 курсу, групи 1КІ-18б
спеціальності 123 — Комп'ютерна інженерія
Терещук Е. П.

Керівник к.т.н., доц. каф. ОТ
Городецька О. С.

Опонент д.т.н., проф. каф. ЗІ
Лужецький В. А.

Допущено до захисту
д.т.н., проф. Азаров О.Д.

" 20 " червня 2022 р.

ВНТУ 2022

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Освітній рівень — бакалавр
Спеціальність — 123 Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри
обчислювальної техніки
проф. Азаров О.Д.
«10» лютого 2022 р.

З А В Д А Н Н Я

НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Терещуку Едуарду Петровичу

1. Тема роботи «Веб-застосунок для організації та оцінювання навчального процесу», керівник Городецька Оксана Степанівна к.т.н., доцент кафедри ОТ, затверджено наказом вищого навчального закладу від «24» березня 2022 року №66.

2. Строк подання студентом проекту 20 червня 2022.

3. Вихідні дані до роботи: середовище розробки IntelliJ, контент для заповнення бази даних.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): вступ, аналіз предметної області, аналіз та обґрунтування вибору технологій, розробка веб-застосунку, тестування веб-застосунку.

5. Дата видачі завдання 24 березня 2022 року.

6. Календарний план виконання БДР приведений в таблиці 1.

Таблиця 1 — Календарний план

№ з/п	Назва етапів виконання комплексної бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Постановка задачі роботи	14.03.22	Виконано
2	Аналіз предметної області	15.03-18.03.22	Виконано
3	Особливості систем управління навчальним процесом	21.03-24.03.22	Виконано
4	Аналіз та обґрунтування вибору технологій	25.03-30.03.22	Виконано
5	Вибір архітектури сервісу	31.03-04.04.22	Виконано
6	Вибір технологій для створення клієнтської частини застосунку	05.04-.11.04.22	Виконано
7	Розробка веб-застосунку	12.04-02.05.22	Виконано
8	Тестування серверної та клієнтської частини	03.05-19.05.22	Виконано
9	Підготовка матеріалів та опис розробки інформаційної системи	23.05-26.05.22	Виконано
10	Аналіз виконання роботи, висновки, додатки	27.05-31.05.22	Виконано
11	Перевірка якості виконання бакалаврського проекту та усунення недоліків	01.06-11.06.22	Виконано

Студент _____ Терещук Е. П.

Керівник роботи _____ Городецька О. С.

АНОТАЦІЯ

Пояснювальна записка містить 102 сторінки, 36 рисунків, 5 таблиць та 29 посилань.

В даній бакалаврській дипломній роботі був розроблений веб-застосунок організації та оцінювання навчального процесу.

На основі аналізу предметної області та порівняльного аналізу методів проектування та побудови клієнт-серверних сервісів обґрунтовано вибір оптимальних технологій як інструментів розробки веб-застосунку для оцінювання навчального процесу.

Відповідно до поставленої задачі було розроблено загальну структуру, базу даних, клієнт-серверної частини застосунку та їх тестування. Також було передбачено можливість функціонування застосунку у різних операційних системах. Таким чином було отримано гнучкий, мультиплатформенний, легкий для масштабування та розширення функціональності застосунок, що спрощує організацію навчального процесу та дозволяє легко оцінювати навчання користувачів у системі.

Ключові слова: веб-застосунок, веб-ресурс, процес, оцінювання, дистанційне навчання.

ABSTRACT

The explanatory note contains 102 pages, 36 figures, 7 tables and 29 references.

In this bachelor's thesis, a web application for the organization and evaluation of the educational process was developed.

Based on the analysis of the subject area and comparative analysis of methods of design and construction of client-server services, the choice of optimal technologies as tools for developing a web application for evaluating the educational process is substantiated.

In accordance with the task, the general structure, database, client-server part of the application and their testing were developed. It was also possible to operate the application in different operating systems. This resulted in a flexible, multi-platform, easy-to-scale and scalable application that simplifies the organization of the learning process and allows easy assessment of user learning in the system.

Keywords: web application, web resource, process, evaluation, distance learning.

ЗМІСТ

ВСТУП

Останнім часом, у багатьох закладах середньої та вищої освіти виникла проблема щодо проведення освітнього процесу в умовах дистанційного навчання. Зокрема через пандемію COVID-19, дистанційне навчання стало трендом сучасної освіти в світі. Отож, педагоги змушені активно впроваджувати дистанційні технології в процес навчання у закладах освіти. Освітні веб-ресурси — це електронні інформаційні ресурси, які призначені для забезпечення процесу навчання на визначених ступенях освіти і з певної предметної галузі, розміщенні у веб-просторі локальної чи глобальної мережі у вигляді різних форматів [1].

Вирішенням даної проблеми може стати розробка веб-ресурсу для організації та проведення тестів і опитувань, що дозволить дуже швидко здійснювати контроль, максимально автоматизувати процеси тестування і опитування, а також використовувати тести і опитування у дистанційному навчанні. Застосування інформаційних технологій в освіті сприяє підвищенню мотивації навчання учнів, економії навчального часу, а інтерактивність і наочність сприяє кращому уявленню, розумінню та засвоєнню навчального матеріалу, дозволяє значно полегшити організацію навчального процесу, перевірку та оцінювання знань учнів [2].

Також із розвитком новітніх технологій, поширенням та доступністю інтернет-зв’язку з’являється унікальна можливість для викладачів та студентів проводити та освоювати навчальну програму. Зокрема для викладачів є більш зручним введення електронного журналу, оскільки він економить час, в нього легко можна внести зміни, та є можливість поширити його серед студентів та інших викладачів. Програмне забезпечення для даного застосунку може бути використано на будь-якій операційній системі, на якій встановлено браузер, який підтримує останні веб-стандарти, а також має постійний доступ до інтернету.

Отже, веб-застосунок для організації та оцінювання навчального процесу має великий ряд переваг над іншими варіантами реалізації навчального процесу. Користувачами такої системи можуть бути викладачі, які створюють власні напрями навчання по конкретній спеціальності, та студенти, що мають доступ до всіх можливих навчальних курсів, які бажають отримати знання у певній сфері.

Об’єктом дослідження є процес проектування веб-застосунку, створення оцінювання робіт студентів.

Предметом дослідження є програмне забезпечення реалізації веб-застосунку.

Метою написання бакалаврської роботи є розробка веб-застосунку для оцінювання та організації навчального процесу.

Для досягнення поставленої мети необхідно вирішити такі задачі:

— проаналізувати предметну область та провести порівняльну характеристику аналогів оцінювання навчального процесу;

— обґрунтувати вибір технологій для розробки веб-застосунку;

— розробити загальну структуру веб-застосунку та бази даних;

— розробити веб-застосунок;

— провести тестування серверної та клієнтської частини веб-застосунку.

Апробація результатів роботи здійснена в доповіді на LI науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії.

Матеріали роботи доповідались та опубліковувались [3]:

Терещук Е.П. Веб-застосунок для організації та оцінювання навчального процесу / Е.П. Терещук, О. С. Городецька, Л. А. Савицька // Тези доповіді. LI науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії. Вінниця 2022 р. Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2022/paper/view/15131/12753.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття «Веб-застосунок»

Веб-застосунок **(**веб-додаток) — це розподілений застосунок, в якому клієнтом виступає браузер, а сервером – веб-сервер. Зазвичай браузер може бути реалізацією так званих тонких клієнтів, або ж пристроїв уведення та відображення інформації. Логіка застосунку зосереджується на сервері, а функція браузера полягає переважно у зображенні інформації, що завантажується мережею з сервера, і передачі назад даних користувача. Однією з переваг такого підходу є той факт, що клієнти не залежать від конкретної операційної системи користувача, тому веб-застосунки є міжплатформовими сервісами. [4]

Виділяють 4 ознаки, які дозволяють відрізнити веб-застосунки від інших типів додатків:

— вони вимагають лише одного процесу розробки;
— веб-застосунки не потрібно завантажувати;
— вони доступні з будь-якого браузера;
— займають своє місце в пошукових системах;

Хоча нативні програми дозволяють розробляти складніші розробки, вони також дорожчі, а іноді й непотрібні.

Веб-застосунки в свою чергу мають власну класифікацію. Ця класифікація заснована на функціональності, яку дозволяє використовувати застосунок та способі представлення цієї функціональності. Таким чином веб-застосунки поділяються на 5 різних типів, кожен з яких має свої особливості. То ж вони поділяються на:

— статичні веб-застосунки;
— динамічні веб-застосунки;
— односторінкові програми;
— веб-портали (Portal веб app);
— системи управління контентом (CMS);

При потребі створити статичну веб-програму, перше, що потрібно знати, це те, що веб-застосунки такого типу показують дуже мало вмісту і не дуже гнучкі. Зазвичай вони розробляються за допомогою HTML і CSS. Однак анімовані об’єкти, такі як банери, GIF-файли, відео тощо, також можуть бути включені та показані в них. Їх також можна розробляти за допомогою jQuery та Ajax.

Прикладами розробки статичних веб-додатків є професійні портфоліо або цифрові навчальні програми. Подібним чином веб-сторінка, що представляє компанію, також може використовувати такий тип веб-програми для відображення контактної інформації тощо.

Крім того, змінювати вміст статичних веб-програм непросто. Для цього спочатку потрібно завантажити HTML-код, потім змінити його і, нарешті, знову завантажити на сервер. Ці зміни можуть бути внесені лише веб-майстром або компанією-розробником, яка спочатку спланувала та розробила програму.

Динамічні веб-застосунки набагато складніші на технічному рівні. Вони використовують бази даних для завантаження даних, і їх вміст оновлюється щоразу, коли користувач звертається до них. Зазвичай вони мають панель адміністрування (так звану CMS), з якої адміністратори можуть виправляти або змінювати вміст програми, включаючи текст та зображення. PHP та ASP є найпоширенішими мовами, які використовуються для цієї мети, оскільки вони дозволяють структурувати вміст.

У таких програмах оновлювати вміст дуже просто, і навіть не потрібно звертатися до сервера під час його зміни. Крім того, він дозволяє реалізувати безліч функцій, таких як форуми або бази даних. Дизайн, окрім змісту застосунку, можна змінити відповідно до уподобань адміністратора.

Якщо веб-додаток є інтернет-магазином або магазином, його розробка, швидше за все, буде нагадувати розробку мобільної комерції (m-commerce) або сайту електронної комерції (e-commerce). Процес розробки такого додатка є складнішим, оскільки він повинен включати електронні платежі за

допомогою кредитних карток, PayPal або інших способів оплати. Розробник також повинен створити панель керування для адміністратора. Він використовуватиметься для переліку нових продуктів, їх оновлення або видалення та керування замовленнями та платежами [5].

Під порталом, в свою чергу, ми маємо на увазі своєрідну програму, в якій ми отримуємо доступ до кількох його розділів або категорій через домашню сторінку.

Ці програми можуть містити багато речей:

— форуми;

— чати;

— електронну пошту;

— пошукові системи;

— частини програм, доступ до яких здійснюється через реєстрацію.

Контент необхідно постійно оновлювати, коли справа доходить до розробки веб-додатків, тому встановлення системи керування вмістом (контентом) (CMS) є серйозним варіантом, який варто розглянути. Адміністратор може використовувати цю CMS для внесення змін та оновлень. Ці менеджери контенту інтуїтивно зрозумілі та дуже зручні для користувачів. Приклад такої системи управління можемо спостерігати на рисунку 1.1 [6].

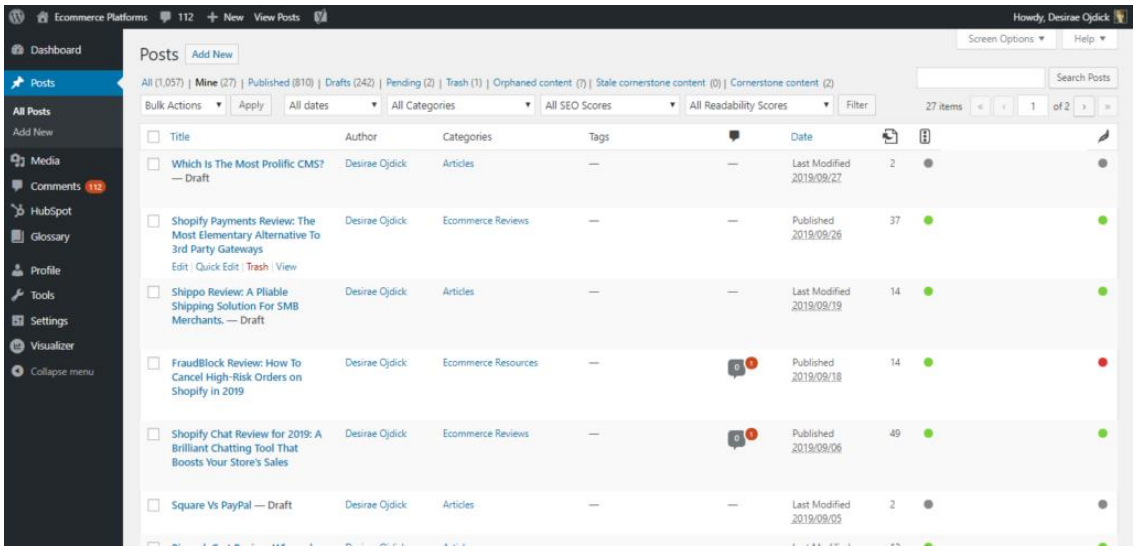


Рисунок 1.1 — Приклад інтерфейсу системи управління контентом

Ось приклади деяких систем керування вмістом:

— WordPress;
— Joomla;
— Drupal.

Багато компаній вирішують створювати веб-додатки, щоб покращити взаємодію користувачів зі своїми послугами. Ми можемо навіть використовувати деякі з них у повсякденному житті, не підозрюючи про це. Google, наприклад, має версію веб-застосунків усіх своїх сервісів: Календар, Диск, Карти, Gmail, YouTube. Версія веб-додатків цих порталів дуже схожа на оригінальні програми.

За прикладом електронної комерції варто звернутися до Amazon'у. Знову ж таки, він пропонує дуже схожий досвід з оригінальною програмою, а також має версію веб-застосунку для своїх платформ Amazon Video та Amazon Music.

Іноді розробка веб-додатків здійснюється, щоб зробити мобільний додаток доступним з комп'ютера. Прикладом є платформа обміну миттєвими повідомленнями WhatsApp або Telegram. Обидва дозволяють обмінюватись повідомлення, отримувати доступ до мультимедійних файлів тощо.

1.2 Переваги веб-застосунків

У порівнянні з настільними додатками, веб-додатки надають цілий ряд переваг для бізнесу. Використання веб-програм зазвичай називають програмним забезпеченням як послугою (SaaS), де програми працюють у віртуальному хмарному середовищі. Додатки SaaS дають відчутні переваги для бізнесу в порівнянні з локальним програмним забезпеченням. До таких переваг відносять:

— доступність на різних пристроях;
— сумісність з багатьма платформами;
— високий рівень захищеності даних;
— зниження витрат на розробку продукту;

— підвищена гнучкість та масштабованість продукту.

Розглянемо кожну з переваг детальніше.

Програми SaaS можна використовувати в більшості браузерів і працюватимуть вони однаково в кожній операційній системі, незалежно від оновлення або версії, що використовується. Більшість програмного забезпечення для настільних комп'ютерів насправді можна розробити у вигляді веб-застосунку, а це означає, що користувач може використовувати його з будь-якого мобільного пристрою, де б він не перебував. Також веб-застосунки дають набагато більші можливості для інтеграції з іншими системами, ніж настільні програми. Локалізоване програмне забезпечення ізольоване в порівнянні з веб-застосунки, які значно краще взаємодіють. Це пояснюється тим, що веб-застосунки можуть бути пов'язані між собою легше, ніж дві повністю окремі системи [7].

Щодо убезпечення даних користувача, веб-застосунок пропонує хороший спосіб безпечного доступу до централізованих даних. До серверів буде звертатися безпосередньо особа, яка ними керує. Завдяки використанню хмарних обчислень сервери можуть бути повністю резервними та реплікованими, щоб запобігти простою в результаті непередбаченої ситуації, або помилки. Це дозволяє уникнути необхідності підтримувати безпеку кожного пристрою, на якому використовується програма. Загалом ризик несанкціонованого доступу зменшується, а введення заходів безпеки стає простішим, оскільки це робиться централізовано.

За допомогою веб-застосунків можна значно знизити витрати завдяки зменшенню підтримки та обслуговування, меншим вимогам до системи кінцевого користувача та спрощеній архітектурі. Подальша оптимізація бізнес-операцій за рахунок використання веб-додатків часто дозволяє знайти додаткову економію [8].

Зазвичай веб-застосунки не зупиняються у плані розробки нового функціоналу, або розширення вже існуючого. Оскільки потрібно, щоб більша кількість процесів відбувалися одночасно, програмне забезпечення може

сприяти цьому. І веб-застосунки виконують цю умову. Чудова можливість, що підтримується ними — це розробка за мікросервісною технологією. Архітектура мікросервісів дотримується модульного підходу до розробки різних програм. Мікросервісна архітектура містить набір невеликих, незалежних та автономних модулів, які надають різні послуги. Цей варіант надає можливість легко пов’язати між собою кілька застосунків та розробляти кожен з них використовуючи абсолютно різні програмні засоби, наприклад, бази даних, мови програмування та тому подібне.

1.3 Особливості систем управління навчальним процесом

Система управління навчанням, або Learning Management Systems (LMS) — це система, створена спеціально для того, щоб допомогти керувати, відстежувати й оцінювати безперервний процес навчання та всю його діяльність у системах освіти через мережу інтернет, або локальну мережу. Вона має більше, ніж комунікаційні функції та інструменти для моніторингу користувачів. Онлайн системи управління навчанням характеризується здатністю керувати, організовувати, використовувати, впроваджувати курси розроблені в електронному вигляді для шкіл, університетів та різноманітних компаній, що зацікавленні у розвитку власних працівників. LMS визначають як інтегровані програмні пакети, які є системою для управління процесом онлайн навчання та забезпечують комунікацію між сторонами освітньої системи в будь-який час і з будь-якого місця через Інтернет, або локальну мережу з метою покращення процесу навчання учнів, або персоналу [9].

Хоча використання LMS має багато переваг, як-от зниження витрат і безпека, головна мета системи управління навчанням полягає в тому, щоб забезпечити та відстежувати процес навчання в одному місці. Те, як LMS виглядає та працює, залежить від цілей організації, але функції LMS повинні дозволити спеціалістам спростити такі процеси як:

— запис та підготовка розкладу для учнів курсу;
— збереження даних учнів та викладачів;

— запис на курс;
— відстеження та аналіз прогресу користувачів;
— адміністрування курсу;
— спілкування між викладачем і студентом.

Після встановлення головних цілей LMS слід визначити, яка саме система нам потрібна. Існує багато способів класифікації систем управління навчанням, але ось три основні типи:

— хмарні, або встановлені;
— з відкритим кодом, або приватні;
— безкоштовні, або платні.

Постачальники LMS, що пропонують встановлені рішення, займають меншу частину ринку. Це пов’язано з тим, що встановлені LMS часто вважаються «незграбнішими» та більш громіздкими, ніж хмарні сервіси. Вони вимагають, щоб користувачі встановили екземпляр LMS на власних серверах із власним обслуговуванням, а це означає, що клієнту потрібен спеціальний IT-фахівець із знаннями безпеки. Це робить встановлені рішення LMS дорожчими, ніж їхні хмарні аналоги.

Хмарні рішення LMS є сучасним рішенням та несуть менше витрат на налаштування та обслуговування, ніж встановлені версії. Крім того, постачальники LMS, які надають хмарні сервіси, також підтримують своє програмне забезпечення, безперешкодно пропонуючи оновлення своїм клієнтам. Навіть з цими додатковими функціями хмарні рішення мають нижчі щомісячні витрати і зазвичай не вимагають значної плати за налаштування.

Як встановлені, так і хмарні LMS пропонують програмне забезпечення з відкритим і закритим кодом. Програмне забезпечення з відкритим вихідним кодом передбачає, що власник авторських прав надає користувачам право вивчати, змінювати та поширювати програмне забезпечення будь-кому та для будь-яких цілей. Програмне забезпечення з відкритим вихідним кодом може розроблятися спільно з громадськістю. Програмне забезпечення із закритим

кодом — це програмне забезпечення власник якого зберігає право інтелектуальної власності.

Як згадувалося вище, програмне забезпечення LMS з відкритим кодом також надає користувачам більше свободи. Однак це накладає на компанію більше відповідальності за підтримку системи. Для багатьох корпоративних компаній краще знайти постачальника LMS, який буде займатися обслуговуванням системи [10].

Як і все інше на ринку технологій користувачам доступні безкоштовні та платні версії LMS. Якщо користувач обмежений в бюджеті, то безкоштовні LMS стають його ідеальним варіантом. Однак для компаній з великою кількістю користувачів безкоштовні варіанти не рекомендуються. Особливо, якщо компанія передбачає використання сервісу у постійному доступі та з найвищим рівнем обслуговування, що у разі неполадки системи понесе великі збитки для компанії.

Для того, щоб отримати найбільшу вигоду для користувача потрібно знайти найкращий сервіс LMS. Для користувачів, які прагнуть покращити досвід навчання потрібно знайти платформу, яка підтримує наступні функції LMS:

— інтеграція з іншими системами;
— можливість відстеження даних;
— персоналізований доступ до системи;
— офлайн трекери навчання;
— автоматичні сповіщення;
— централізація навчальних матеріалів;
— гнучка звітність та аналітика;
— підтримка доступу з різних пристроїв;
— безпека;
— інструменти оцінювання.

При виборі системи, що підтримує вище перераховані функції, користувач отримає максимальний досвід користуванням обраною системою.

1.4 Порівняльна характеристика аналогів

Серед великої кількості існуючих веб-застосунків оцінювання та організації навчального процесу було обрано 2 найпопулярніші сервіси, а саме Google Classroom та NEO.

Google Classroom — це вебсервіс Google розроблений у 2014 році для навчальних закладів з метою спрощення створення, поширення та класифікації завдань безпаперовим шляхом (рисунок 1.2) [11].

Рисунок 1.2 — Логотип Google Classroom

NEO LMS — це система дистанційного навчання, які являє собою хмарну інтелектуальну навчальну платформу для шкіл та університетів (рисунок 1.3) [11].

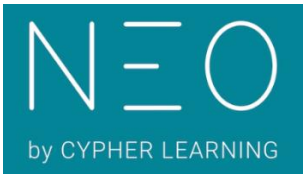


Рисунок 1.3 — Логотип NEO

Хоча ці сервіси призначенні для схожого завдання, спрощення роботи у навчальному процесі, вони мають ряд відмінностей між собою, які є їхніми перевагами, або недоліками. Розглянемо деякі з них.

Для їхнього порівняння оберемо декілька категорій, серед яких:

— інтерфейс взаємодії із системою;
— простота використання;
— персональний підхід;
— інтеграція з іншими системами;

— налаштування системи;

— комунікація між користувачами;

— вартість.

Інтерфейс взаємодії із системою. Інтерфейс NEO має адаптивний дизайн, який виглядає чітко і автоматично налаштовується залежно від типу пристрою. Він надає інформаційні панелі на основі плиток для студентів, викладачів та адміністраторів. Користувачі також можуть переглядати завдання, майбутні події, групи, календар тощо.

Інформаційна панель Google Classroom проста, але не настільки графічна. Викладачі та студенти можуть бачити лише ті курси, які вони викладають або на які зараховані.

NEO пропонує просту спливаючу навігаційну систему з ярликами та легким доступом до всіх областей сайту, незалежно від того, де ви знаходитесь на платформі. Користувачі можуть отримати доступ до класів, груп, налаштувань тощо зі спливаючого меню на лівій панелі замість того, щоб завантажувати іншу сторінку.

У Google Classroom головне меню можна приховати або показати, але немає спливаючої навігації для основних елементів у меню. Усі класи відображаються в лівому меню, тому користувачам потрібно прокрутити вниз ліву панель, щоб знайти клас, якщо їх у списку більше кількох (рисунок 1.4).

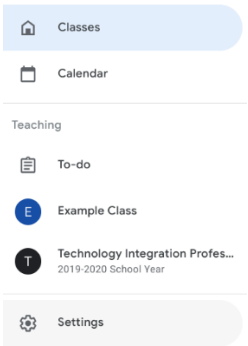


Рисунок 1.4 — Навігаційна панель в Google Classroom

Макет класу NEO дуже інтуїтивно зрозумілий, і користувачі можуть вибрати формат графічної плитки або рядка (рисунок 1.5). Кожна плитка

відображає важливу інформацію, як-от значки, бали та сертифікати, які присуджує урок. Учні можуть бачити свій прогрес у класі й уроці, позначених піктограмами на кожній плитці.

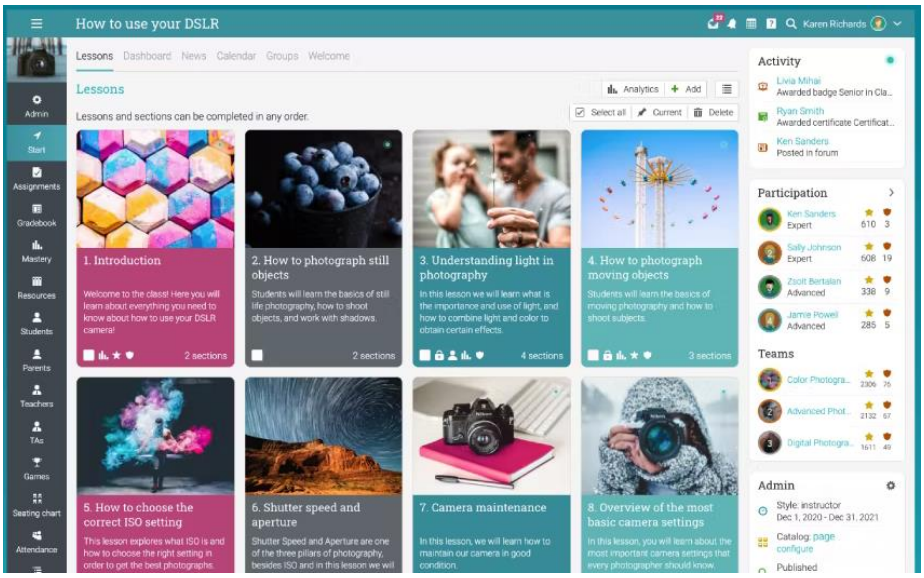


Рисунок 1.5 — Сторінка навчального курсу в NEO

Простота використання. NEO дуже інтуїтивно зрозумілий і простий у використанні. Крім того, він містить онлайн-довідковий центр із відео, посібниками з початку роботи та довідковим вмістом, доступним для пошуку. NEO також має форум підтримки швидкого реагування, де співробітники зазвичай відповідають на запитання протягом 15-30 хвилин у робочий час цілодобово.

Google Classroom дещо простіший для розуміння та використання, має довідковий центр з навчальними посібниками та довідковий форум, хоча не має настільки високого рівня підтримку з боку компанії, що надає свої послуги.

Особливості системи. NEO має всі основні функції, які допомагають вчителям заощадити час і зробити навчання більш цікавішим для учнів. У той час, коли Google Classroom надає дуже обмежений набір функцій, оскільки в ньому немає багатьох основних інструментів, таких як вбудований інструмент для створення підручника, автоматизації та має невеликий діапазон доступних типів завдань.

NEO пропонує 15 типів завдань, включаючи вікторини, есе, команду, дебати, дискусії, опитування. Весь перелік типів завдань можемо спостерігати на рисунку 1.6. У Google Classroom дуже обмежені можливості, оскільки вчителі можуть створювати завдання, додавати запитання (обговорення) та можливість вкладати файли з Google Drive. Щоб додати вікторину, вчителі мають використовувати Google Forms.

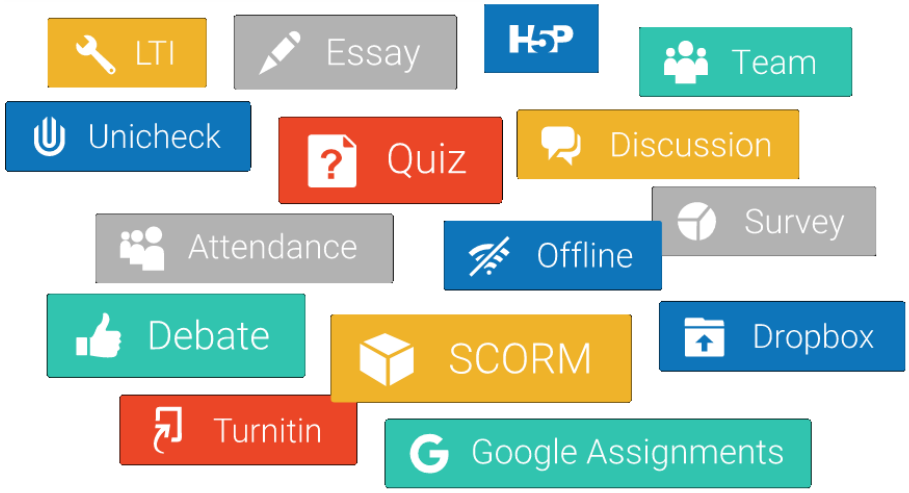


Рисунок 1.6 — Перелік всіх можливих типів завдань у NEO

NEO також має потужний інструмент для зведення оцінок із централізованим переглядом усіх оцінок, що дозволяє користувачам швидко додавати результати та вносити зміни. У Google Classroom немає функцій журналу оцінок.

Персональний підхід до учня. У NEO вчителі можуть вибирати дії, які слід виконувати, коли учні закінчують заняття, записуються в групи, відстають у своєму рівні майстерності тощо. Це допомагає їм заощадити час і зробити навчання більш персоналізованим. Google Classroom не має жодних функцій, які можуть зробити заняття персональними для учня та цікавішими.

Інтеграція з іншими сервісами. NEO інтегрується з найпопулярнішими та корисними освітніми інструментами, такими як Zapier, iCal, xAPI, Microsoft 365, OneDrivei Zoom. Крім того, він інтегрується з Google Drive, Google Workspace тощо.

У Google Classroom є можливість використовувати лише такі інструменти, як Google Drive або інструменти, які є частиною G Suite for Education.

Налаштування системи. У NEO школи можуть налаштовувати багато аспектів платформи, включаючи її колірну схему, шрифти, термінологію тощо.

Інтерфейс Google Classroom не можна налаштувати, окрім дрібних аспектів, таких як зміна основної теми уроку. Це може бути незручністю для шкіл, які хочуть додати персоналізований портал для відвідувачів, логотип, колірну схему або взагалі хочуть зробити свій портал схожим на власний.

Комунікація між користувачами. NEO допомагає користувачам співпрацювати за допомогою різних інструментів, таких як чат, повідомлення, групи, команди, соціальні мережі, стрічки новин, вікі, блоги тощо. Користувачі також можуть ділитися ресурсами через каталог ресурсів, який організовано за категоріями та підкатегоріями.

Google Classroom має обмежену функціональність для спілкування між викладачами та учнями, зокрема відсутній чат, вбудовані повідомлення, групи та соціальні мережі.

Вартість. NEO має безкоштовний план із повним набором основних функцій системи управління навчальним процесом та преміум-план для шкіл, які потребують більш потужних функцій. Немає жодної плати за налаштування системи, плати за підтримку та інших прихованих витрат. Хоча облікові записи вчителів, викладачів та батьків не входять у цей список.

Google Classroom є безкоштовним і доступним для будь якого користувача з обліковим записом Google, якщо вони використовують його окремо, а не для створення класів для школи чи університету. Однак школи чи університети, яким потрібні такі функції, як спільні календарі та необмежене хмарне сховище, можуть вибрати G Suite Enterprise for Education за 4 долари США за користувача на місяць для викладачів, співробітників і студентів.

Провівши аналіз аналогів, їхніх переваг та недоліків, визначено, яким має бути розроблюваний веб-застосунок, а саме: зі зручним та інтуїтивним інтерфейсом, можливість легкого редагування компонентів, з потужністю достатньою обробити велику кількість запитів на сервер та базою даних, яка зможе зберігати всі потрібні ресурси.

2 АНАЛІЗ ТА ОБГРУНТУВАННЯ ВИБОРУ ТЕХНОЛОГІЙ

2.1 Вибір архітектури сервісу

2.1.1 Типи архітектурних стилів проектування сервісу

Архітектура програмного забезпечення відіграє вирішальну роль у здатності сервісу масштабуватися та задовольняти потреби користувачів з часом. Недоліки будь-якого програмного забезпечення мають значний вплив на працездатність системи. Основною причиною будь-якого збою програмного забезпечення може бути вибір неправильної архітектури побудови сервісу.

Архітектурний шаблон проектування програмного забезпечення можна вважати контуром, який дозволяє виразити та визначити структурну схему для всіх видів програмних систем. Це рішення, яке надає заздалегідь визначений набір підсистем, ролей та обов’язків, включаючи правила та план для визначення взаємовідносин між ними. Це допомагає вирішити різні проблеми розробки програмного забезпечення, такі як обмеження продуктивності, високе навантаження на систему, мінімізація бізнес-ризиків тощо.

Незважаючи на те, що архітектурний шаблон є приблизним зображенням, або планом системи, це не фактична архітектура. Натомість це концепція, яка допомагає зрозуміти елементи архітектури програмного забезпечення. Може бути незліченна кількість архітектурних елементів, які реалізують той самий шаблон [13].

Відомими прикладами архітектурних шаблонів є багатошарова архітектура, мікросервісна, однорангова та клієнт-серверна архітектура.

Багатошарова, або n-рівнева архітектура — це архітектура, яка складається з кількох рівнів. Часто багатошарову архітектуру поділяють на чотири різні шари: шар презентації, бізнес-логіки, контролю та база даних. Однак шаблон не обмежується вказаними шарами, і може існувати рівень програм, рівень сервісу, або рівень доступу до даних. Популярні фреймворки, такі як Java EE, використовували цей шаблон архітектури.

Цей шаблон виділяється тим, що кожен шар відіграє окрему роль у програмі та позначений як закритий. Це означає, що запит повинен пройти через шар безпосередньо через нього, щоб перейти до наступного шару. Ще одна з його концепцій – шари ізоляції – дозволяє змінювати компоненти в межах одного шару, не впливаючи на інші шари.

Мікросервісна архітектура розглядається як альтернатива монолітним додаткам і сервіс-орієнтованим архітектурам. Компоненти такої системи представленні як окремі блоки за допомогою ефективної, спрощеної комунікації між ними. Переваги шаблону полягають у підвищеній масштабованості та низькому ступеню зв’язаності між блоками в програмі.

Завдяки своїм незалежним характеристикам, доступ до компонентів здійснюється через протокол віддаленого доступу. Більше того, ті самі компоненти можна окремо розроблятися, розгортатися та тестуватись без взаємозалежності від будь-якого іншого компонента служби [14].

Однорангова архітектура (peer-to-peer architecture) – це концепція інформаційної мережі, в якій її ресурси розосереджені по всіх системах. Окремий компонент (вузол) такої системи може діяти як клієнт, сервер, або те і інше, змінюючи динамічно свою роль з часом. Як клієнт, одноранговий вузол може запитувати обслуговування в інших однорангових компонентів, а як сервер — надавати послуги іншим однорангом вузлам. Істотна відмінність між одноранговою та клієнт-серверною архітектурою полягає в тому, що кожен комп’ютер у мережі має значні повноваження та відсутність централізованого сервера. Його потужність збільшується, оскільки все більше комп’ютерів приєднуються до мережі. Прикладом моделі однорангової архітектури можуть бути файлообмінні мережі, такі як Skype, BitTorrent і Napster [15].

Клієнт-серверна архітектура описується як розподілена структура додатка, що має два основні компоненти – клієнт і сервер. Ця архітектура полегшує зв'язок між клієнтом і сервером, які можуть бути, або не бути, в одній мережі. Клієнт запитує певні ресурси для отримання із сервера, які можуть бути у формі даних, вмісту, послуг, файлів тощо. Сервер ідентифікує

зроблені запити та відповідає клієнту належним чином, надсилаючи запитувані ресурси.

Функціональні характеристики клієнта і сервера є прикладом програм, які взаємодіють одна з одною в рамках програми. Функціональність цієї архітектури дуже гнучка, оскільки один сервер може обслуговувати кілька клієнтів, або один клієнт може використовувати кілька серверів. Сервери можна класифікувати за послугами або ресурсами, які вони надають, незалежно від того, як вони працюють.

Клієнт-серверна архітектура підходить найкраще для потреб системи керування навчальним процесом. Таким чином ми отримуємо можливість легко розширювати функціонал сервісу та обслуговувати систему.

2.1.2 Проектування клієнт-серверного застосунку

Отже як вище згадано архітектура клієнт-сервер складається з двох окремих частин, де клієнти запитують і отримують послуги від централізованого сервера. Клієнтські комп’ютери забезпечують інтерфейс, що дозволяє користувачеві комп’ютера надсилати запит серверу та відображати результати, які повертає сервер. Сервери чекають надходження запитів від клієнтів, а потім відповідають на них. В ідеалі сервер надає клієнтам стандартизований інтерфейс, щоб клієнтам не потрібно було знати про особливості системи, тобто апаратного та програмного забезпечення, яка надає послугу.

Клієнти часто знаходяться на робочих станціях, або на персональних комп’ютерах, тоді як сервери розташовані в інших місцях мережі, як правило, на більш потужних машинах. Ця обчислювальна модель особливо ефективна, коли клієнти і сервер мають різні завдання, які вони регулярно виконують. При обробці лікарняних даних, наприклад, на клієнтському комп’ютері може бути запущена прикладна програма для введення інформації про пацієнта, тоді як на серверному комп’ютері запущена інша програма, яка керує базою даних, в якій постійно зберігається інформація.

Також у ході розробки веб-сервісу розробник повинен визначити, яку концепцію потрібно обрати: stateful чи stateless. Stateful – зберігання додаткової інформацію на стороні сервера. Якщо клієнт надсилає запит на сервер, то він очікує певної відповіді, якщо він не отримує відповіді, то він надсилає запит повторно. Stateful підтримує стан всіх своїх сеансів, будь то сеанс аутентифікації або запит клієнта на інформацію. Статусними є ті, які можуть використовуватися неодноразово, наприклад, онлайн-банкінг або електронна пошта. Вони виконуються в контексті попередніх транзакцій, в яких зберігаються стани, і те, що відбувалося в попередніх транзакціях, може вплинути на поточну транзакцію. У зв'язку з цим stateful програми використовують ті самі сервери щоразу, коли вони виконують запит користувача. Прикладом збереження стану є FTP (File Transfer Protocol) та Telnet. Під час сеансу FTP, який часто включає багато передачі даних, клієнт встановлює контрольне з'єднання. Після цього відбувається передача даних.

Stateless в свою чергу зберігає додаткову інформацію на стороні клієнта (у веб-браузері), передаючи додаткову інформацію, «нагадуючи» серверу про попередні дії. Це зазвичай реалізується за допомогою файлів cookie або localStorage. Використання stateless веб-сервісу, або програми має ряд переваг перед stateful реалізацією. Реалізація stateful сервера складна для налаштування. Вони вимагають можливості посилатися на кілька станів, що може призвести до ряду незавершених транзакцій і сеансів у всій системі. Управління з'єднанням від клієнта до сервера також є важливим фактором, воно вимагає тісного з'єднання між кожною парою клієнта і сервера, що за своєю суттю є складним для масштабування обмеженням.

На базі цих двох концепцій було створено дві реалізації: SOAP та REST. SOAP — це абревіатура від Simple Object Access Protocol. SOAP — це протокол, який з'явився до REST. Основною причиною появи SOAP було забезпечення того, щоб програми, вбудовані в різні платформи або мови програмування, могли легко обмінюватися даними [16]. SOAP реалізовує

простий протокол доступу до об’єктів. Але оскільки він використовує складний формат XML, він, як правило, є повільним.

SOAP визначає структуру заголовка, яка визначає дії, які виконуватимуть різні вузли SOAP над повідомленням. Концепція маршрутизації повідомлення через ряд вузлів, які виконують різні функції, полягає в тому, як SOAP підтримує такі речі, як адресація, безпека та незалежність від формату [16].

На відміну від SOAP протоколу, REST — це архітектура дизайну API. Ключова відмінність між SOAP і REST полягає в тому, що SOAP використовує сервісні інтерфейси для розкриття функціональних можливостей, тоді як REST використовує Uniform Service Locator (уніфікований локатор послуг) для доступу до компонентів. REST дозволяє використовувати велику різноманітність форматів, тоді як SOAP підтримує лише XML. Також оскільки повідомлення SOAP мають більший розмір, вони вимагають великої пропускної здатності, що є протилежністю у випадку з REST.

Виходячи з потреб веб-застосунку найкраще підходить для розробки REST архітектура, через її високу пропускну спроможність запитів, що надходитимуть на сервер, можливість кешування даних, коли користувачу потрібно часто звертатися до тих самих ресурсів кілька разів, та можливість простого написання коду, що дозволить простим чином реалізувати потрібні служби, на відмінну SOAP.

2.2 Вибір бази даних для серверу

2.2.1 Вибір типу бази даних

Існує багато підходів до проектування баз даних, кожен з яких має власне призначення та виконує певні дії краще ніж інші. Для того, щоб визначити який саме підхід ми будемо обирати для реалізації власної бази даних потрібно розглянути їхні види та переваги. Типи баз даних поділяються на реляційні та нереляційні.

Реляційні база даних — це бази даних, в яких дані зберігаються в окремих таблицях, звідки до них можна отримати доступ або повторно зібрати різними способами за допомогою визначених користувачем реляційних таблиць. Таблиця складається з рядків і стовпців, і в реляційній базі даних рядки називаються записами, а стовпці називаються полями. Кожна таблиця має спеціальний стовпець, який містить лише окремі та унікальні значення, які називаються первинним ключем (primary key). Цей первинний ключ використовується для визначення зв'язку між таблицями, і якщо первинний ключ 1-ї таблиці, тобто ідентифікатор, використовується в 2-й таблиці, то цей стовпець називається зовнішнім ключем. Маючи реляційну базу даних ми можемо змінити або покращити дизайн бази даних шляхом її нормалізації, що представляє з себе процес структурування бази даних, щоб видалити надлишковість даних і покращити цілісність даних. Прикладом такої бази даних є MySQL.

Нереляційні бази даних (NoSQL) — це набір неструктурованих, або напівструктурованих елементів, зазвичай представлених у формі ключ-значення, які не потребують жодних таблиць, полів чи записів. Кожен елемент може представляти документ, таблицю або будь-що інше. Зазвичай різні елементи в одній або в різних колекціях не пов'язані один з одним, хоча можна додати до елемента посилання, щоб посилатись на інший елемент. Приклади нереляційних механізмів баз даних:

— MongoDB;
— Apache Cassandra;
— Redis;
— Apache HBase.

Бази даних NoSQL повністю відрізняються від баз даних SQL і працюють інакше. Вони можуть володіти будь-якими даними, будь то JSON, XML тощо. Отже, створювати дані та керувати ними в NoSQL легко та швидше.

Оскільки наш сервіс потребує зберігання структурованих даних про користувачів найкраще підійде реляційна база даних. Користувач мусить мати унікальний ідентифікатор, що характеризує його, та потрібен для зв’язування даних, які відповідають за зв’язки з навчальними курсами до яких відноситься користувач і для цієї задачі ідеально підійдуть первинні та зовнішні ключі, які підтримуються лише у реляційних базах даних.

2.2.2 Вибір підходу до проектування баз даних

Реляційні бази даних також поділяються за своєю архітектурою та організацією. Тому для збільшення ефективності варто обрати відповідну базу даних, що підходить для нашого сервісу по своїх архітектурі. Виділяються такі бази даних як централізовані та розподілені.

Централізована база даних — це тип бази даних, яка зберігається, розміщується та підтримується лише в одному місці. Цей тип бази даних змінюється та керується з місця свого розташування. Таким чином, це місце в основному є будь-якою системою баз даних або централізованою комп’ютерною системою. Доступ до централізованого розташування здійснюється через Інтернет (LAN, WAN тощо). Цю централізовану базу даних в основному використовують установи чи організації. Ось деякі з переваг централізованих баз даних:

— оскільки всі дані зберігаються лише в одному місці, легше отримати доступ до даних і координувати їх;

— централізована база даних має дуже мінімальну надмірність даних, оскільки всі дані зберігаються в одному місці;

— дешевше в порівнянні з усіма іншими доступними базами даних.

Розподілена база даних — це тип бази даних, який складається з кількох баз даних, які з'єднані одна з одною і розподілені в різних фізичних місцях. Таким чином, даними, які зберігаються в різних фізичних місцях, можна керувати незалежно від інших фізичних місць. Таким чином, зв’язок між

базами даних у різних місцях здійснюється за допомогою комп'ютерної мережі. Деякими перевагами даної архітектури виступає те, що:

— базу даних можна легко розширити, оскільки дані вже поширені в різних фізичних місцях;

— до розподіленої бази даних можна легко отримати доступ з різних мереж;

— база даних є більш безпечною в порівнянні з централізованою базою даних.

Для нашого застосунку оптимальним рішенням буде база даних з централізованою архітектурою, оскільки вона володіє всіма важливими характеристиками, які потребує наша система.

2.2.3 Вибір системи управління базами даних

Тепер оберемо систему управління реляційною базою даних, які відповідає визначеним критеріям. До таких можна віднести Oracle, MySQL та PostgresSQL.

Oracle — це система реляційних баз даних, яка забезпечує автономне керування, самозахист, самовідновлення та призначена для усунення схильного до помилок ручного керування базою даних. Oracle — це кросплатформна система баз даних, яка може працювати на різних операційних системах. Це дозволяє швидко та безпечно зберігати та отримувати дані. Вона доступна для студентів безкоштовно, але не може використовуватися в комерційних цілях. Це перше програмне забезпечення для баз даних, розроблене для бізнес-цілей для маніпулювання даними за допомогою мови запитів. Oracle був випущений в 1980 році з основними функціями SQL. Це програмне забезпечення є масштабованим, портативним, розподіленим і програмованим. Також Oracle має мережеві стеки, які дозволяють плавно обмінюватися додатками на різних платформах з базою даних Oracle.

MySQL є популярною системою керування базами даних, яка використовується для управління реляційною базою даних. Це програмне забезпечення баз даних з відкритим вихідним кодом, яке підтримується компанією Oracle. Це швидка, масштабована та проста у використанні система керування базами даних у порівнянні з Microsoft SQL Server та Oracle Database. Він зазвичай використовується зі сценаріями PHP для створення потужних і динамічних серверних або веб-додатків підприємства. MySQL підтримує багато операційних систем, таких як Windows, Linux, MacOS тощо з мовами C, C++ та Java [18].

PostgresSQL — це об'єктно-реляційна система управління базами даних, в той час коли MySQL просто реляційна. Це означає, що Postgres включає такі функції, як успадкування таблиць і перевантаження функцій, які можуть бути важливими для певних програм. Також вона започаткувала багато концепцій. PostgresSQL — це система реляційних баз даних корпоративного класу, її легко налаштувати та встановити. Вона пропонує підтримку SQL і NoSQL. Однією з переваг Postgres це обробка паралельності, на це я кілька причин, а саме: реалізує багатоверсійного контроль паралельності без блокування читання; підтримка паралельних запитів, які можуть використовувати кілька процесорів, або ядер; захищає цілісність даних на рівні транзакції, що робить Postgres менш вразливим до пошкодження даних.

Проаналізувавши переваги та недоліки кожної системи можна впевнено сказати, що для потреб веб-застосунку найкраще підходить MySQL, через її підтримку багатьох операційних систем та мов програмування.

2.3 Вибір технологій для створення серверної частини застосунку

2.3.1 Вибір мови програмування

Визначившись з архітектурою сервера та підходом до проектування бази даних потрібно обрати мову програмування та фреймворк, що дозволять побудувати внутрішню логіку застосунку. Щодо мов програмування, які орієнтовані на написання серверної частини сервісу виділяють такі мови

програмування як PHP, Node.js (JavaScript), Python, Ruby, Java. Розглянемо детальніше переваги та недоліки кожної мови програмування.

PHP є однією з найпопулярніших мов у веб-розробці, оскільки це була одна з перших мов, спеціально розроблених для створення динамічного контенту. Сьогодні понад 60% веб-сайтів так чи інакше використовують PHP, особливо тому що це чудовий інструмент для створення форумів та програм для обміну повідомленнями. Що стосується її основних переваг, то це те, що вона є відкритою, вона містить велику спільноту, що добре обслуговує, підтримує її і має десятки бібліотек для яких потрібно докласти дуже мало зусиль при їх використанні, та які чудово інтегруються з реляційними базами даних, і це є досить легким у використанні та розумінні [19]. Що стосується її слабких сторін, то через те, що вона є дуже відкритою збільшується відсоток появи помилки. Недосвідчені розробники можуть створювати вкрай неефективні рішення простих проблем за допомогою PHP.

JavaScript є однією з найпопулярніших мов програмування у світі, тому цілком зрозуміло, що зрештою придумали спосіб використовувати її для розробки серверної сторони додатків. JavaScript володіє великою спільнотою, що допомагає цій мові розширювати пул її можливостей, Node.js є однією з цим розширень. Перевага JavaScript полягає в тому, що Node.js — це легке й економне середовище виконання. Що дозволяю виконувати сценарії досить швидко, особливо в порівнянні з іншими конкурентами, такими як PHP. Той факт, що Node.js настільки легкий, робить його чудовим вибором для масштабованості мікросервісів. Що стосується його недоліків, то найяскравішим є те, що він заважає важким обчисленням. З кількох причин Node.js фактично однопотоковий, а це означає, що все, що покладається на велику обчислювальну потужність, буде працювати надзвичайно повільно [20].

Python — це досить потужна мова програмування, яку дуже легко читати й вивчати, вона має величезну підтримку та є бібліотека майже для всього, що може знадобитися при будь якій проблему та потребі. Більшість компаній

люблять Python через те, що за допомогою її легко розробляти програмні продукти. Однак більшість розробників серверних систем не використовують лише Python для створення своїх серверних частин. Замість цього вони покладаються на такі фреймворки, як Django і Flask, які справляються з великою кількістю важких завдань. З одного боку, Django обмінює універсальність на швидкість. Якщо розробник обмежений в часі, то він може розгорнути новий проект лише за кілька годин за допомогою таких фреймворків. Flask в свою чергу надзвичайно легкий, настільки, що йому не вистачає функцій, які надаються в інших фреймворках. Що стосується недоліків Python, то це що вона є мовою інтерпретатора, що робить Python повільною. Якщо швидкість або керування пам'яттю важливі для проекту, то краще знайти рішення в іншій мові програмування [21].

Ruby — це інтерпретована, високорівнева мова програмування загального призначення, яка підтримує декілька парадигм програмування [22]. Головним причиною популярності Ruby є те, що вона дуже дружня та легка для читання мова програмування, яка доступна для всіх рівнів навичок. Зокрема, Ruby on Rails був одним із перших фреймворків у своєму роді, який пропонував чудові рішення, які можуть дати результати за рекордно короткий термін. Ruby on Rails встановив стандарт розробки, який існує і донині. Що стосується його недоліків, то він дуже інтенсивний для роботи процесора та оперативної пам'яті, і хоча з кожним його оновленням стає все краще, але він все ще важкий у порівнянні зі своїми конкурентами. З іншого боку, Ruby on Rails так само, як і Django, значною мірою покладається на шаблони та стандарти через його дуже обмежену гнучкість [23].

Java, в свою чергу, є широко використовуваною мовою для веб-розробки, особливо на стороні сервера. Веб-програми Java — це розподілені програми, які працюють в Інтернеті. Веб-розробка на Java дозволяє нам створювати динамічні веб-сторінки, де користувачі можуть взаємодіяти з інтерфейсом. Перевагами Java є:

— незалежність від платформи;

— високий рівень безпеки;
— великий набір API;
— автоматизація процесів;
— підтримка багатопотоковості;
— легке масштабування проекту;
— підтримку об'єктно-орієнтованого підходу програмування.

Також, що варто зазначити, існують різні способи створення динамічних веб-сторінок на Java. Платформа Java EE (Enterprise Edition) надає розробникам різні технології Java для веб-розробки. Java EE надає також такі послуги, як розподілені обчислення, веб-сервіси тощо. Програми можна розробляти на Java без використання будь-якої додаткової мови сценаріїв.

Хоча Java має великий ряд переваг, але головним її недоліком є складність цієї мови, яка вимагає багато досвіду для вправного використання.

Таким чином проаналізувавши всі вище згадані мови програмування, найкраще підходить Java, оскільки вона володіє об’єктно-орієнтованим підходом, зі всіма її плюсами, великим переліком технологій, що дозволяє легко під’єднуватись до баз даних та маніпулювати її даними, що є досить великою перевагою, оскільки веб-застосунок, що розробляється, потребуватиме постійне звертання до інформації з бази даних та тому подібне.

2.3.2 Вибір фреймворку

Використовуючи мову програмування Java ми отримуємо перелік технологій, що доступні для розробки веб-застосунків та фреймворки, які заточені на цю задачу.

Java ідеально підходить для розробки великих веб-додатків завдяки своїй здатності взаємодіяти з великою кількістю систем. До таких послуг, як однорангові веб-сервіси, підключення до бази даних і серверні послуги, також можна отримати доступ за допомогою веб-розробки Java.

Нижче наведено деякі з технологій веб-розробки на мові Java:

— Servlet API;

— Java Database Connectivity (JDBC);

— Java Persistence API;

— Thymeleaf.

Розглянемо їх більш детальніше.

Servlet API збільшує можливості серверів, які використовуються для розміщення програм. Веб-додатки, розроблені за допомогою сервлету на Java, дотримуються моделі запит-відповідь. Сервлет має життєвий цикл, починаючи від ініціалізації до збирання сміття garbage колектором.

JDBC містить методи та запити для доступу до бази даних. Клієнти можуть оновлювати будь-яку інформацію в базі даних через веб-додатки, які містять драйвери JDBC.

Клієнти можуть підключатися до бази даних через програми, створені через JDBC API, і можуть оновлювати, видаляти, зберігати та отримувати доступ до даних, що дозволяє спроектувати додаток, що підтримує всі основні функції управління даними. Такий додаток називають CRUD-додатком. JDBC здатний читати будь-яку базу даних і автоматично створює XML-формат даних з бази даних.

Java Persistence API (JPA) використовує об’єктно-реляційне відображення для підключення об’єктно-орієнтованої моделі до бази даних. Реляційними даними в програмах Java можна легко керувати за допомогою Java Persistence [24]. Це допомагає постійно зберігати або витягувати велику кількість даних у або з бази даних.

Для взаємодії з базою даних не потрібно використовувати багато коду, власних фреймворків тощо, JPA надає прості засоби зв’язку з базою даних за допомогою об’єктно-реляційного підходу. JPA — це набір ефективних класів і методів, які можуть підключити розробника до бази даних. Однією з найпопулярніших реалізацій JPA є Hibernate.

В свою чергу, Thymeleaf — це сучасна серверна технологія Java для обробки та створення HTML, XML, JavaScript, CSS та тексту, який легко

дозволяє створювати динамічні веб-сторінки. Це робить розробку дуже швидкою в порівнянні з іншими популярними шаблонами.

Для реалізації системи управління взаємодії з клієнтом обрано найбільш популярний Java-фреймворк Spring. Spring фреймворк — це платформа додатків з відкритим кодом. Він призначений для спрощення проектування додатків за рахунок автоматичної обробки низькорівневих функцій. Це дозволяє розробнику сконцентруватися на бізнес-логікі [25].

Розглянемо основні переваги використання Spring:

— використання простих об’єктів Java (POJO);
— гнучкість;
— легке зв'язування об'єктів;
— великий рівень абстракції;
— відокремлена конфігурація;
— просте тестування;
— безпека.

Випускаються регулярні оновлення, які забезпечують безпеку даних і програм. Розробник може зробити власну програму безпечною, використовуючи фреймворк Spring Security. Він надає стандартні схеми безпеки та забезпечує надійне рішення, яке є безпечним за замовчуванням [25].

Надає гнучкі бібліотеки, яким довіряють розробники з усього світу. Для параметрів конфігурації розробник може вибрати анотації, XML конфігурацію, або простий Java-код. Функції IoC (інверсії контролю) і DI (введення залежностей) створюють основу для широкого набору функцій і функціональних можливостей. Це досить сильно спрощує роботу;

Spring також підтримує розробку веб-застосунків відповідно шаблону MVC (Model-View-Controller) (рисунок 2.1).

Даний шаблон складається з трьох основних компонентів: Model, View, Controller.

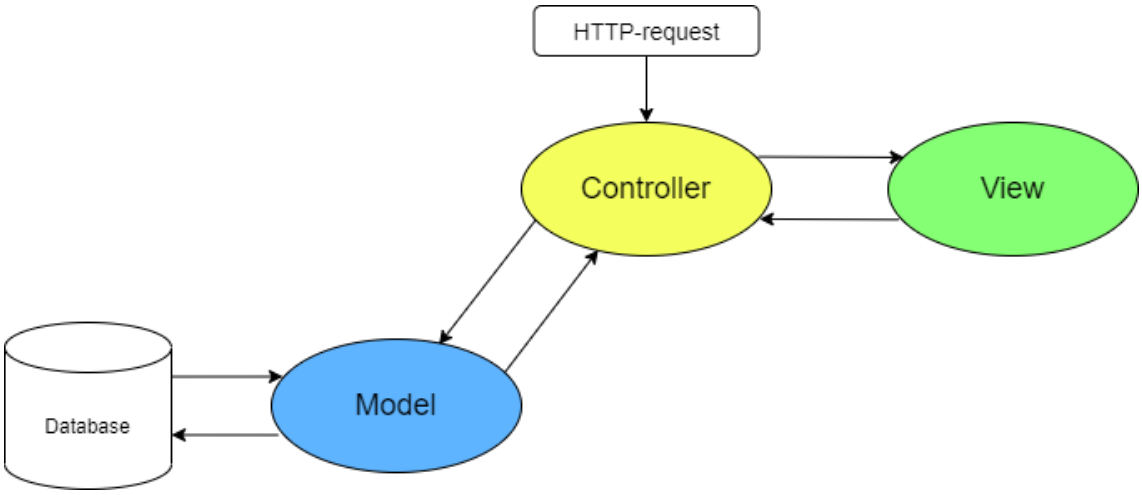


Рисунок 2.1 — Схема MVC-архітектури

Model (Модель) — представляє бізнес-рівень програми. View (Представлення) — використовується для представлення даних моделі користувачеві. Controller (Контролер) — керую потоком даних між моделлю та представленням.

У контексті програмування на Java модель складається з простих класів Java, View відображає дані, а контролер складається із сервлетів. Це поділ призводить до того, що запити користувачів обробляються таким чином:

— браузер на клієнті надсилає запит на сторінку контролеру, що присутній на сервері;

— контролер виконує дію виклику моделі, тим самим одержуючи дані, необхідні йому у відповідь на запит;

— контролер передає отримані дані у представлення;

— представлення надсилається назад клієнту для відображення браузером.

Поділ програмного додатка на ці три окремі компоненти є гарною ідеєю з ряду причин, а саме: декілька розробників можуть працювати з трьома шарами (модель, перегляд і контролер) одночасно; пропонується краща масштабованість; декілька компонентів мають низьку залежність один від одного, що запобігає виникнення помилок між ними.

Таким чином ми обрали технології, які нам підходять для розробки веб-застосунку та фреймворк, який забезпечить нас відповідними інструментами.

2.3.3 Вибір середовища розробки

Коли ми визначились із мовою програмування, технологіями та інструментами, якими ми будемо користуватися у ході розробки веб-застосунку, потрібно обрати у якому саме середовищі розробки ми будемо це виконувати. Для роботи з Java потрібно лише пакет JDK (Java Development Kit), що дозволить нам розробляти програми на Java, але без відповідного середовище розробки це буде доволі важко зробити. Коли доходить до вибору найкращої IDE для Java зазвичай обирають Eclipse, або Intellij.

IntelliJ — це зручна для початківців IDE, яка пропонує готову підтримку Java, Groovy, Kotlin і XML. Він також забезпечує підтримку інших мов програмування за допомогою плагінів. Eclipse пропонує готову підтримку лише для Java, але підтримує більше мов програмування, ніж IntelliJ IDEA, завдяки своїй розгалуженій системі плагінів, що робить його кращим вибором для досвідчених розробників.

Eclipse — популярна IDE для розробки Java, спочатку створена IBM, але тепер розроблена Eclipse Foundation. Вона написана на C і Java. Вона доступна на таких платформах як Linux, Windows і macOS. Eclipse також є у представленні веб-інтегрованої IDE — через Dirigible, Eclipse Che, Orion і Theia, які дозволяє розробникам працювати так як їм зручно. Це великі проекти, який містить широкий спектр функціональності, розроблених для певних цілей. Наприклад, Eclipse Dirigible — це платформа для розробки бізнес-додатків, в той час коли Eclipse Orion дозволяє розробникам виконувати веб-розробку прямо зі своїх веб-браузерів [26].

Eclipse краще підходить для досвідчених розробників завдяки своєму вичерпному набору плагінів. Крім того, він вимагає низьких системних ресурсів і забезпечує високу продуктивність. Однак додавання великої кількості плагінів зробить його більш об’ємним.

Так як при розробці важливо, щоб розробник краще орієнтувався у середовищі програмування, що дозволить йому думати лише про бізнес-логіку проекту, оптимальним виробом буде Intellij IDEA, оскільки вона має інтуїтивно зрозуміліший інтерфейс ніж Eclipse. Також IntelliJ має краще автоматичне генерування коду та вбудовану систему контролю версій, що обов'язково знадобиться при розробці великого проекту.

2.4 Вибір технологій для створення клієнтської частини застосунку

Розглянемо клієнтську частину застосунку, або front-end частину, з якою користувач може взаємодіяти і контактувати напряму із застосунком. У front-end входить відображення функціональних завдань призначеного для користувача інтерфейсу, що виконуються на стороні клієнта, а також обробка запитів користувачів. По суті, front-end – це все те, що бачить користувач при відкритті веб-сторінки.

Для реалізації інтерфейсу користувача буде обрано найпопулярніші технології у веб-розробці. Тому у подальшій роботі використовуватиметься HTML, CSS та фреймворк Bootstrap.

HTML, або мова гіпертекстової розмітки, дозволяє користувачам веб-ресурсів створювати та структурувати розділи, абзаци та посилання за допомогою елементів, тегів та атрибутів. HTML має багато варіантів використання, а саме:

— веб-розробка;

— інтернет навігація;

— веб-документація.

Загальну структуру html-документу можемо спостерігати на рисунку 2.2.

Після того, як представлено загальну структуру html-файлу, його варто стилізувати. CSS, або каскадна таблиця стилів — це проста мова дизайну, призначена для спрощення процесу створення презентаційних веб-сторінок. Структуру CSS зображено на рисунку 2.3.

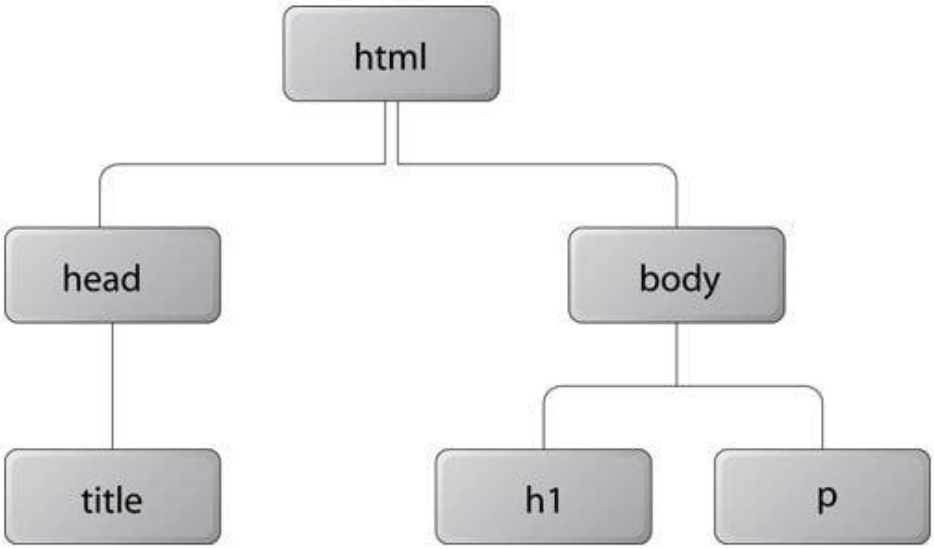


Рисунок 2.2 — Структура html-документа

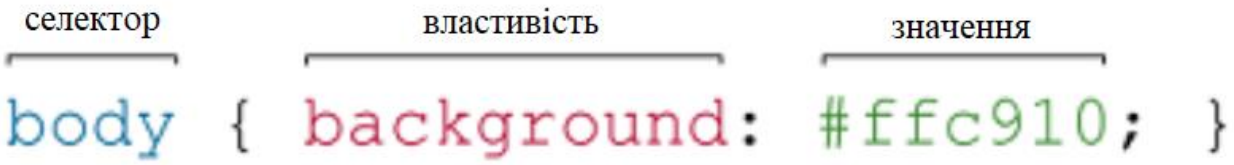


Рисунок 2.3 — Структура CSS

Використовуючи CSS, ми можете керувати кольором тексту, стилем шрифтів, інтервалом між абзацами, розміром і розташуванням стовпців, фоновими зображеннями або кольорами, дизайном макета, варіантами відображення для різних пристроїв і розмірів екрана. а також ряд інших ефектів. CSS легкий у вивченні, зрозумілий у використанні, але він забезпечує потужний контроль над поданням HTML-документа. Ось деякі з переваг CSS:

— повторне використання коду;
— швидке завантажування сторінки;
— просте обслуговування;
— краще стилі порівнянно з HTML;
— сумісність з різними пристроями.

Також для реалізації клієнтської частини використовується популярний фреймворк Bootstrap. Bootstrap — це безкоштовна фреймворк розробки інтерфейсу з відкритим вихідним кодом для створення веб-сайтів і веб-

програм. Фреймворк Bootstrap побудований на HTML, CSS і JavaScript (JS) для полегшення розробки адаптивних сайтів і програм, орієнтованих на мобільні пристрої [27].

Адаптивний дизайн дає змогу веб-сторінці, або додатку визначати розмір і орієнтацію екрана відвідувача та автоматично адаптувати відображення відповідно до цього. Підхід на основі мобільних пристроїв передбачає, що смартфони, планшети та мобільні додатки для конкретних завдань є основними інструментами співробітників для виконання роботи та відповідає вимогам цих технологій у дизайні.

Bootstrap включає в себе компоненти інтерфейсу користувача, макети та інструменти JS разом із платформою для реалізації. Програмне забезпечення доступне попередньо скомпільованим, або у вигляді вихідного коду. Версія вихідного коду використовує препроцесор Less CSS, що дозволяє автоматизувати верстку сторінки.

Версія вихідного коду містить вихідний код стилів, написаний на Less (або Sass), весь JavaScript та супровідну документацію. Це дозволяє дизайнерам і розробникам змінювати та налаштовувати за своїм бажанням усі надані стилі та створювати власну версію Bootstrap [28]. Але якщо розробник не володіє знаннями з Less (або Sass), тоді передбачений попередньо скомпільований CSS.

Таким чиною сумісність html, css коду та використання фреймворку Bootstrap визначає графічний інтерфейс через який користувач взаємодіє із застосунком.

3 РОЗРОБКА ВЕБ-ЗАСТОСУНКУ

3.1 Загальна структура веб-застосунку

Відповідно до потреб користувачів структура застосунку оцінювання навчального процесу складається з 7 головних сторінок (рисунок 3.1), які мають різну функціональність для користувачів з різними ролями (викладачем, або студентом).

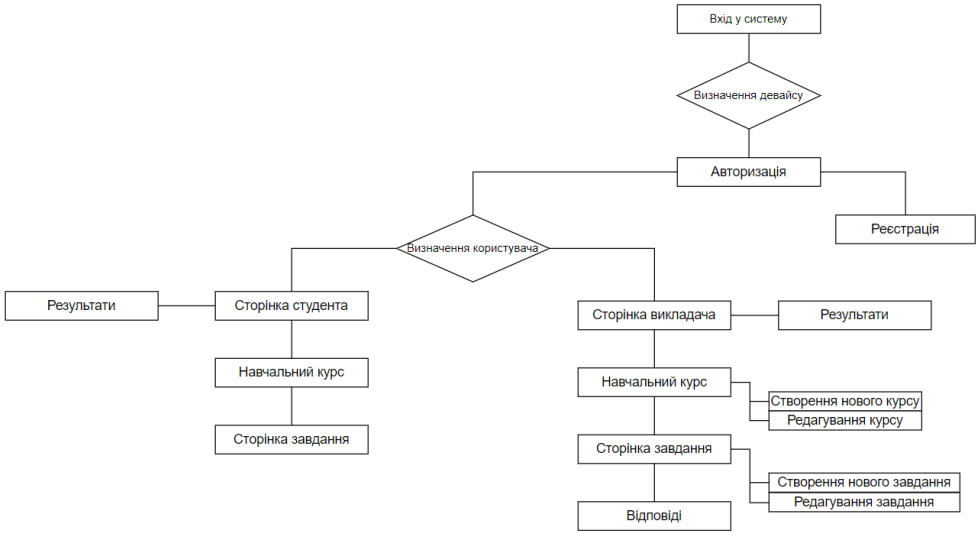


Рисунок 3.1 — Структура веб-застосунку

Початок роботи застосунку стартує з етапу визначення пристрою з якого користувач заходить у систему. Реалізація даного етапу виконується за допомогою фреймворку Bootstrap, що аналізує ширину дисплею пристрою та підлаштовується під його розмір.

Перша сторінка системи — АВТОРИЗАЦІЯ. Сторінка авторизації дозволяє увійти у власний обліковий запис системи оцінювання шляхом вводу особистих даних, а саме за допомогою адреси електронної пошти та паролю. У разі якщо користувач не володіє обліковим записом у системі, то у нього є можливість перейти на сторінку реєстрації.

Друга сторінка — РЕЄСТРАЦІЯ, де користувач зможе заповнити потрібні поля для створення облікового запису. Дана сторінка, як і сторінка авторизації містить валідацію полів вводу, тобто передбачається перевірка

даних, що вводить користувач. У разі вводу некоректних даних, буде оновлено сторінку та повідомлено користувачу про помилку, яка виникла у ході авторизації, або реєстрації. Після того, як користувач буде зареєстрований його перенаправить на сторінки авторизації для входу у систему.

Після того, як користувач введе коректні дані для входу в систему, буде визначено, якою роллю володіє користувач. Виходячи із значення «ролі» користувача буде визначено на яку саме сторінку перенаправити користувача. Якщо користувач володіє роллю «Викладач» його буде перенаправлено на сторінку викладача, якщо ж він є «Студентом» тоді на сторінку студента.

Третя сторінка — СТОРІНКА КОРИСТУВАЧА (викладача, або ж студента). Сторінка користувача залежить від ролі, яку обрам користувач при реєстрації облікового запису. Якщо користувач з роллю «Викладач», то на його особистій сторінці він має право на створювання нових навчальних курсів та перегляд курсів, що вже створенні ним попередньо. Якщо користувач є «Студентом», то для нього є можливість переглянути курси до яких він приєднаний та знайти потрібний навчальний курс за допомогою пошукової стрічки «Search», яка знайде збіги за введеною назвою курсу та відобразить їх.

Наступні сторінки будуть окремими для кожної ролі, якою може володіти користувач. Визначення на яку саме версію сторінки переадресовувати користувача буде базуватись на його ролі, яка визначається під час входу на сторінку користувача.

Четверта сторінка — НАВЧАЛЬНИЙ КУРС. Сторінка навчального курсу не сильно відрізняється для користувача з роллю «Студент» та «Викладач». В обох випадках користувачі бачать перелік завдань, які прив’язанні до курсу, і які можна переглянути натиснувши на назву завдання та можливість перегляду результатів. Також на даній сторінці для «викладача» відкрита можливість створювати нові та змінювати вже існуючі завдання курсу. Відповідні дії (створення та редагування) доступні на окремих сторінках, які мають потрібну для цього функціональність.

П’ята сторінка — РЕЗУЛЬТАТИ. Сторінка результатів дещо відрізняється для «студента» та «викладача», з різницею в тому, що «викладач» бачитиме результати всіх студентів, а «студент» лише свої власні.

Шоста сторінка — ЗАВДАННЯ. Сторінка завдання містить його назву та опис. Викладач може прикріпити потрібні для нього матеріалі, що відобразяться у комірці «Useful files» та які можна буде завантажити натиснувши на назву файлу. Студент також може завантажити файли, які представляють рішення відповідного завдання.

Сьома сторінка — ВІДПОВІДІ. Дана сторінка доступна лише для користувача з роллю «Викладач» і яка доступна з сторінки «Завдання». Вона дозволяє переглядати ресурси, які надали студенти для вирішення завдання. Для зручного перегляду вся інформація зберігається у вигляді таблиці, яка складається з 4 стовбців, а саме стовбці: «Student» — зберігає прізвище та ім’я студента; «Files» — файли, що прикріпив студент для вирішення завдання; «Mark» — вказує на те, яка оцінка стоїть у студента на даних момент; «Change mark» — представляє з себе поле вводу, яке дозволить змінити оцінку студента та кнопку «Confirm», що надсилає системі запит на зміну оцінки та заносить оновлену інформацію у базу даних. На рисунку 3.2 зображено вигляд даної таблиці.

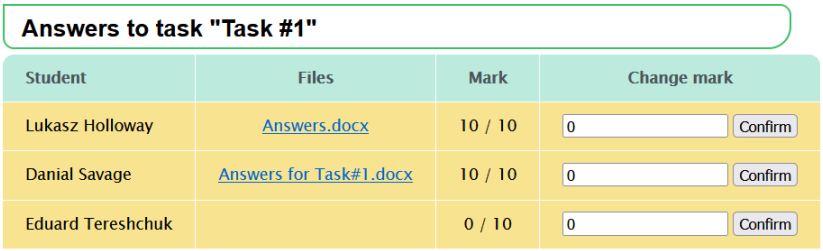


Рисунок 3.2 — Таблиця відповідей до завдання

3.2 Структура бази даних

База даних системи оцінювання складається з таких таблиць: Users, Courses, Tasks, Answers, Courses_Has_Users. Концептуальна модель бази даних має наступний вигляд (рисунок 3.3).

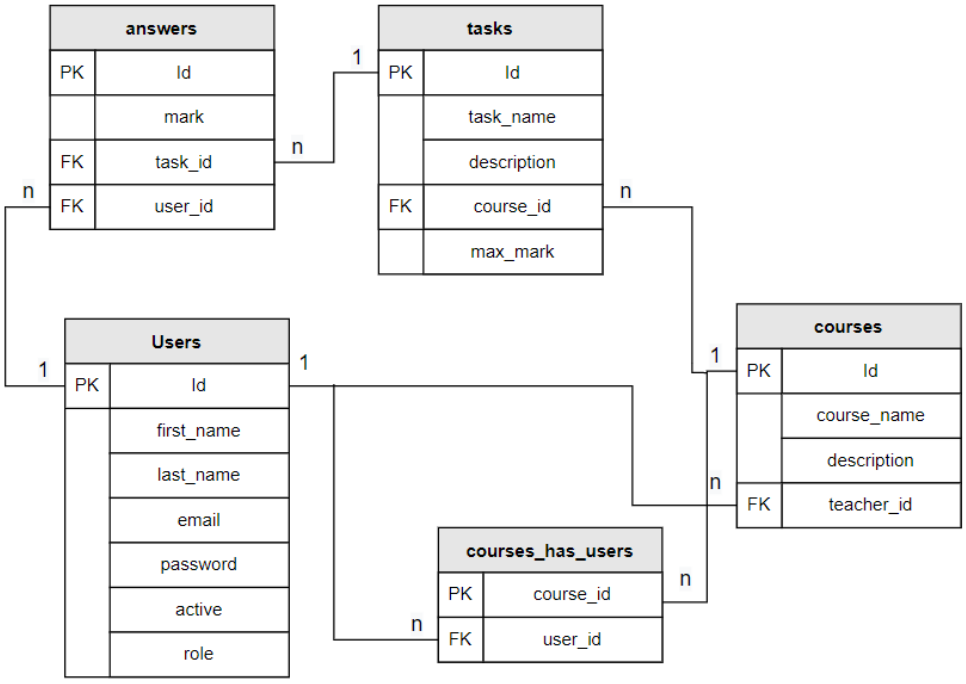


Рисунок 3.3 — Концептуальна модель бази даних

Структура таблиці «Users» представлена наступним чином (таблиця 3.1).

Таблиця 3.1 — Структура таблиці «Users»

Ім’я поля	Тип і розмір поля	Опис поля
Id	int	первинний ключ
First_name	varchar(32)	ім’я
Last_name	varchar(32)	прізвище
Email	varchar(64)	електронна пошта
Password	varchar(255)	пароль
Active	varchar(255)	тип користувача
role	varchar(32)	роль користувача

Таблиця складається з таких полів як Ідентифікатор, ім’я, прізвище, електронна пошта, пароль, тип користувача, тобто чи він є активним чи ні, та ролі, яка має два варіанти: «Студент» та «Викладач».

Структуру таблиці «Courses» представлено нижче (таблиця 3.2).

Таблиця 3.2 — Структура таблиці «Courses»

Ім’я поля	Тип і розмір поля	Опис поля
Id	int	первинний ключ
Course_name	varchar(255)	назва курсу
Description	varchar(500)	опис курсу
Teacher_id	int	ідентифікатор вчителя

Таблиця «Courses» складається з полів Ідентифікатор, Назва курсу, Опис курсу, та Ідентифікатора вчителя, що містить зв’язок з таблицею «Users», так як виражає інформацію про користувача, що створив курс.

Структуру таблиці «Courses_has_users» представлено у таблиці 3.3.

Таблиця 3.3 — Структуру таблиці «Courses_has_users»

Ім’я поля	Тип і розмір поля	Опис поля
Course_id	int	id курсу
User_id	int	id студента

Таблиця «Courses_has_users» складається з полів Ідентифікатора курсу та Ідентифікатори студента. Дана таблиця реалізовує зв’язок «багато-до-багатьох» між таблицями «Users» та «Courses». Цей зв’язок вказує на те, що один студент може бути приєднаним до багатьох курсів, а один курс може містити багато студентів. Таким чином нам потрібно було створити таблицю «Courses_has_users» для реалізації цього зв’язку.

Структуру таблиці «Tasks» представлена в таблиці 3.4.

Таблиця «Tasks» складається з полів Ідентифікатор, Назва завдання, Опис завдання, Ідентифікатор курсу та Максимальна можлива оцінка, яку можна отримати. Поля Ідентифікатор курсу має зв’язок з таблицею «Courses» по полю Id, так як виражає до якого саме курсу відноситься завдання.

Структуру таблиці «Answers» представлена в таблиці 3.5.

Таблиці 3.4 — Структура таблиці «Tasks»

Ім’я поля	**Тип і розмір поля**	**Опис поля**
Id	int	первинний ключ
Task_name	varchar(255)	назва завдання
Description	varchar(500)	опис завдання
Course_id	int	ідентифікатор курсу
Max_mark	int	максимальна оцінка

Таблиці 3.5 — Структура таблиці «Answers»

Ім’я поля	**Тип і розмір поля**	**Опис поля**
Id	int	первинний ключ
Mark	int	оцінка
Task_id	int	ідентифікатор завдання
User_id	int	ідентифікатор студента

Таблиця «Answers» складається з полів Ідентифікатор, Оцінка, Ідентифікатор завдання та ідентифікатор студента. У таблиці «Answers» також реалізований зв’язок «багато-до-багатьох», який зв’язує таблиці «Tasks» та «Users», оскільки завдання може бути вирішено багатьма студентами, а студент може мати багато завдань.

3.3 Створення веб-застосунку

Як зазначалось у попередніх розділах, для розробки обрано такий стек технологій: HTML, CSS, Bootstrap, Thymeleaf та Java.

Для початку потрібно створити сам проект. Для цього використаємо середовище програмування Intelij IDEA, виходячи з попереднього аналізу його плюсів.

Для того щоб створити проект нам потрібно натиснути кнопку «New Project», після чого відкриється вікно створення нового проекту (рисунок 3.3).

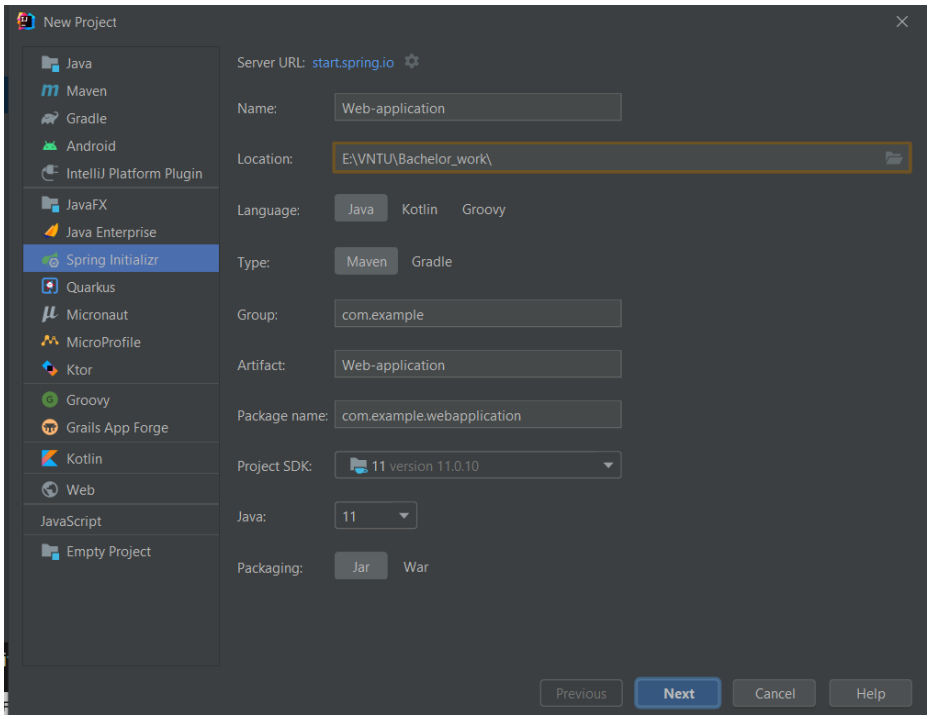


Рисунок 3.3 — Вікно створення нового проекту

У відкритому вікні потрібно ввести назву проекту, його місце розташування, мову програмування, тип системи автоматичної збірки проекту та версія Java. У нашому випадку буде використовуватись система збірки Maven. Після того як ми ввели дані, які потрібно ввести у вікні на рисунку 3.3, натискаємо кнопку «Next», що відкриє нам вікно вибору технологій, що будуть використовуватись у проекті (рисунок 3.4).

На рисунку 3.4 у колонці «Added dependencies» вказано перелік технологій, що будуть використовувати, а саме: Lombok, Spring Web, Thymeleaf, Spring Security, Spring Data JPA, MySQL Driver, Validation. Кожен елемент з цього переліку відповідає за частину функціоналу без якого наш проект не буде довершеним. Наприклад, Spring Web дозволяє створити сам веб-застосунок, Spring Security відповідає за безпеку системи та захист від

непередбаченого входу у систему шляхом авторизації користувача, а Spring Data JPA допомагає зв’язувати сутності бази даних з Java-класами.

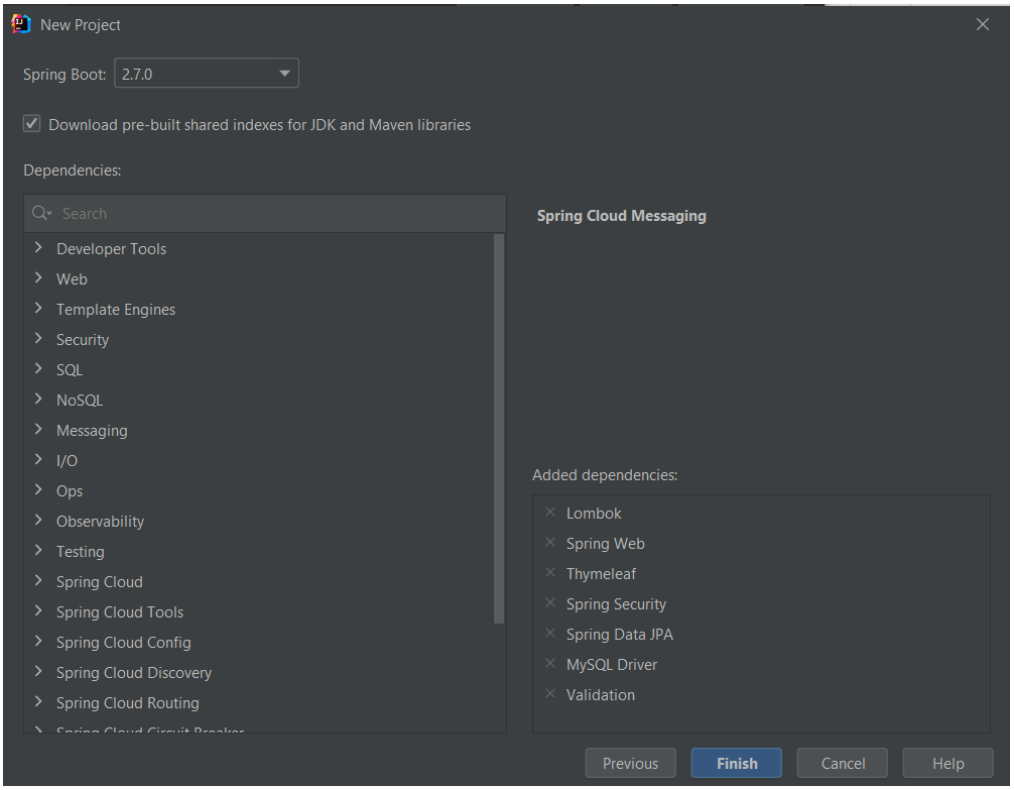


Рисунок 3.4 — Вибір технологій для проекту

Коли було обрано стек технологій, що потрібні, натиснувши кнопку «Finish» середовище «збудує» наш проект і ми зможемо його розробляти.

Оскільки використовується Maven, то головним файлом у нашому проекті є файл *pom.xml*. Цей файл відповідає за об’єктну модель проекту, тобто він створює відповідні «залежності» («dependencies») між різними модулями проекту для їхньої коректної роботи. Нижче приведено лістинг, що містить частину залежностей (лістинг 3.1).

Лістинг 3.1 — Залежності між модулями системи

```
</dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-thymeleaf</artifactId>
```

```
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-security</artifactId>
        </dependency>
    </dependencies>
```

Весь код pom.xml наведено у додатку Б.

Після конфігурування залежностей можемо приступити до написання коду. Проект поділений на кілька логічних частин, кожна з яких відповідає за власну частину функціоналу та зберігається у відповідній папці. Таким чином отримуємо структуру папок, яка зображена на рисунку 3.5.

Кожна директорія містить файли з кодом, які відповідають частині функціональності застосунку:

— bean містить код перевірки правильно завантаження файлів та роботи із сесією;

— config містить конфігурацію системи безпеки;

— controllers містить контролери, що визначають, яке “view” повернути користувачу;

— exception містить класи виключних ситуацій, що можуть виникнути у ході роботи;

— models містить класи, що представляють сутності бази даних;

— repositories представляє класи репозиторіїв, що дозволяють взаємодіяти з базою даних;

— security містить налаштування процесу авторизації;

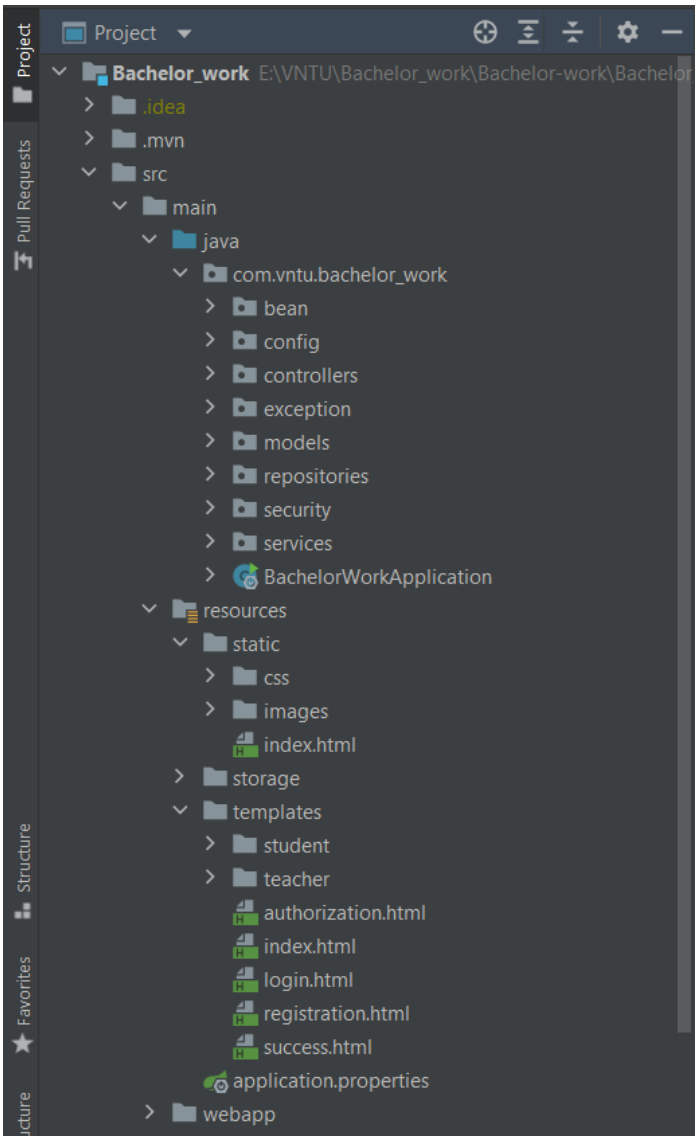
— services визначає класи сервіси для роботи з даними, які надходять з репозиторія;

— resources/static містить статичний контент додатку, файли з розширенням css та медіа файли;

— resources/templates зберігає view додатку, які будуть представленні користувачу;

— resources/storage зберігає файли, що надходять від користувача;

— resources/application.properties містить властивості, що використовуються у проекті.



Рисунку 3.5 — Структура директорій проекту

Всі властивості, що використовуються у проекті вказано на рисунку 3.6.

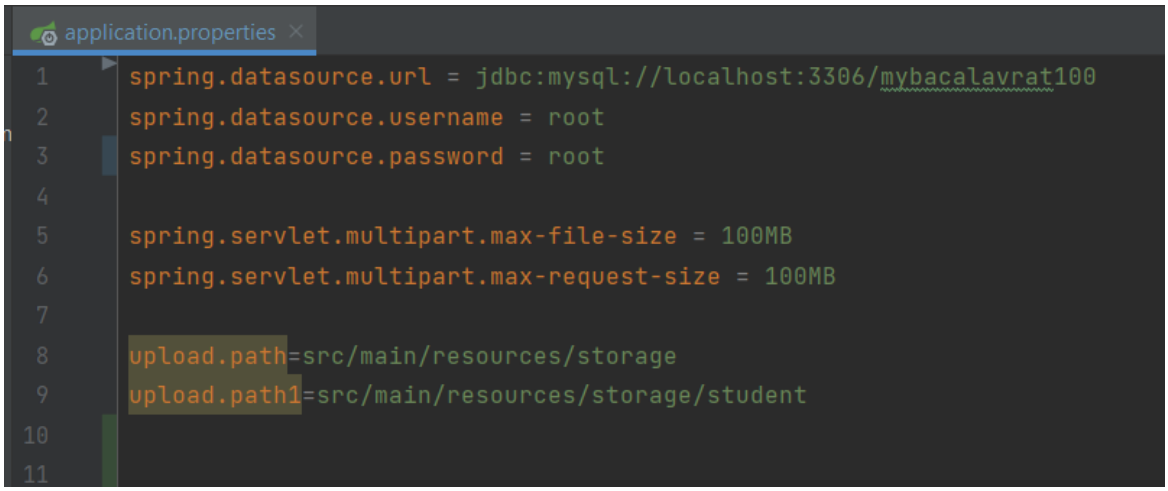


Рисунок 3.6 — Властивості проекту

Дані властивості визначають змінні, якими буде оперувати система для підключення до бази даних, шляхи для завантаження, вивантаження файлів та максимальна розмірність цих файлів.

Виконавши вище згадані етапи проектування отримуємо готовий веб-застосунок. Залишається лише протестувати клієнтську та серверну його частину.

4 ТЕСТУВАННЯ ВЕБ-ЗАСТОСУНКУ

4.1 Тестування серверної частини

4.1.1 Встановлення необхідного програмного забезпечення

Перед тим як почати безпосереднє тестування серверної частини застосунку потрібно встановити необхідне програмне забезпечення. Для цього нам знадобиться такі програми як MySQL Workbench для перегляду даних з бази даних та Postman, що дозволяє надсилати запити на сервер та отримувати від нього відповідь.

MySQL Workbench можна завантажити з офіційного сайту MySQL. Після того як програму буде встановлено з’явиться можливість працювати з базами даних. Відкривши дану програму потрібно авторизуватися. Вікно авторизації зображено на рисунку 4.1.

Рисунок 4.1 — Вікно авторизації

Після успішної авторизації відкривається файл у якому користувач може підключитися до відповідної бази даних та надіслати запит.

Postman — це API-платформа для створення, використання та тестування API. Postman спрощує кожен крок життєвого циклу API та дозволяє створювати кращі API набагато швидше [29]. Дану програму також можна завантажити із офіційного сайту Postman.

Після встановлення та відкриття програми можемо спостерігати головне вікно програми, яке зображене на рисунку 4.2.

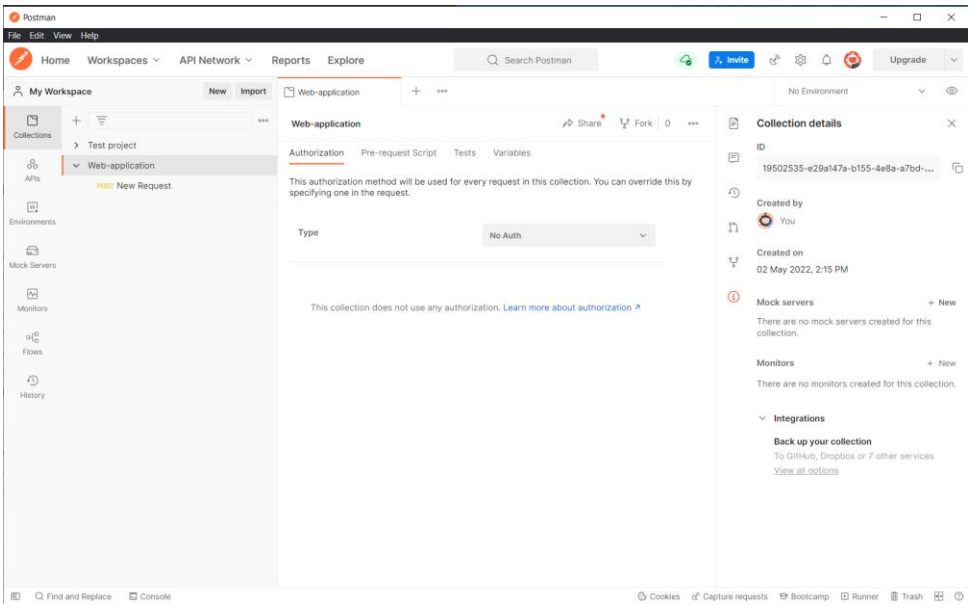


Рисунок 4.2 — Вікно програми Postman

Postman дозволяє створити так званий “Collection”, який представляє з себе папку, що буде зберігати у собі всі запити до сервера, які ми будемо надсилати. Даний підхід до тестування серверної частини значно є простішим та швидшим, ніж надсилання запитів використовуючи браузер. Postman дозволяє створити запит, який буде зберігатися у конкретному місці та який можна буде багаторазово повторювати без постійного введення даних. Це і є головною перевагою тестування API використовуючи Postman.

4.1.2 Підключення та налаштування бази даних

Встановивши програмне забезпечення для тестування потрібно налаштувати та підключити базу даних із якою буде взаємодіяти система. Дану задачу можна виконати використовуючи раніше згадане середовище Intelij.

Відкривши проект, у правому верхньому куті знаходиться вкладка «Database». Натиснувши на неї ми можемо додати потрібну базу даних до проекту. Підключення бази даних зображено на рисунку 4.3.

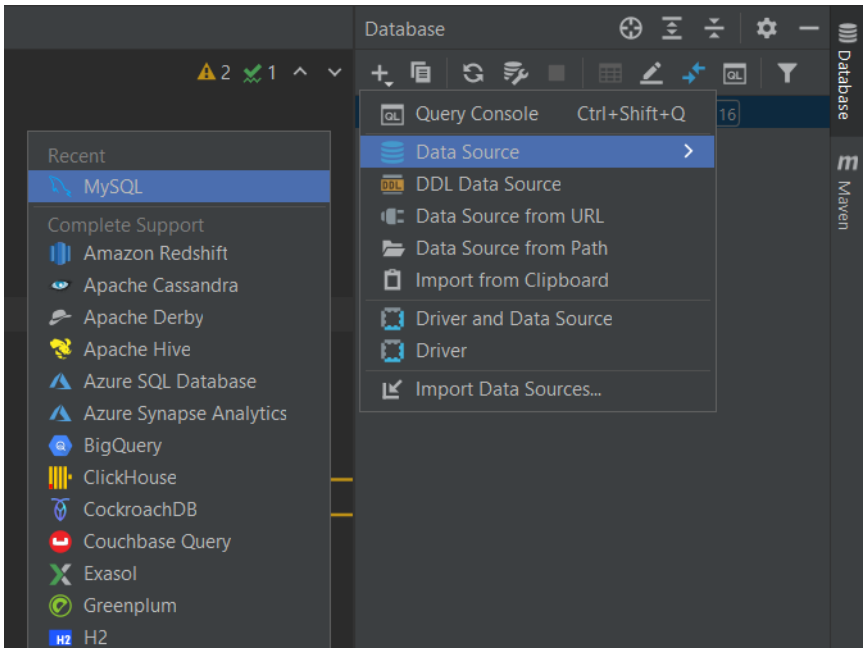


Рисунок 4.3 — Підключення бази даних

Так як використовується база даних MySQL, тому обирається відповідний пункт серед списку всіх баз даних. Після цього відкривається вікно налаштування (рисунок 4.4), у якому ми обираємо базу даних, вводимо логін та пароль, які потрібні для авторизації, щоб отримати доступ до бази даних та перевіряємо з’єднання з нею.

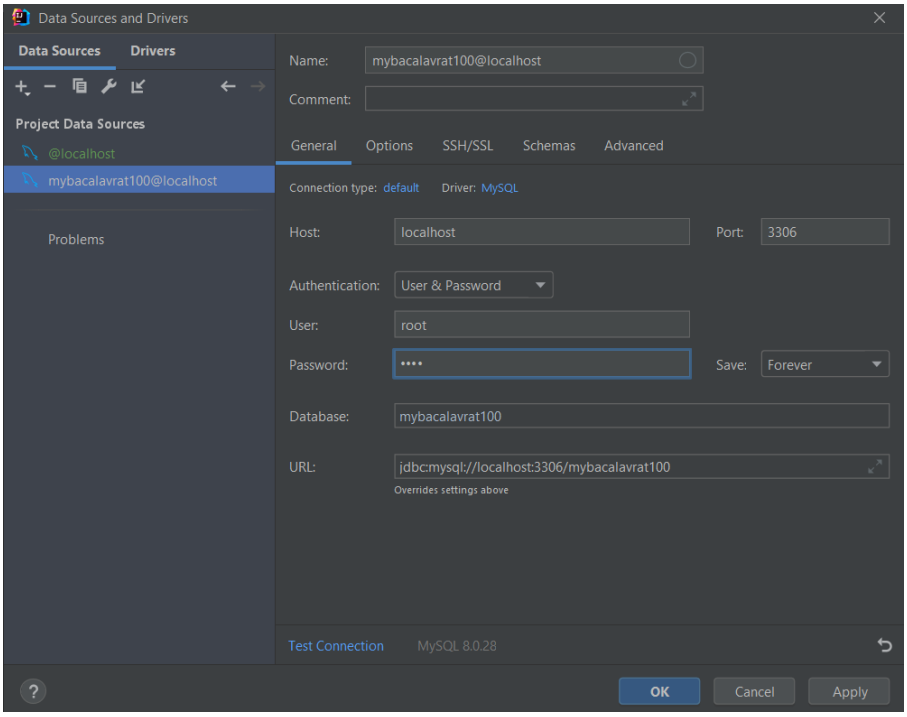


Рисунок 4.4 — Налаштування бази даних

4.1.3 Створення репозиторію та контролеру користувачів

Після підключення бази даних нам потрібно якось з нею взаємодіяти. І для цього спеціально створено технологію ORM.

ORM (Object-Relational Mapping) – це технологія програмування, що потрібна для того, щоб забезпечити перетворення даних при їх обміні між реляційною базою даних та Java. Для Java ORM фреймворків передбачений стандартний інтерфейс — JPA. І саме JPA, а точніше одна із її реалізацій, Hibernate, дозволяє нам реалізувати репозиторії для роботи з базою даних.

Репозиторії відповідають за зв’язок з базою даних для роботи з її інформацією. А контролери знадобляться для правильної обробки даних, які будуть надходити на нього та відображення правильного view користувачеві.

Інтерфейс репозиторію містить відповідну анотацію “Repository“, вказуючи, що інтерфейс забезпечує механізм зберігання, пошуку, оновлення, видалення та пошуку об’єктів. Також репозиторій розширює існуючий інтерфейс JpaRepository, який містить у собі реалізацію базових запитів до бази даних, що спрощує реалізацію функціоналу, який знадобиться для роботи з даними. Так як інтерфейс JpaRepository є узагальненим (generics), то це дозволяє нам вказати у кутових дужках два параметри, які відповідатимуть за тип об’єкта, який надходитиме з бази даних та тип значення первісного ключа у таблиці. Після виконання даних операцій можна приступити до написання самих запитів. Приклад написання даного запиту наведено у лістингу 4.1

Лістинг 4.1 — Створення запиту до бази даних

```
@Transactional
@Modifying(clearAutomatically = true)
@Query(value = "insert users(first_name, last_name, email, `password`,
role, active) values (:#{#user.getFirstName()},
:#{#user.getLastName()},
:#{#user.getEmail()},
```

```
:#{#user.getPassword()},
:#{#user.getRole().toString()},
 :#{#user.getStatus().toString()});",
nativeQuery = true)
int addUser(@Param("user") User user);
```

Даний код застосовується для зберігання нових користувачів у базі даних. За допомогою анотації «Query» ми можемо записати сам запит, який буде застосовуватися до бази даних і підставити у цей запис дані, які надходять як параметр у метод int addUser(@Param("user") User user). Таким чином реалізовується зв’язок з базою даних. Повний лістинг реалізації репозиторіїв наведено у додатку В.

Для реалізації контролера передбачено відповідні анотації, які дозволять легко налаштувати контролер. Серед них анотації, що відповідають за надходження запитів на контролер використовуючи різні HTTP методи, такі як GET, POST, PUT, DELETE, та анотації, що відповідають за налаштування контролера. Головними анотаціями конфігурування є анотація RestController, RequestMapping та RequiredArgsConsructor, які визначають надсилання відповіді у JSON-форматі, адресу по якій буде доступні методи контролера та генерацію конструктора, призначеного для ініціалізації полів класу. Приклад реалізації контролера наведено в лістингу 4.2.

Лістинг 4.2 — Реалізація контролера

```
@Controller
@RequiredArgsConstructor
@RequestMapping("/teacher")
public class TeacherController {
   @GetMapping
   public String getUser(Model model) {
      model.addAttribute("teacher", httpSessionBean.getUser());
```

```
        model.addAttribute("courses",
          courseService.getAllTeachersCourses(httpSessionBean
          .getUser()
          .getId()));
        return "teacher/teacherPage";
    }
}
```

Повний код реалізації контролерів наведено у додатку Г.

4.1.4 Тестування роботи API

Щоб протестувати функціонал сервісу скористаємося встановленою програмою Postman, яка дозволить створити запити та формувати відповідь серверу. Протестуємо запис нового користувача у базу даних та перегляд курсів, якими володіє один із викладачів.

Для запису нового користувача створимо HTTP POST запит, обравши Auth контролер метод реєстрації користувача за адресою http://localhost:8080/auth/registration (Лістинг 4.3)

Лістинг 4.3 — Тіло запиту реєстрації користувача

```
Body:
{ "firstName": "Eduard",
   "lastName": "Tereshchuk",
   "email": "tereshchuk@gmail.com",
   "password": "12345678",
   "role": "STUDENT",
   "status": "ACTIVE"}
```

Результат виконання запиту реєстрації користувача на рисунку 4.5.

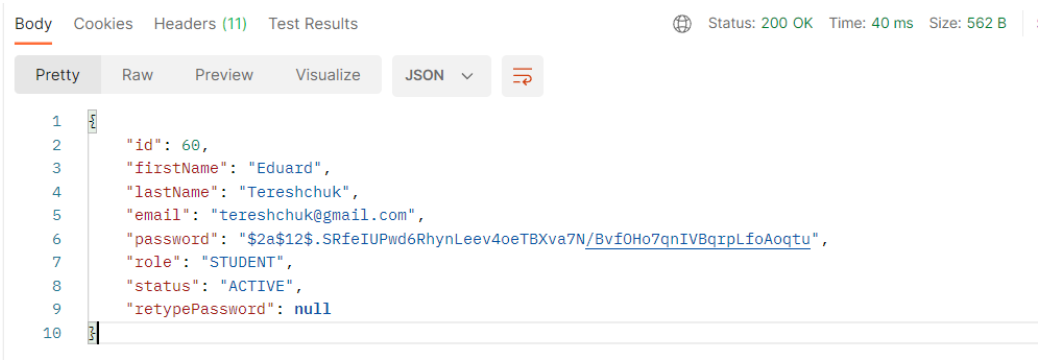


Рисунок 4.5 — Результат виконання запиту

Оскільки тіло відповіді збігається із тілом запиту, то запис користувача у базу даних пройшов успішно. Так як наша система забезпечує безпеку особистих даних, і захищає від викрадення паролів, паролі перед записом у базу даних шифруються, тому поле “password” у тілі запиту та тілі відповіді будуть відрізнятися.

Тепер перевіримо функціонал перегляду курсів викладача. Для цього створимо HTTP GET запит на Teacher контролер метод GetCourses використавши адресу http://localhost:8080/teacher/58. Результат виконання запиту зображено на рисунку 4.6

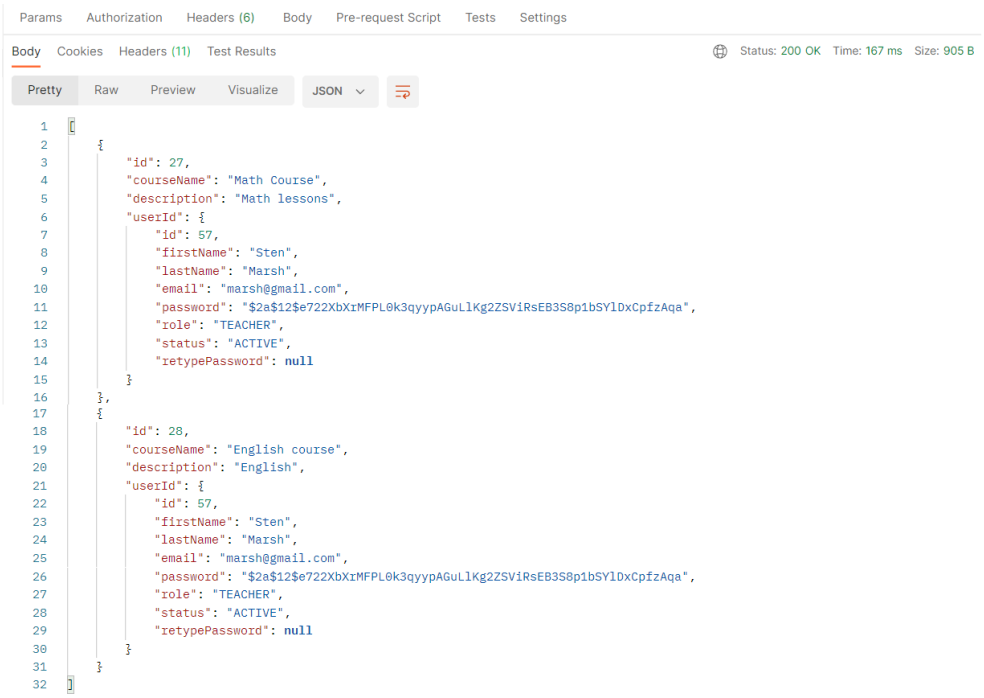


Рисунок 4.6 — Результат виконання запиту отримання курсів

Як видно з відповіді серверу було надано інформацію про перелік курсів, яким володіє викладач, час виконання запиту становить 167 мілісекунд. Таким чином було протестовано та доведено коректність роботи серверної частини додатку.

4.2 Тестування клієнтської частини

Для тестування клієнта скористаємося браузером Mozilla Firefox. Спочатку відкриємо головну сторінку програми, сторінку авторизації, для цього потрібно у адресному рядку браузера ввести домен хоста клієнта, так як застосунок хоститься на 8080 порті, то адреса буде http://localhost:8080/auth/login. Сторінка авторизації зображена на рисунку 4.7.

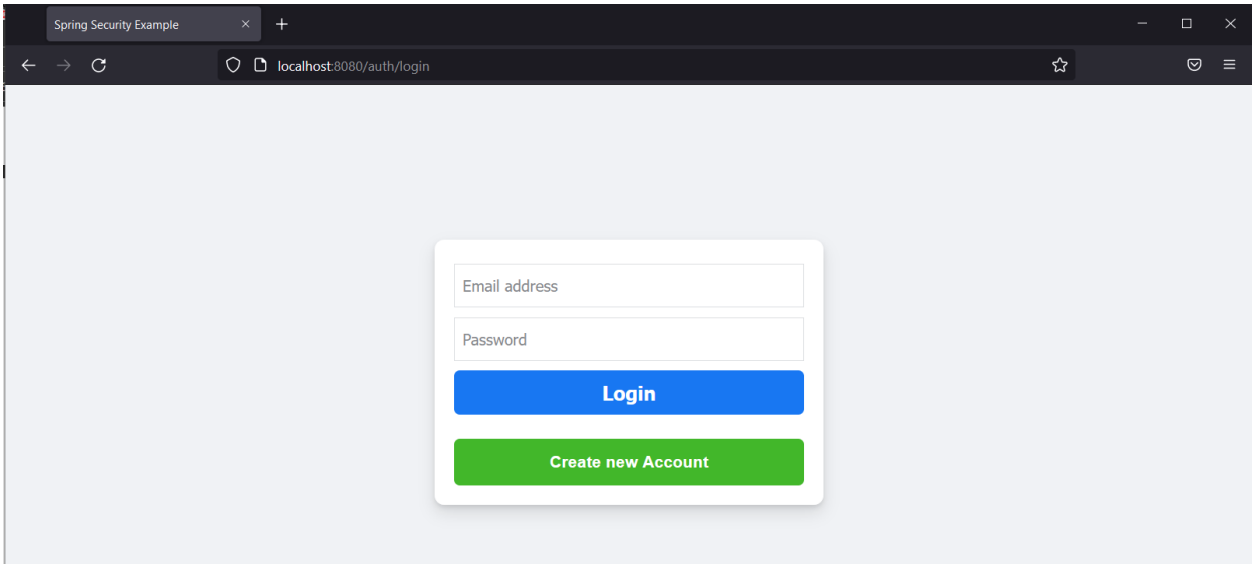


Рисунок 4.7 — Сторінка авторизації

При умові, якщо користувач не володіє власним обліковим записом у системі, то у цьому випадку передбачена кнопка «Create new Account» при натисканні на яку користувача буде перенаправлено на сторінку реєстрації (рисунок 4.8).

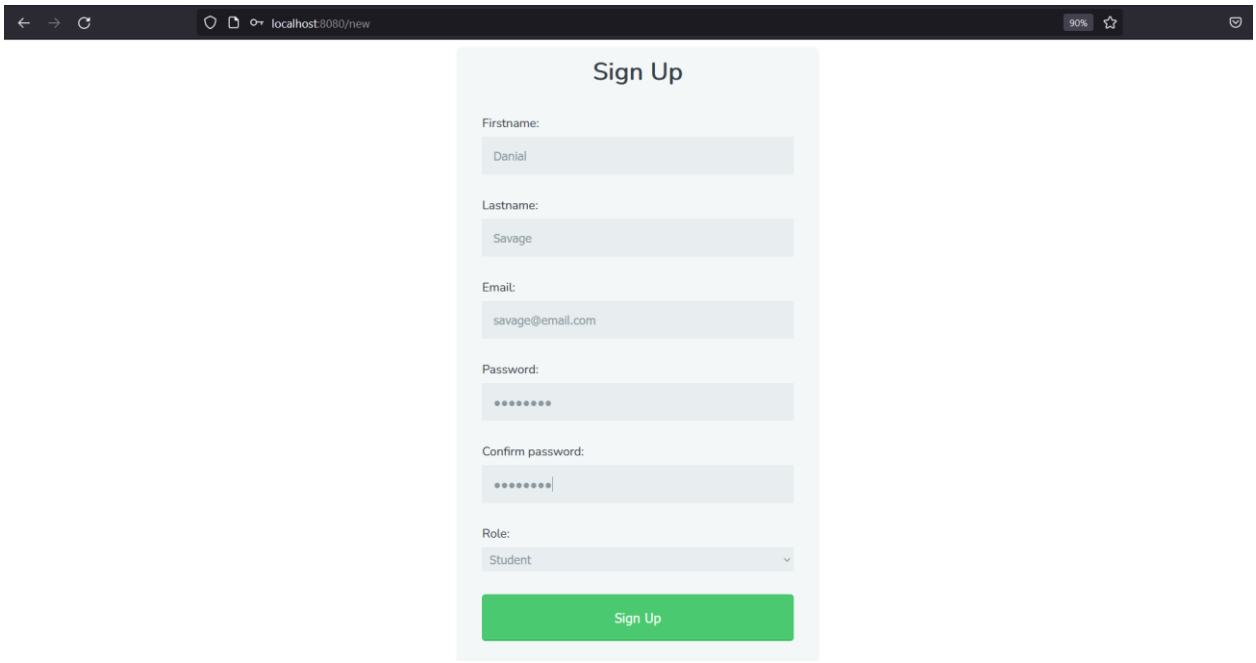


Рисунок 4.8 — Сторінка реєстрації користувача

Після реєстрації та авторизації у систему користувача переводить на його особисту сторінку. Так як зареєстрований користувач є студентом, то його переводить на сторінку студента (рисунок 4.9).

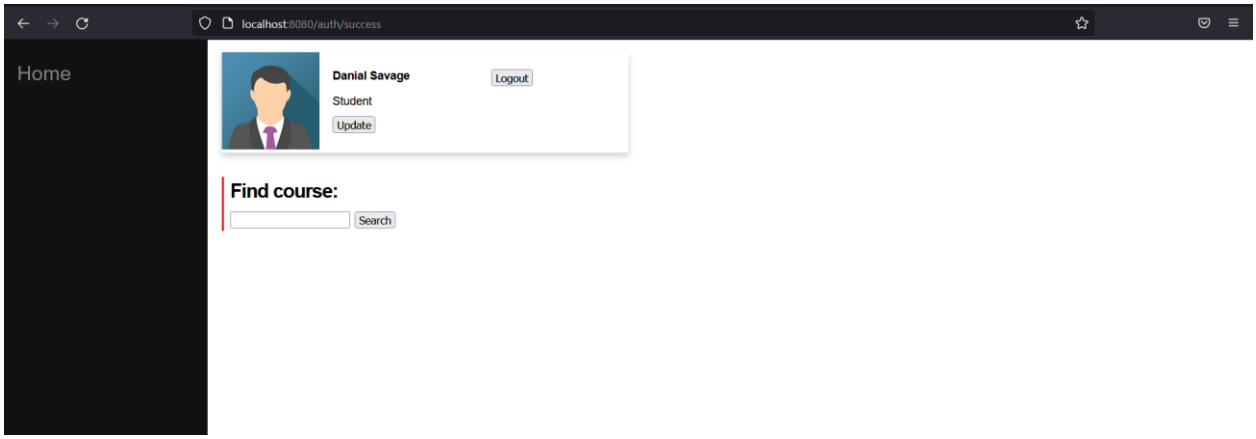


Рисунок 4.9 — Сторінка студента

Оскільки студент лише зареєструвався у системі, то у нього відсутній список курсів до яких він приєднаний. На сторінці студента присутня пошукові стрічка «Find course», за допомогою якої він може віднайти потрібний йому навчальний курс. Нехай студенту потрібно знайти курс, який містить у собі букву «е». Результат даного пошуку показано на рисунку 4.10.

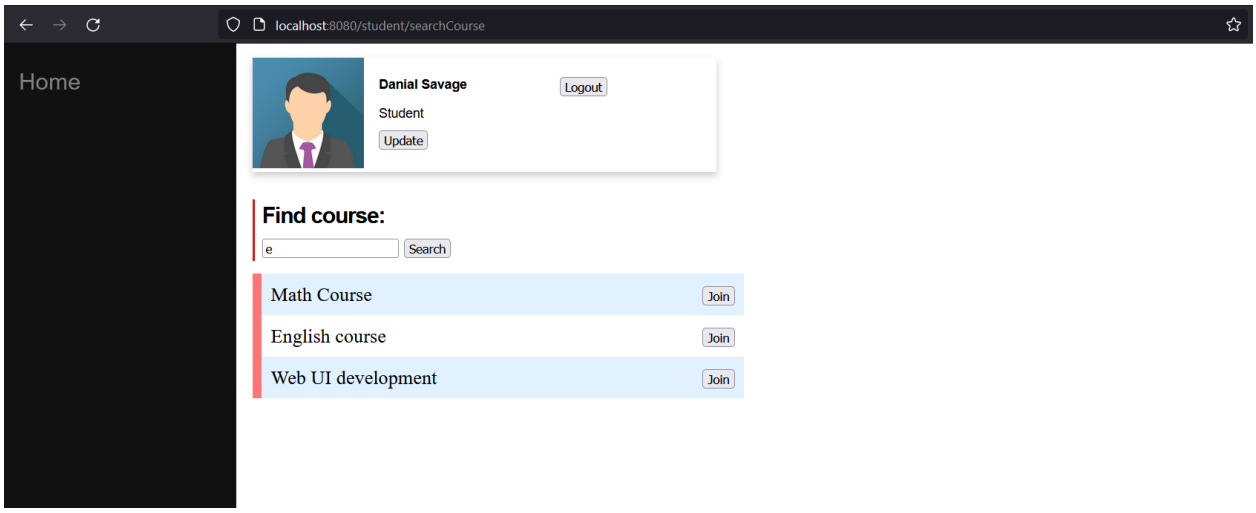


Рисунок 4.10 — Результат пошуку курсу

При знаходженні потрібного курсу, студент натискає на кнопку «Join», яка приєднує студента до відповідного курсу. У нашому випадку студент приєднався до курсу «Math Course». Після приєднання сторінка студента дещо видозмінилася (рисунок 4.11).

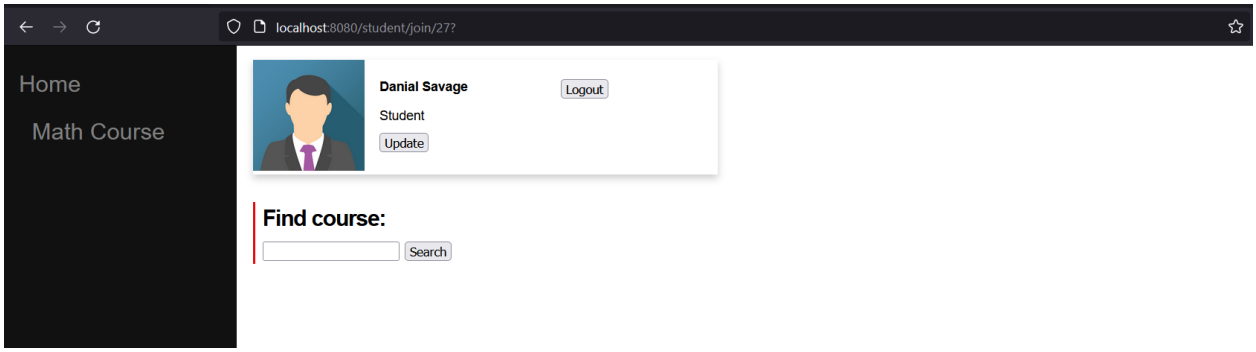


Рисунок 4.11 — Оновлена сторінка студента

Після приєднання до курсу, у навігаційному меню, що знаходиться у лівому куті екрану з’являється назва курсу до якого приєднаний користувач. При натисканні на назву курсу студента перенаправить на сторінку курсу (рисунок 4.12).

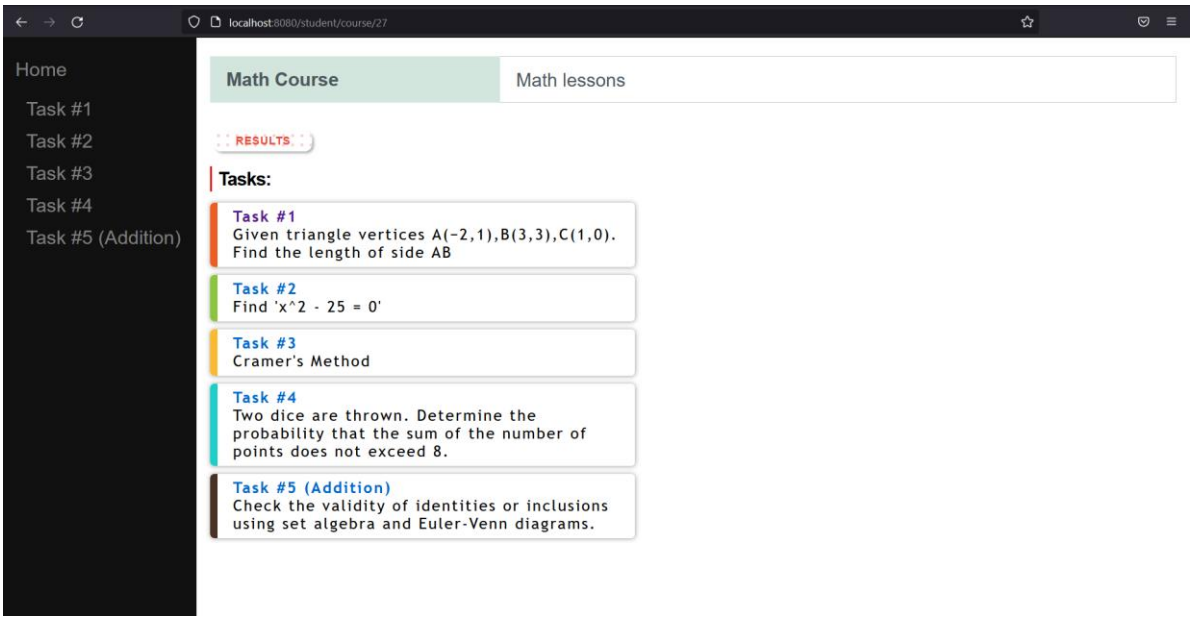


Рисунок 4.12 — Сторінка навчального курсу студента

Сторінка навчального курсу містить навігаційне меню, яке містить пункт «Home», що повертає користувача на його особисту сторінку, та перелік завдань, які прикріплені до курсу. На основній частині сторінки також знаходиться список завдань, але з більш детальним їх описом.

При натисканні на назву завдання студента перенаправить на сторінку завдання (рисунок 4.13).

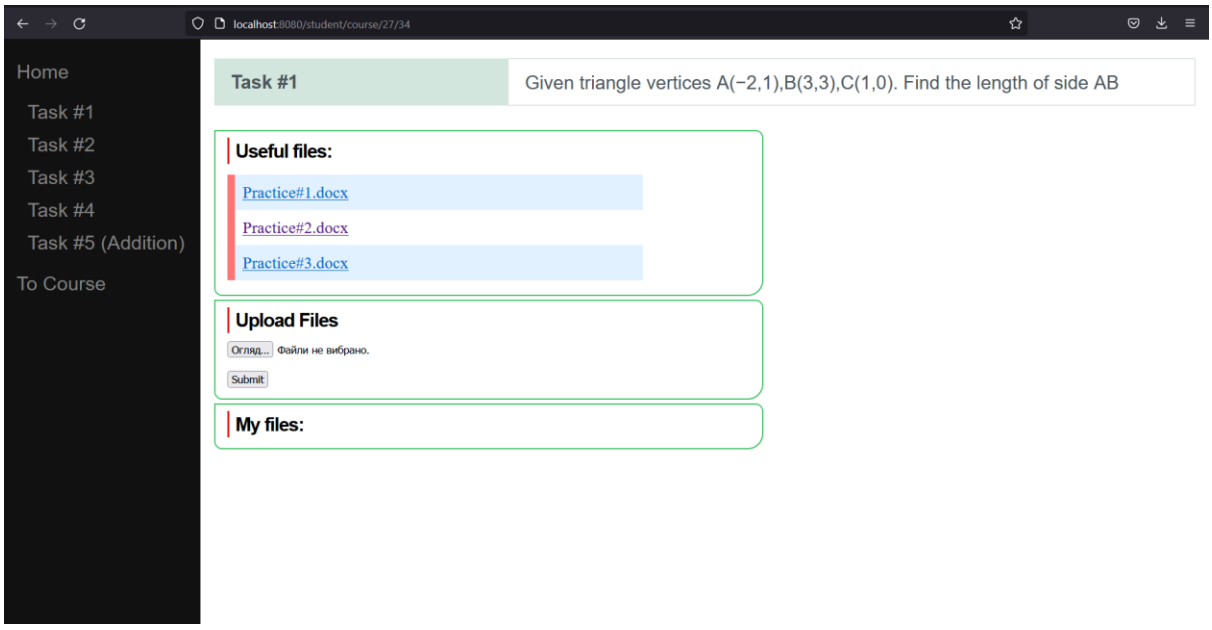


Рисунок 4.14 — Сторінка завдання

Дана сторінка містить назву та опис завдання, перелік файлів, що прикріпив викладач (Useful files), стрічку для завантаження файлів (Upload Files) та перелік файлів (My Files), які прикріпив студент, як розв'язок до завдання. При натисканні на кнопку «Огляд…» відкривається вікно вибору файлів, обравши потрібний файл натискаємо на кнопку зберегти та «Submit». Результат завантаження файлу зображено на рисунку 4.15.

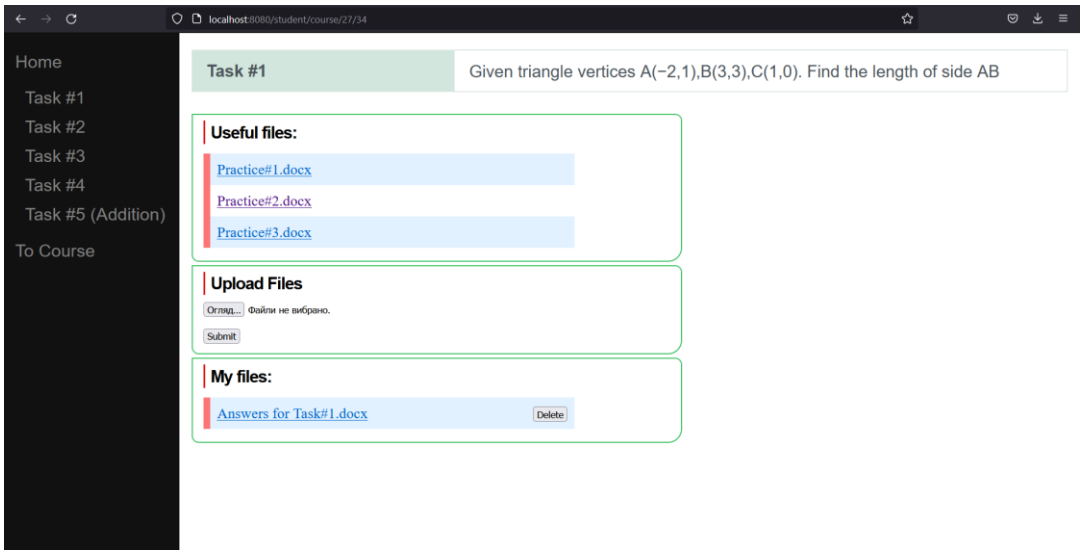


Рисунок 4.15 — Результат завантаження файлів

Після того як студент завантажив потрібні файли до завдання, викладач мусить їх переглянути та виставити оцінку. Для цього повернемося на головну сторінку за допомогою пункту у навігаційному меню «Home» та натиснемо на кнопку «Logout», щоб розлогінитися та зайти з під акаунту викладача (рисунок 4.16).

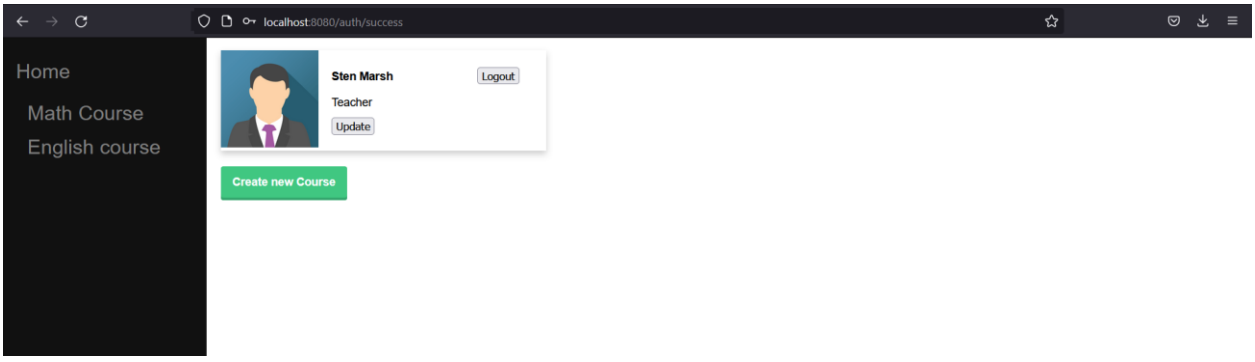


Рисунок 4.16 — Сторінка викладача

На сторінці викладача знаходиться кнопка “Create new Course” для створення нового навчального курсу та навігаційне меню з переліком курсів, що були створені ним. Так як викладачеві потрібно оцінити роботу студента, то натискаємо на назву навчального курсу, що перенесу його на сторінку курсу з переліком завдань (рисунок 4.17).

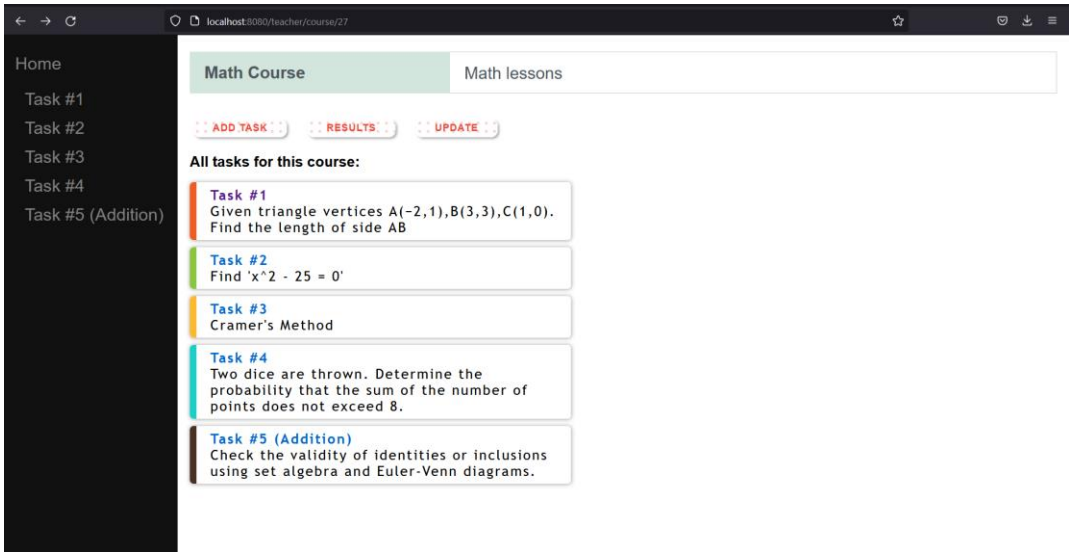


Рисунок 4.17 — Сторінка курсу викладача

Дана сторінка містить такі кнопки як «ADD TASK» — для створення нового завдання, «RESULTS» — перегляду результатів, «UPDATE» — оновлення назви та опису курсу.

Натиснувши на «Task #1» у навігаційному меню викладач потрапляє на сторінку завдання (рисунок 4.18)

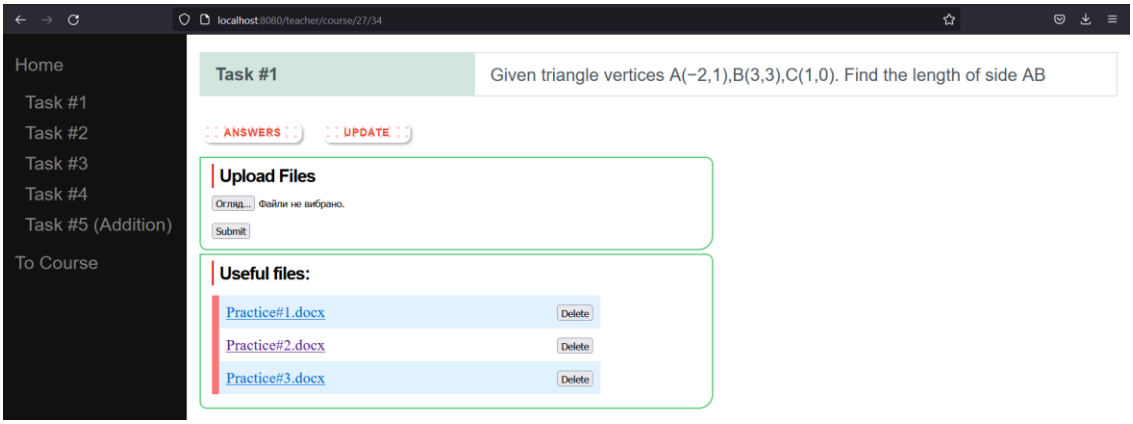


Рисунок 4.18 — Сторінка завдання викладача

Натиснувши на кнопку «ANSWERS» викладач потрапляє на сторінку відповідей, які надали студенти для вирішення задачі (рисунок 4.19).

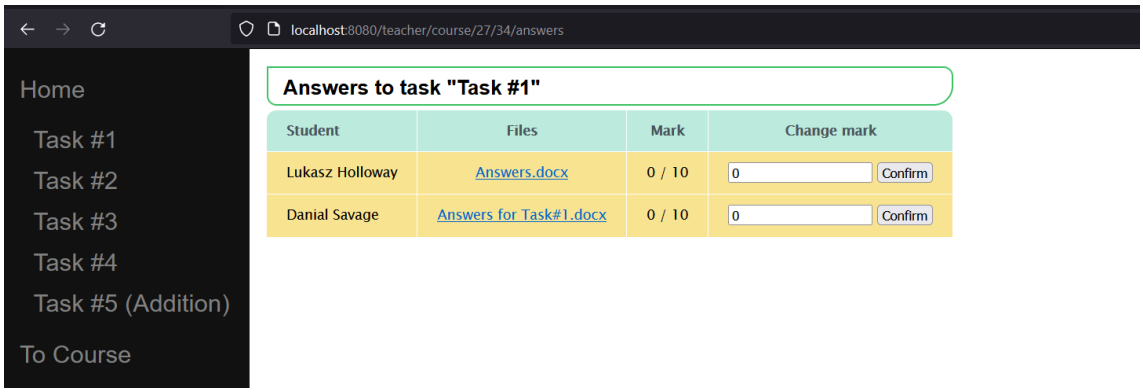


Рисунок 4.19 — Сторінка відповідей

Після того як викладач перейшов на сторінку відповідей він може переглянути файли, які прикріпили студенти, шляхом завантаження файлів на особистий пристрій та виставити відповідну оцінку у комірці «Change mark».

Після виставлення оцінок викладач може повернутися на сторінку курсу, натиснути на кнопку «RESULTS» та переглянути результати студентів до всіх завдань (рисунок 4.20).

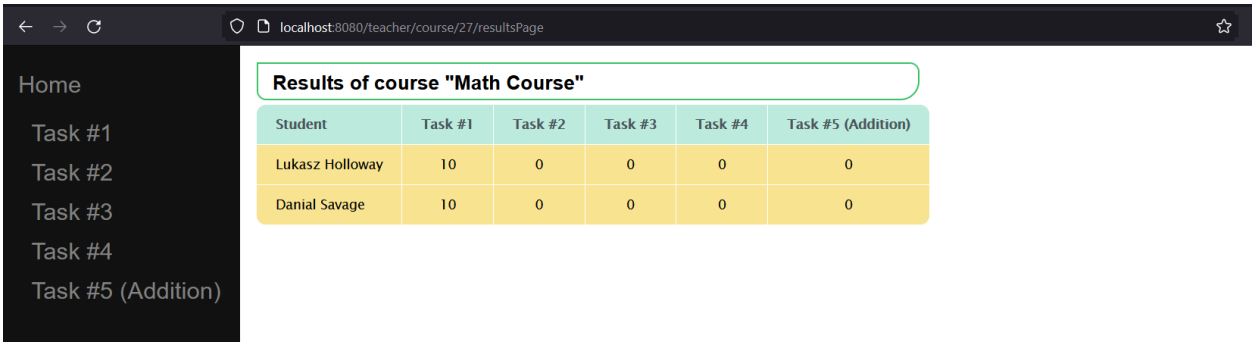


Рисунок 4.20 — Сторінка результатів

Отже функціонал клієнтської частини веб-застосунку було успішно протестований та доведено коректність його роботи.

ВИСНОВКИ

Під час виконання бакалаврської дипломної роботи було досліджено роль веб-застосунків у сучасному світі, типи веб-застосунків, доцільність їх конструювання та реалізацію.

Розглянуто підходи до проектування веб-застосунків та визначено плюси та мінуси кожного підходу. Також проаналізовано концепції баз даних і конкретні реалізації систем управління базами даних, з яких було обрано найкращий підхід для цієї розробки.

Вибрано мову програмування Java, що дозволяє реалізувати роботу застосунку незалежно від платформи, яку використовує користувач, високий рівень безпеки та легке масштабування застосунку. Також вибрано базу даних MySQL для легкого зберігання даних, швидкого доступу до них та підтримки на різних операційних системах.

Було розроблено навчальний веб-застосунок для оцінювання навчального процесу з використанням клієнт-серверної архітектури, технологій JDBC, JPA, Thymeleaf та Spring framework, включаючи систему авторизації та захисту приватної інформації користувачів. Даних програмний засіб створено в середовищі Intelij та MySQL Workbench. Структура застосунку складається з логічних елементів, що визначають операційну систему та пристрій, який використовує користувач та визначає доступ до елементів системи виходячи з «ролі», якою володіє користувач.

Після розробки було проведено тестування застосунку організації та оцінювання навчального процесу з метою перевірки коректності його роботи.

Розроблений веб-застосунок є зручним і практичним рішенням, яке не залежить від платформи, спрощує організацію навчального процесу та має потенціал для подальшого розширення функціональності завдяки чіткій структурі проекту.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Биков В.Ю., Лапінський В.В. Методологічні та методичні основи створення і використовування електронних засобів навчального призначення — №3 . — 2020

2. П. Федорук і М. Дутчак, Побудова бази знань адаптивних систем дистанційного навчання на основі фреймової та продукційної моделей представлення знань, Управляючі системи і машини №5, с.35-42, 2019

3. Терещук Е. П., Городецька О. С., Савицька Л. А. Веб-застосунок для організації та оцінювання навчального процесу, ВНТУ — 2022

4. Веб-застосунок [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Вебзастосунок

5. Розробка веб-додатків [Електронний ресурс]. Режим доступу: https://en.yeeply.com/blog/6-different-kinds-веб-app-development/#definition

6. Що таке CMS? [Електронний ресурс]. Режим доступу: https://ecommerce-platforms.com/glossary/content-management-system-cms

7. Переваги використання веб-застосунків [Електронний ресурс]. Режим доступу: https://www.geeks.ltd.uk/insights/the-benefits-of-using-веб-based-applications

8. The benefits of веб applications in todays technological era [Електронний ресурс]. Режим доступу: https://www.kcsitglobal.com/blogs/detail-blog/the-benefits-of-веб-applications-in-todays-technological-era

9. A Cloud Computing Based Learning Management Systems (LMSs) Architecture [Електронний ресурс]. Режим доступу: https://www.researchgate.net/publication/329465743_A_Cloud_Computing_Based_Learning_Management_Systems_LMSs_Architecture

10. Features of a Learning Management System [Електронний ресурс]. Режим доступу: https://www.peoplefluent.com/blog/learning/13-must-have-features-of-a-learning-management-system/

11. Google Classroom [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Google_Classroom

12. NEO LMS [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/NEO_LMS

13. The top 5 software architecture patterns [Електронний ресурс]. Режим доступу: https://techbeacon.com/app-dev-testing/top-5-software-architecture-patterns-how-make-right-choice

14. Types of Software Architecture Patterns [Електронний ресурс]. Режим доступу: https://www.geeksforgeeks.org/types-of-software-architecture-patterns/

15. Однорангова архітектура [Електронний ресурс]. Режим доступу: https://studfile.net/preview/9052707/page:4/

16. SOAP vs. REST vs. GraphQL: Difference Between Web API Services [Електронний ресурс]. Режим доступу: https://javascript.plainenglish.io/soap-vs-rest-vs-graphql-difference-between-web-api-services-461eee5d1ad7

17. SOAP (Simple Object Access Protocol) [Електронний ресурс]. Режим доступу: https://www.techtarget.com/searchapparchitecture/definition/SOAP-Simple-Object-Access-Protocol

18. Difference between MySQL and Oracle [Електронний ресурс]. Режим доступу: https://www.javatpoint.com/mysql-vs-oracle

19. PHP Advantages and Disadvantages | What is PHP Language? Merits and Demerits of PHP [Електронний ресурс]. Режим доступу: https://www.aplustopper.com/php-advantages-and-disadvantages/

20. Pros and Cons of JavaScript [Електронний ресурс]. Режим доступу: https://data-flair.training/blogs/advantages-disadvantages-javascript/

21. What Is Python Used For? A Beginner’s Guide [Електронний ресурс]. Режим доступу: https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python

22. Ruby [Електронний ресурс]. Режим доступу: https://en.wikipedia.org/wiki/Ruby_(programming_language)

23. Top 5 Languages and Frameworks for Server-Side Scripting [Електронний ресурс]. Режим доступу: https://www.bairesdev.com/blog/top-languages-server-side-scripting/

24. What is JPA? Introduction to the Jakarta Persistence API [Електронний ресурс]. Режим доступу: https://www.infoworld.com/article/3379043/what-is-jpa-introduction-to-the-java-persistence-api.html

25. Spring Framefork [Електронний ресурс]. Режим доступу: https://spring.io/projects/spring-framework

26. Intellij vs. Eclipse [Електронний ресурс]. Режим доступу: https://www.interviewbit.com/blog/intellij-vs-eclipse/

27. Bootstrap [Електронний ресурс]. Режим доступу: https://www.techtarget.com/whatis/definition/bootstrap

28. What is Bootstrap? A Short Bootstrap Tutorial on the What, Why, and How [Електронний ресурс]. Режим доступу: https://www.toptal.com/front-end/what-is-bootstrap-a-short-tutorial-on-the-what-why-and-how

29. Postman API Platform [Електронний ресурс]. Режим доступу: https://www.postman.com/

ДОДАТОК А

Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ

проф., д.т.н.. Азаров О.Д.

“28” квітня 2022 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання бакалаврської дипломної роботи

“Веб-застосунок для організації та оцінювання навчального процесу”

08-23.БДР.014.00.000 ТЗ

Науковий керівник: доцент к.т.н.

_____ Городецька О. С.

Студент групи 1КІ-18б

_____ Терещук Е. П.

м. Вінниця – 2022

1 Підстава для використання бакалаврської дипломної роботи (БДР)

1.1 Актуальність розробки полягає у спрощенні організації та оцінювання навчального процесу, шляхом розробки веб-застосунку, головним завданням якого є оптимізація навчального процесу для викладачів та студентів.

1.2 Наказ про затвердження теми бакалаврської дипломної роботи.

2 Мета і призначення БДР

2.1 Мета проекту — розробка веб-застосунку для організації та оцінювання навчального процесу, призначеного для загального користування.

2.2 Призначення розробки — виконання бакалаврського дипломного проекту із подальшим впровадженням та розвитком продукту.

3 Вихідні дані для виконання БДР

3.1 Розгляд принципів роботи веб-застосунків.

3.2 Проведення аналізу сучасних підходів до побудови веб-застосунків.

3.3 Розробка веб-застосунку та створення бази даних.

3.4 Тестування серверної та клієнтської частин.

4 Вимоги до виконання БДР

Головна вимога — розробити мультиплатформенний, легко масштабований веб-застосунок для організації та оцінювання навчального процесу.

5 Етапи БДР та очікувані результати

Етапи роботи та очікувані результати приведено в Таблиці А.1.

6 Матеріали, що подаються до захисту БДР

До захисту подаються: пояснювальна записка БДР, ілюстративні матеріали, протокол попереднього захисту БДР на кафедрі, відгук наукового

керівника, анотації до БДР українською та іноземною мовами, довідка про відповідність оформлення БДР діючим вимогам.

Таблиця А.1 — Етапи БДР

№ з/п	Назва етапів виконання комплексної бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Постановка задачі роботи	14.03.22	
2	Аналіз предметної області	15.03-18.03.22	
3	Особливості систем управління навчальним процесом	21.03-24.03.22	
4	Аналіз та обґрунтування вибору технологій для розробки застосунку організації та оцінювання навчального процесу	25.03-30.03.22	
5	Вибір архітектури сервісу	31.03-04.04.22	
6	Вибір технологій для створення клієнтської частини застосунку	05.04-.11.04.22	
7	Розробка веб-застосунку для організації та оцінювання навчального процесу	12.04-02.05.22	
8	Тестування серверної та клієнтської частини	03.05-19.05.22	
9	Підготовка матеріалів та опис розробки інформаційної системи	23.05-26.05.22	
10	Аналіз виконання роботи, висновки, додатки	27.05-31.05.22	
11	Перевірка якості виконання бакалаврського проекту та усунення недоліків	01.06 -.11.06.22	

7 Порядок контролю виконання та захисту МКР

Виконання етапів графічної та розрахункової документації БДР контролюється науковим керівником згідно зі встановленими термінами. Захист БДР відбувається на засіданні Екзаменаційної комісії, затвердженої наказом ректора.

8 Вимоги до оформлювання та порядок виконання МКР

При оформлюванні МКР використовуються:

— ДСТУ 3008 : 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302 : 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— ГОСТ 2.104-2006 «Єдина система конструкторської документації. Основні написи»;

— документами на які посилаються у вище вказаних.

Технічне завдання до виконання отримав_____ Терещук Е. П.

ДОДАТОК Б

Налаштування конфігураційного файлу “pom.xml”

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.6.7</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.vntu</groupId>
    <artifactId>Bachelor_work</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>Bachelor_work</name>
    <description>Bachelor_work</description>
    <properties>
        <java.version>11</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-rest</artifactId>
        </dependency>
        <!--    Thymeleaf    -->
        <dependency>
```

```
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
    <optional>true</optional>
</dependency>
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <scope>runtime</scope>
</dependency>
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
```

```
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.hibernate.validator</groupId>
            <artifactId>hibernate-validator</artifactId>
            <version>6.1.6.Final</version>
        </dependency>
        <!-- Security -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-security</artifactId>
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <configuration>
                    <excludes>
                        <exclude>
                            <groupId>org.projectlombok</groupId>
                            <artifactId>lombok</artifactId>
                        </exclude>
                    </excludes>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

ДОДАТОК В

Реалізація контролерів

Лістинг файлу “src/java/com/vntu/bachelor_work/AuthController.java”

```
package com.vntu.bachelor_work.controllers;
import com.vntu.bachelor_work.bean.HttpSessionBean;
import com.vntu.bachelor_work.models.Course;
import com.vntu.bachelor_work.models.Role;
import com.vntu.bachelor_work.services.CourseImpl;
import com.vntu.bachelor_work.services.CoursesHasStudentsServise;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
@Controller
@RequestMapping("/auth")
public class AuthController {
    private final HttpSessionBean httpSessionBean;
    private final CourseImpl courseService;
    private final CoursesHasStudentsServise coursesHasStudentsServise;
    @Autowired
    public AuthController(HttpSessionBean httpSessionBean, CourseImpl
courseService, CoursesHasStudentsServise coursesHasStudentsServise) {
        this.httpSessionBean = httpSessionBean;
        this.courseService = courseService;
        this.coursesHasStudentsServise = coursesHasStudentsServise;}
    @GetMapping("/login")
    public String getLoginPage() {
```

```
        return "login";}
    @GetMapping("/success")
    public String getSuccessPage(Model model) {
        if(httpSessionBean.getUser().getRole().equals(Role.TEACHER)) {
            model.addAttribute("teacher", httpSessionBean.getUser());
System.out.println(courseService.getAllTeachersCourses(httpSessionBean.getUser
().getId()));
            model.addAttribute("courses",
courseService.getAllTeachersCourses(httpSessionBean.getUser().getId()));
            return "teacher/teacherPage";}
        model.addAttribute("student", httpSessionBean.getUser());
        model.addAttribute("course1", new Course());
        model.addAttribute("joinCourses",
coursesHasStudentsServise.getCoursesOfStudentById(httpSessionBean.getUser().
getId()));
        return "student/studentPage";
    } }
```

Лістинг файлу “src/java/com/vntu/bachelor_work/StudentController.java”

```
package com.vntu.bachelor_work.controllers;
import com.vntu.bachelor_work.bean.FileLoader;
import com.vntu.bachelor_work.bean.HttpSessionBean;
import com.vntu.bachelor_work.bean.MediaTypeUtils;
import com.vntu.bachelor_work.models.Course;
import com.vntu.bachelor_work.models.Task;
import com.vntu.bachelor_work.services.*;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.core.io.InputStreamResource;
```

```
import org.springframework.http.HttpHeaders;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;
import javax.servlet.ServletContext;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.util.Arrays;
import java.util.List;
@Controller
@RequiredArgsConstructor
@RequestMapping("/student")
public class StudentController {
    private final UserService userService;
    private final CourseService courseService;
    private final CoursesHasStudentsServise coursesHasStudentsServise;
    private final TaskService taskService;
    private final FileService fileService;
    private final AnswerService answerService;
    private final HttpSessionBean httpSessionBean;
    @Autowired
    private ServletContext servletContext;
    private static final String DIRECTORY = "src/main/resources/storage/student";
    private static final String DEFAULT_FILE_NAME = "";
```

```
    @Value("${upload.path1}")
    private String path;
    @Value("${upload.path}")
    private String teacherPathFiles;
    @GetMapping
    public String getUser(Model model) {
        model.addAttribute("student", httpSessionBean.getUser());
        model.addAttribute("course1", new Course());
        model.addAttribute("joinCourses",
coursesHasStudentsServise.getCoursesOfStudentById(httpSessionBean.getUser().
getId()));
        return "student/studentPage";}
    @PostMapping("/searchCourse")
    public String searchCourses(@ModelAttribute("course1") Course course,
                                Model model) {
        List<Course> courseList =
courseService.getCoursesByName(course.getCourseName());
        model.addAttribute("student", httpSessionBean.getUser());
        model.addAttribute("courses", courseList);
        model.addAttribute("joinCourses",
coursesHasStudentsServise.getCoursesOfStudentById(httpSessionBean.getUser().
getId()));
        return "student/studentPage";}
    @GetMapping("/all")
    public String getAll(Model model) {
        model.addAttribute("users", userService.getAllStudents());
        return "student/allStudents";}
    @GetMapping("/join/{cid}")
    public String joinCourse(@PathVariable("cid") long cid,
                             Model model) {
```

```
coursesHasStudentsServise.addStudentToCourse(httpSessionBean.getUser().getId(
), cid);
        answerService.addUserToAnswers(httpSessionBean.getUser().getId());
        model.addAttribute("student", httpSessionBean.getUser());
        model.addAttribute("course1", new Course());
        model.addAttribute("joinCourses",
coursesHasStudentsServise.getCoursesOfStudentById(httpSessionBean.getUser().
getId()));
        return "student/studentPage";}
    @GetMapping("/course/{cid}")
    public String coursePage(@PathVariable("cid") long cid, Model model) {
        Course course = courseService.getCourseById(cid);
        model.addAttribute("course", course);
        model.addAttribute("tasks", taskService.getAllTaskOfCourseId(course));
        return "student/coursePage";}
    @GetMapping("/course/{cid}/{tid}")
    public String taskPage(@PathVariable("cid") long cid,
                           @PathVariable("tid") long tid,
                           Model model) {
        model.addAttribute("tasks",
taskService.getAllTaskOfCourseId(courseService.getCourseById(cid)));
        model.addAttribute("task", taskService.getTaskById(tid));
        String p = path + "/" + cid;
        File a = new File(p);
        if(!a.exists()) {
            a.mkdir();
        }
        p += "/" + tid;
        a = new File(p);
        if(!a.exists()) {
```

```
            a.mkdir();
        }
        List<File> files = FileLoader.getAllFilesOfDirectory(p);
        List<File> teachersFiles =
FileLoader.getAllFilesOfDirectory(teacherPathFiles + "/" + cid + "/" + tid);
        model.addAttribute("teacherFiles", teachersFiles);
        model.addAttribute("files", files);
        model.addAttribute("uid", httpSessionBean.getUser().getId());
        return "student/taskPage";}
    @RequestMapping("/{cid}/{tid}/{uid}/download")
    public ResponseEntity<InputStreamResource> downloadFile(
            @PathVariable("cid") long cid,
            @PathVariable("tid") long tid,
            @PathVariable("uid") long uid,
            @RequestParam(defaultValue = DEFAULT_FILE_NAME) String
fileName) throws IOException {
        MediaType mediaType =
MediaTypeUtils.getMediaTypeForFileName(this.servletContext, fileName);
        File file = new File(DIRECTORY + "/" + cid + "/" + tid + "/" + uid + "/" +
fileName);
        InputStreamResource resource = new InputStreamResource(new
FileInputStream(file));
        return ResponseEntity.ok()
                .header(HttpHeaders.CONTENT_DISPOSITION,
                "attachment;filename=" + file.getName())
                .contentType(mediaType)
                .contentLength(file.length())
                .body(resource);}
    @PostMapping("/course/{cid}/{tid}/{uid}")
    public String uploadFiles(
```

```
        @RequestParam("files") MultipartFile[] files, RedirectAttributes
redirectAttributes,
        @PathVariable("cid") long cid,
        @PathVariable("tid") long tid,
        @PathVariable("uid") long uid,
        Model model) {
      String p = path + "/" + cid;
      File a = new File(p);
      if(!a.exists()) {
        a.mkdir();
      }
      try {
        String finalP = p;
        Arrays.asList(files)
            .stream()
            .forEach(file -> fileService.uploadFile(finalP, file));

        redirectAttributes.addFlashAttribute("message",
            "You successfully uploaded all files!");
      }
      catch (Exception exc) {
        Task task = new Task();
        task.setCourseId(courseService.getCourseById(cid));
        model.addAttribute("task", task);
        return "/student/taskPage";
      }
      return "redirect:/student/course/" + cid + "/" + tid;
    }
}
```

Лістинг файлу “src/java/com/vntu/bachelor_work/TeacherController.java”

```
package com.vntu.bachelor_work.controllers;
import com.vntu.bachelor_work.bean.FileLoader;
import com.vntu.bachelor_work.bean.HttpSessionBean;
import com.vntu.bachelor_work.bean.MediaTypeUtils;
import com.vntu.bachelor_work.models.Answer;
import com.vntu.bachelor_work.models.Course;
import com.vntu.bachelor_work.models.Task;
import com.vntu.bachelor_work.services.*;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.core.io.InputStreamResource;
import org.springframework.http.HttpHeaders;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;
import javax.servlet.ServletContext;
import javax.validation.Valid;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.util.*;
import java.util.stream.Collectors;
```

```
@Controller
@RequiredArgsConstructor
@RequestMapping("/teacher")
public class TeacherController {
    private static final String DICT =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456
789";
    private static final Random random = new Random();
    private final UserService userService;
    private final CourseService courseService;
    private final TaskService taskService;
    private final FileService fileService;
    private final AnswerService answerService;
    private static final String DIRECTORY = "src/main/resources/storage";
    private static final String DEFAULT_FILE_NAME = "";
    @Autowired
    private ServletContext servletContext;
    private final HttpSessionBean httpSessionBean;
    @Value("${upload.path}")
    private String path;
    @Value("${upload.path1}")
    private String path1;
    @GetMapping
    public String getUser(Model model) {
        model.addAttribute("teacher", httpSessionBean.getUser());
        model.addAttribute("courses",
courseService.getAllTeachersCourses(httpSessionBean.getUser().getId()));
        return "teacher/teacherPage"; }
    @GetMapping("/change")
    public String changeInfoAboutTeacher() {
```

```
    return "teacher/";}
@GetMapping("/all")
public String getAllTeacher(Model model) {
    model.addAttribute("users", userService.getAllTeachers());
    return "teacher/allTeachers";}
@GetMapping("/createCourse")
public String createCourse(@ModelAttribute("course") Course course) {
    return "/teacher/createCourse";    }
@PostMapping("/createCourse")
public String createCourse(@ModelAttribute("course") @Valid Course course,
                        BindingResult bindingResult) {
    if(bindingResult.hasErrors()) {
        return "/teacher/createCourse";        }
    course.setUserId(httpSessionBean.getUser());
    courseService.addCourse(course);
    return "redirect:/teacher/";  }
@GetMapping("/course/{cid}")
public String takeCourse(@PathVariable("cid") long cid,
                        Model model) {
    Course course = courseService.getCourseById(cid);
    model.addAttribute("course", course);
    model.addAttribute("tasks", taskService.getAllTaskOfCourseId(course));
    return "/teacher/coursePage"; }
@GetMapping("course/{cid}/{tid}")
public String getTask(@PathVariable("cid") long cid,
                    @PathVariable("tid") long tid,
                    Model model) throws IOException {
    Course course = courseService.getCourseById(cid);
    model.addAttribute("task", taskService.getTaskById(tid));
    model.addAttribute("tasks", taskService.getAllTaskOfCourseId(course));
```

```
        String p = path + "/" + cid;
        List<File> files = FileLoader.getAllFilesOfDirectory(p);
        model.addAttribute("files", files);
        return "/teacher/taskPage";
    }
    @GetMapping("/course/{cid}/addTask")
    public String addTaskForCourse(
                            @PathVariable("cid") long cid,
                               Model model) {
        Task task = new Task();
        task.setCourseId(courseService.getCourseById(cid));
        model.addAttribute("task", task);
        return "/teacher/createTask" }
@PostMapping("/course/{cid}/addTask")
    public String addTaskForCoursePost(@PathVariable("cid") long cid,
                                @ModelAttribute("task") @Valid Task task,
                                BindingResult bindingResult) {
        task.setCourseId(courseService.getCourseById(cid));
        if(bindingResult.hasErrors()) {
            return "/teacher/createTask";
        }
        taskService.addTask(task);
        Task task1 = taskService.getTaskByNameAndCourseId(task.getTaskName(),
task.getCourseId().getId());
        answerService.addUserToTaskList(task1.getId());
        return "redirect:/teacher/course/" + task.getCourseId().getId();}
    @PostMapping("/course/{cid}/{tid}")
    public String uploadFiles(
            @RequestParam("files") MultipartFile[] files, RedirectAttributes
redirectAttributes,
```

```
            @PathVariable("cid") long cid,
            @PathVariable("tid") long tid,
            Model model) {
        String p = path + "/" + cid;
        File a = new File(p);
        if(!a.exists()) {
            a.mkdir();
        }
        try {
            String finalP = p;
            Arrays.asList(files)
                    .stream()
                    .forEach(file -> fileService.uploadFile(finalP, file));
            redirectAttributes.addFlashAttribute("message",
                    "You successfully uploaded all files!");
        }
        catch (Exception exc) {
            Task task = new Task();
            task.setCourseId(courseService.getCourseById(cid));
            model.addAttribute("task", task);
            return "/teacher/taskPage";
        }
        return "redirect:/teacher/course/" + cid + "/" + tid;
    }
    @RequestMapping("/{cid}/{tid}/download")
    public ResponseEntity<InputStreamResource> downloadFile(
            @PathVariable("cid") long cid,
            @PathVariable("tid") long tid,
            @RequestParam(defaultValue = DEFAULT_FILE_NAME) String
fileName) throws IOException {
```

```
        MediaType mediaType =
MediaTypeUtils.getMediaTypeForFileName(this.servletContext, fileName);
        File file = new File(DIRECTORY + "/" + cid + "/" + tid + "/" + fileName);
        InputStreamResource resource = new InputStreamResource(new
FileInputStream(file));
        return ResponseEntity.ok()
                .header(HttpHeaders.CONTENT_DISPOSITION,
"attachment;filename=" + file.getName())
                .contentType(mediaType)
                .contentLength(file.length()) //
                .body(resource);
    }
    @GetMapping("/course/{cid}/{tid}/answers")
    public String answersForTask(@PathVariable("cid") long cid,
                                 @PathVariable("tid") long tid,
                                 Model model) {
        Course course = courseService.getCourseById(cid);
        model.addAttribute("tasks", taskService.getAllTaskOfCourseId(course));
        String p = path + "/student" + "/" + cid;
        File a = new File(p);
        if(!a.exists()) {
            a.mkdir();
        }
        List<Answer> answers = answerService.getAnswersByTaskId(tid);
        Map<Long, List<File>> answersAndFilesList = new HashMap<>();
        for(int i = 0; i < answers.size(); i++) {
            String finalP = p;
            String pp = finalP + "/" + answers.get(i).getUser().getId();
            System.out.println(pp);
```

```
            File aaa = new File(pp);
            answersAndFilesList.put(answers.get(i).getUser().getId(),
FileLoader.getAllFilesOfDirectory(pp));
            FileLoader.getAllFilesOfDirectory(pp).forEach(System.out::println);
        }
        model.addAttribute("task", taskService.getTaskById(tid));
        model.addAttribute("answers", answers);
        model.addAttribute("answer1", new Answer());
        model.addAttribute("studentsFiles", answersAndFilesList);
        return "teacher/answersForTask";
    }
    @PostMapping("/course/{cid}/{tid}/{uid}")
    public String setMark(@PathVariable("cid") long cid,
                          @PathVariable("tid") long tid,
                          @PathVariable("uid") long uid,
                          @ModelAttribute("answer1") @Valid Answer answer,
                          Model model) {
if(answer.getMark() > taskService.getTaskById(tid).getMaxMark()) {
            return "redirect:/teacher/course/" + cid + "/" + tid + "/" + "answers";
        }
        answerService.setMarkForTask(answer.getMark(), tid, uid);
        return "redirect:/teacher/course/" + cid + "/" + tid + "/" + "answers";}
    private String makeKeyCourse() {
        StringBuilder key = new StringBuilder();
        for (int i = 0; i < 8; i++ )
            key.append(DICT.charAt(random.nextInt(DICT.length())));
        return key.toString();}
    @RequestMapping("/{cid}/{tid}/{uid}/download")
    public ResponseEntity<InputStreamResource> downloadStudentFile(
            @PathVariable("cid") long cid,
```

```
        @PathVariable("tid") long tid,
        @PathVariable("uid") long uid,
        @RequestParam(defaultValue = DEFAULT_FILE_NAME) String
fileName) throws IOException {
    MediaType mediaType =
MediaTypeUtils.getMediaTypeForFileName(this.servletContext, fileName);
    File file = new File(DIRECTORY + "/student"+ "/" + cid + "/" + tid + "/" +
uid + "/" + fileName);
    InputStreamResource resource = new InputStreamResource(new
FileInputStream(file));
    return ResponseEntity.ok()
            .header(HttpHeaders.CONTENT_DISPOSITION,
"attachment;filename=" + file.getName())
            .contentType(mediaType)
            .contentLength(file.length()) //
            .body(resource);}
  @GetMapping("/course/{cid}/resultsPage")
  public String resultsPage(@PathVariable("cid") long cid,
                    Model model) {
    List<Answer> allAnswersList =
answerService.getAllAnswersByCourseId(cid);
    Course course = courseService.getCourseById(cid);
    model.addAttribute("tasks1", taskService.getAllTaskOfCourseId(course));
    List<List<Answer>> listAnswers = new ArrayList<>();
    model.addAttribute("answers",
answerService.getAllAnswersByCourseId(cid));
    model.addAttribute("tasks",
answerService.getAllAnswersByCourseIdDistinct(cid));
    model.addAttribute("firstTask",
answerService.getAllAnswersByCourseIdDistinct(cid).get(0));
```

```
answerService.getAllAnswersByCourseId(cid).stream().forEach(System.out::println);
        List<Long> listStudentId = allAnswersList.stream().map(el -> el.getUser().getId()).distinct().collect(Collectors.toList());
        listStudentId.stream().forEach( el -> listAnswers.add(answerService.getAnswerByCourseIdAndUserId(cid, el)));
        model.addAttribute("groupStudents", listAnswers);
        return "teacher/resultsPage";
    }}
```

ДОДАТОК Г

Реалізація репозиторіїв

Лістинг файлу “com/vntu/bachelor_work/repositories/AnswersRepository.java”

```
package com.vntu.bachelor_work.repositories;
import com.vntu.bachelor_work.models.Answer;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;
import java.util.List
@Repository
public interface AnswersRepository extends JpaRepository<Answer, Long> {
   @Query(value = "select id, mark, task_id, user_id from answers " +
         "where task_id = :tid", nativeQuery = true)
   List<Answer> getAnswersByTaskId(@Param("tid") long tid);
   @Transactional
   @Modifying(clearAutomatically = true)
   @Query(value = "insert into answers (mark, task_id, user_id)" +
         " select 0, t.id, u.id from tasks t" +
         " join courses_has_users chu on t.course_id = chu.course_id" +
         " join users u on u.id = chu.user_id" +
         " where u.id = :uid", nativeQuery = true)
   void addStudentToAnswer(@Param("uid") long uid);
   @Transactional
   @Modifying(clearAutomatically = true)
   @Query(value = "update answers " +
```

```
            "set mark = :mark where task_id = :tid and user_id = :uid",
nativeQuery = true)
    int setMarkForTask(@Param("mark") long mark, @Param("tid") long tid,
@Param("uid") long uid);
    @Modifying
    @Query(value =
        "delete from answers a where a.task_id in " +
        "(select tt.idd from " +
        "(select t.id as idd from tasks t" +
        " inner join courses c on c.id = t.course_id" +
        " where c.id = :cid) as tt);",
        nativeQuery = true)
    void deleteAllAnswersOfCourse(@Param("cid") long cid);
    @Transactional
    @Modifying(clearAutomatically = true)
    @Query(value = "insert into answers(mark, task_id, user_id) " +
        "select 0, :tid, u.id from users u " +
        "join courses_has_users chu on chu.user_id = u.id " +
        "join tasks t on t.course_id = chu.course_id " +
        "where t.id = :tid ", nativeQuery = true)
    void addStudentToTaskList(@Param("tid") long tid);
    @Query(value = "select a.id, a.mark, a.task_id, a.user_id from answers a " +
        "join tasks t on a.task_id = t.id " +
        "join courses c on c.id = t.course_id " +
        "where c.id = :cid " +
        "order by a.task_id, a.user_id", nativeQuery = true)
    List<Answer> getAnswerByCourseId(@Param("cid") long cid);
    @Query(value = "select a.id, a.mark, a.task_id, a.user_id from answers a " +
        "join tasks t on a.task_id = t.id " +
        "join courses c on c.id = t.course_id " +
```

```
        "where c.id = :cid " +
        "group by a.task_id " +
        " order by a.task_id", nativeQuery = true)
    List<Answer> getAnswerByCourseIdDistinct(@Param("cid") long cid);
    @Query(value = "select a.id, a.mark, a.task_id, a.user_id from answers a " +
        "        join tasks t on a.task_id = t.id " +
        "        join courses c on c.id = t.course_id" +
        "        where c.id = :cid and a.user_id = :uid" +
        "        order by a.user_id, a.task_id", nativeQuery = true)
    List<Answer> getAnswerByCourseIdAndUserId(@Param("cid") long cid,
@Param("uid") long uid);
}
```

Лістинг файлу “com/vntu/bachelor_work/repositories/CourseRepository.java”

```
package com.vntu.bachelor_work.repositories;
import com.vntu.bachelor_work.models.Course;
import com.vntu.bachelor_work.models.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;
import java.util.List;
@Repository
public interface CourseRepository extends JpaRepository<Course, Long> {
    @Transactional
    @Modifying(clearAutomatically = true)
    @Query(value = "insert courses(course_name, description, teacher_id) " +
```

```
        "value(:#{#course.getCourseName()},
:#{#course.getDescription()},:#{#course.getUserId().getId()}  );", nativeQuery =
true)
    int addCourse(@Param("course") Course course);
    @Query(value = "select id, course_name, description, teacher_id  " +
        " from courses", nativeQuery = true)
    List<Course> getAllCourses();
    @Query(value = "select id, course_name, description, teacher_id  " +
        " from courses where course_name = :name", nativeQuery = true)
    List<Course> findCourseByName(@Param("name") String name);
    @Query(value = "select id, course_name, description, teacher_id  " +
        " from courses where teacher_id = :id", nativeQuery = true)
    List<Course> findAllTeachersCourses(@Param("id") long id);
    @Query(value = "select c.id, c.course_name, c.description, c.teacher_id " +
        "from courses c " +
        "join users u on c.teacher_id = u.id " +
        "where (c.course_name like :str) and u.role = 'TEACHER'", nativeQuery =
true)
    List<Course> findCoursesByName(@Param("str") String str);
    @Query(value = "select c.id, c.course_name, c.description, c.teacher_id from
courses c " +
        "join courses_has_users cu on c.id = cu.course_id " +
        "where cu.user_id=:sid", nativeQuery = true)
    List<Course> getCoursesOfStudent(@Param("sid") long sid);
}
```

Лістинг файлу “com/vntu/bachelor_work/repositories/TaskRepository.java”

```
package com.vntu.bachelor_work.repositories;
import com.vntu.bachelor_work.models.Task;
import org.springframework.data.jpa.repository.JpaRepository;
```

```
import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;
import java.util.List;
@Repository
public interface TaskRepository extends JpaRepository<Task, Long> {
   @Query(value = "select id, task_name, description, course_id, max_mark from
tasks " +
         "where id = :id", nativeQuery = true)
   Task getTaskById(@Param("id") long id);
   @Query(value = "select id, task_name, description, course_id, max_mark from
tasks " +
         "where course_id = :id", nativeQuery = true)
   List<Task> findAllTaskOfCourseId(@Param("id") long id);
   @Transactional
   @Modifying(clearAutomatically = true)
   @Query(value = "insert tasks(task_name, description, course_id, max_mark) " +
         "values(:#{#task.getTaskName()}, :#{#task.getDescription()}" +
         ", :#{#task.getCourseId().getId()} ,:#{#task.maxMark} );", nativeQuery =
true)
   int addTask(@Param("task") Task task);
   @Modifying
   @Query(value = "delete from tasks t where t.course_id =:courseId, nativeQuery
= true) void deleteAllByCourseId(@Param("courseId") long id);
   @Modifying
   @Query(value = "delete from tasks t where t.id =:id , nativeQuery = true)
   void taskById(@Param("id") long id);
```

```
    @Query(value = "select id, task_name, description, course_id, max_mark from
tasks " +"where task_name = :taskName and course_id = :cid", nativeQuery = true)
    Task getTaskByNameAndCourseId(@Param("taskName") String taskName,
@Param("cid") long id);
}
```

Лістинг файлу “com/vntu/bachelor_work/repositories/UserRepository.java”

```
package com.vntu.bachelor_work.repositories;
import com.vntu.bachelor_work.models.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;
import java.util.List;
import java.util.Optional;
@Repository
public interface UserRepository extends JpaRepository<User, Long> {
    @Transactional
    @Modifying(clearAutomatically = true)
    @Query(value = "insert users(first_name, last_name, email, `password`, role,
active) values (:#{#user.getFirstName()} , :#{#user.getLastName()} ,
:#{#user.getEmail()}, :#{#user.getPassword()}, :#{#user.getRole().toString()} ,
:#{#user.getStatus().toString()});", nativeQuery = true)
    int addUser(@Param("user") User user);
    @Query(value = "select id, first_name, last_name, email, password, role, active
from users where email = :email and password = :password", nativeQuery = true)
User findUserByEmailAndPassword(@Param("email") String email,
@Param("password") String password);
```

```
    @Query(value = "select id, first_name, last_name, email, password, role, active
from users where email = :email ", nativeQuery = true)
    Optional<User> findUserByEmail(@Param("email") String email);
    @Query(value = "select id, first_name, last_name, email, password, role, active
from users where role = 'TEACHER'", nativeQuery = true)
    List<User> findAllTeachers();
    @Query(value = "select id, first_name, last_name, email, password, role, active
from users where role = 'STUDENT'", nativeQuery = true)
    List<User> findAllStudents();
}
```

ДОДАТОК Д

Протокол перевірки навчальної (кваліфікаційної) роботи

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: «Веб-застосунок для організації та оцінювання навчального процесу»
Тип роботи: бакалаврська дипломна робота
Підрозділ кафедра обчислювальної техніки

Показники звіту подібності Unicheck

Оригінальність 93,4% Схожість 6,6%

Аналіз звіту подібності (відмітити потрібне):

- ✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.
- ☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
- ☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Захарченко С.М.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Терещук Е. П.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Городецька О. С.
(підпис) (прізвище, ініціали)