Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Бурлаки Олександра Сергійовича*
(ПІБ)

академічної групи *122-20-2*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему: *Розробка навчальної онлайн-платформи для організації навчального процесу з використанням мови програмування Java*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Спірінцев В.В.*			
розділів:				
спеціальний	*доц. Спірінцев В.В.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2024

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » 2024 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-20-2* *Бурлаки О.С.*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка навчальної онлайн-платформи для організації навчального процесу з використанням мови програмування Java*

затверджена наказом ректора НТУ «ДП» від *23.05.2024* № *469- с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проєктно-технологічної практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***15.06.2024 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***19.06.2024 р.***

Завдання видав _____ *доц. Спірінцев В.В*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Бурлака О.С*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2024 р.*

Термін подання кваліфікаційної роботи до ЕК: *24.06.2024 р.*

РЕФЕРАТ

Пояснювальна записка: 94 с., 64 рис., 3 додатки, 17 джерел.

Об'єкт розробки: навчальна онлайн-платформа для організації навчального процесу з використанням мови програмування Java.

Мета кваліфікаційної роботи: розробка навчальної онлайн-платформи для організації навчального процесу з наявним «user friendly» інтерфейсом.

У вступі аналізується та досліджується сучасний стан проблеми, визначається мета дослідження та сфера галузі застосування кваліфікаційної роботи, а також розглядається актуальність теми та встановлюється постановка завдання.

У першому розділі включено аналіз області дослідження, визначення актуальності завдання та призначення розробки, формулювання постановки завдання, та встановлення зазначених вимог до програмної реалізації, технологій та програмних засобів.

У другому розділі проаналізовано всі існуючі проблеми та їх вирішення, обрано платформу для реалізації онлайн-платформи, виконано проектування всіх потрібних елементів програми, баз даних, та структури проекту та розробку самої програми, описана робота системи, алгоритми та структуру функціонування програми, визначено вхідні та вихідні дані, а також розглянуто характеристики параметрів засобів, виклик та завантаження застосунку а також розкрито принцип роботи програми.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню додатку та розраховано час на його створення.

Практичне значення полягає у створенні навчальної онлайн-платформи, що забезпечує: можливість ефективного застосування програми, використання її користувачами, тобто відтворює максимальний відклик від програми до користувача для отримання всій необхідній інформації завдань, лекцій, та її записів для студентів, та покращує зв'язок між студентом та викладачем.

Актуальність розробки навчальної-онлайн платформи обґрунтовано тим що за необхідністю студенти потребують використовувати систему для відображення всій інформації для отримання 100% реалізації навчального процесу за умов складних часів.

Список ключових слів: ОНЛАЙН-ПЛАТФОРМА, НАВЧАЛЬНА ОНЛАЙН-ПЛАТФОРМА, БАЗА ДАНИХ, JAVA, HTML, CSS, SPRING BOOT,INTELLIJ IDEA, BOOTSTRAP, POSTGRESQL, SPRING MVC, SPRING SECURITY, LOMBOK, THYMELEAF, SPRING BOOT DATA JPA, REST.

ABSTRACT

Explanatory note: 94 P., 64 fig., 3 adj., 17 sources.

Object of development: an online educational platform for organizing the educational process using the Java programming language.

The purpose of the qualification work: to develop an online training platform for organizing the educational process with an existing "user-friendly" interface.

In the introduction, the current state of the problem is analyzed and investigated, the purpose of the study and the scope of application of the qualification work are determined, as well as the relevance of the topic is considered and the task statement is established.

The first chapter includes an analysis of the research area, determining the relevance of the task and the purpose of development, formulating the task statement, and establishing the specified requirements for software implementation, technologies, and software tools.

In the second chapter, all existing problems and their solutions are analyzed, a platform for implementing the online platform is selected, all the necessary elements of the program, databases, and project structure are designed, and the program itself is developed, the system operation, algorithms and structure of the program are described, input and output data are determined, and the characteristics of the parameters of the tools, calling and loading the application are considered, and the principle of operation of the program is revealed.

In the economic section, the complexity of the developed information system is determined, the cost of creating an application is calculated, and the time required to create it is calculated.

Practical significance is to create an online learning platform that provides: the ability to effectively apply the program, use it by users, that is, reproduces the maximum response from the program to the user to get all the necessary information about tasks, lectures, and its recordings for students, and improves communication between the student and the teacher.

The relevance of developing an online learning platform is justified by the fact that if necessary, students need to use the system to display all information in order to get 100% implementation of the educational process in difficult times.

Keyword list: online platform, online training platform, database, JAVA, HTML, CSS,SPRING BOOT, INTELLIJ IDEA, BOOTSTRAP, POSTGRESQL, SPRING MVC, SPRING SECURITY, LOMBOK, THYMELEAF, SPRING Boot Data JPA, REST.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ	–	Програмне забезпечення
ПК	–	Персональний комп’ютер
ВБ	–	Веб-додаток
ОП	–	Онлайн-платформа
ОС	–	Операційна система
БД	–	База даних
CMS	–	Система управління контентом
CSS	–	Cascading Style Sheets
SQL	–	Structured Query Language
HTML	–	Hyper Text Markup Language
DI	–	Dependency Injection

ВСТУП

Веб-технології за останні 10 років мають високий темп розвитку та безліч користувачів по всьому світу. Ця індустрія стала невід'ємною частиною нашого життя та використовується майже в усьому, чи то сайти доставки, чи то магазини, чи то репозиторії з якоюсь інформацією. В випадку цієї передатестаційної практики це сайт онлайн освіти який додає можливості знаходячись вдома реалізовувати всю потужність навчальної програми університетів за умов складних часів.

Онлайн-платформи (ОП) дозволяють ефективно використовувати веб-додатки задля реалізації всіх задуманих ідей, використовувати все необхідне для зручного спілкування між сервісом там користувачем. В цілому кожного дня з'являються тисячі нових сайтів що призводить до нових змін в нашому житті. Наприклад відкриття нового сайту з обслуговуванням якогось ресторану через інтернет призведе до того що буде більше звертати увагу на такі сайт, призведе також для отримання більшої кількості користувачів для клієнтської бази, тому що чим більше часу йде тим більш зручніше це використовувати. Крім того, це все може покращити всі логістичні функції держави, або сервісу, зв'язок між сервісом та користувачем, автоматизувати якісь дії, покращити життя багатьом людям в більш зручному сенсі.

В цьому випадку завдяки ОП для онлайн навчання, студенти та викладачі можуть не виходячи з дому спілкуватись, бачити інформацію об кожному користувачі, лекції, практичні матеріали, та отримувати оцінки за свої роботи не виходячи до університету.

Актуальність розробки навчальної ОП для організації навчального процесу полягає в тому, що в складних умовах війни виростає зручність у тому що з'являється необхідність в сенсі зручності та безпеки для знаходження студентів або викладачів удома. Зокрема війна викликає безліч негативних показників, наприклад, людям все складніше буває виходити із дому, це дає незручності та обмеження в пересуванні, зменшення кількості студентів та

викладачів які можуть прийти не вчасно на заняття або зовсім не прийти за будь яких обставин. І це може стати єдиним вирішенням проблеми.

Також онлайн-платформа для навчання може дати будь які інструменти та засоби, для ідеальних практичних занять, чи то програми які можуть бути більш краще використовувані вдома чим в університеті, та технічні засоби комп'ютерів.

Результатом виконання кваліфікаційної роботи є ОП для організації навчального процесу, яка має зручний для використання інтерфейс, користувачі навіть без суттєвих навичок зможуть використовувати цю систему. Більш орієнтована система на користувачів, система має якісний фідбек, зручне застосування, детальну але в тій же час просту для застосування інформацію та інше.

ОП написана на сучасних мовах програмування та їхніх фреймворках, такі як Java, Spring framework (Spring boot, Spring Security, Spring boot data jpa, Spring mvc, Lombok, thymeleaf, postgresql, html та css). І тому вона має довгострокову підтримку в обслуговувані, та модернізації якісь структур для покращення самої системи.

Об'єктом дослідження є навчальна ОП для організації навчального процесу з використанням мови програмування Java.

Мета кваліфікаційної роботи: розробка навчальної онлайн-платформи для організації навчального процесу з використанням мови програмування Java, яка буде зручна та зрозуміла для використання користувачам.

Для вирішення поставленої мети необхідно вирішити наступні задачі:

– розглянути більш масштабні проблеми такі як логістика в обміну в інформації, зручність в навігації по сайту, зручність в додаванні нового матеріалу та зручність в його застосуванні;

– визначити основні вимоги, які повинні бути враховані при розробці ОП. Ці вимоги будуть складатися з потрібних характеристик ПК, потрібних програм, встановлених на комп'ютер фреймворків для їх роботи та інше;

– розробити навчальну ОП для організації навчального процесу та описати її алгоритми роботи та структуру баз даних;

– виконати економічну частину цієї роботи для розуміння витрат на розробку програмного продукту.

Практичне значення полягає у навчальної ОП для організації навчального процесу, що забезпечує: можливість ефективної роботи між студентом та викладачем за умов онлайн спілкування та отримання інформації. ОП забезпечує отримання інформації, оцінок, лекцій, методичних матеріалів, практичних матеріалів або навіть відео-матеріалів, забезпечує спілкування між студентом та користувачем. В результаті виконання цих критеріїв буде розроблено ОП яка може конкурувати між новими розробками в цієї темі.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

З появою складних умов на території Україні почала зростати потреба закладам середньої, вищої освіти надавати свої послуги с питань освіти та науки онлайн. Зручність використання мережі інтернет обґрунтовано тим що логістика обміну інформації між студентом та викладачем може бути дедалі складним через обмеження в пересуванні та інше, також тим що зручність в навігації по сайту в пошуках потрібній інформації найбільш легший чим це було би знаходячись в закладах освіти, і також ОП для організації навчання можуть надавати більш зручніше матеріал на сайти чим тільки під час занять в університетах.

ОП для надання послуг з питань освіти та науки розширюють можливості закладів вищої освіти в надавані своїх послуг та розширюють можливості в студентах бути в темпі навчального процесу.

Але навіть якщо подивитись на всі зручності онлайн навчання в цій системі є деякі мінуси: наприклад питання контролю найбільш загострене чим те було би в університетах, чи те контроль заходи чи те сесія але викладачам складно слідкувати за тим щоб було все зручно та прозоро.

Вирішенням цієї проблеми може водночас складним та водночас легким. Розробники ОП для навчання повинні зробити деякі функції, наприклад якщо це тестування то неможливість відкривати інші сторінки браузеру та неможливість відкривати навіть їх на інших моніторах комп’ютеру.

Також є проблема з доступністю до потрібних для навчання технічного обладнання. Це може бути проблемою для багатьох студентів, тому що не в всіх є можливість в наявності в себе новітнього ПК для більш зручного застосування онлайн програм для навчання. Це все вирішується теж не сильно складно, потрібно обґрунтовувати завдання та робити його зручнішим для всіх студентів

та оптимізувати ОП в вигляді веб-сайту щоб вона використовувала менше ресурсів комп’ютера.

Остання проблема яка буде зазначена, це проблема в дисципліні. Більшість студентів коли почують всю свободу онлайн навчання можуть більш лінитись там відволікатись від навчального процесу. Але навіть так це можна вирішити більш частішим контролем якості навчання.

1.2. Призначення розробки та галузь застосування

В цій кваліфікаційній роботі розглядається створення навчальної ОП для організації навчання з використанням мови програмування Java та її фреймворків.

Призначення розробки:

– ОП має зручності в надаванні всіх лекцій, практичного матеріалу, методичних матеріалів, та інше не виходячи з дому;

– розробка маж доступність та зручність в використанні для всіх користувачів;

– студент сам може підлаштовувати свій темп навчання під темп навчання університету, що надає студентам більш зручніше використовувати свій час для навчання та застосування матеріалу;

– ОП для навчання надає викладачам більш зручніше викладати матеріал та слідкувати за роботою студента під час онлайн навчання. Та студенті відвідувати, отримувати інформацію, отримувати оцінки за свої роботи не приходячи до університету;

– великим плюсом онлайн навчання є економічна частина. Спадає попит на використання транспорту та харчової промисловості(в самому університеті), що призводить до того що витрати на це все вже не потрібні;

– зручність в використанні всіх необхідних новітніх технологій та в доступності їх облаштування.

Призначення розробленої системи:

– забезпечення всім необхідним для надання всіх необхідних послуг в сфері освіти та науки за умов онлайн навчання;

– забезпечення всіх інструментів які необхідні для надання інформації між викладачем там студентами;

– забезпечення всім необхідним для найкращої логістики обміну інформації між студентами та викладачами для забезпечення максимальної ефективності навчання.

ОП для організації навчального процесу повинна містити та забезпечувати ці пункти:

– зрозумілий та зручний в використанні для користувача, тобто для студента, викладача та інших;

– забезпечення в додаванні, використанні, видаленні, або редагуванні інформації яка необхідна для реалізації задумів навчального процесу;

– функції спілкування з викладачем та студентом в обидві сторони для обговорення проблем, питань, та їх подальшого вирішення.

Через проведений аналіз з питань та пошуку існуючих рішень, зазначено основні особливості навчальної ОП:

– наявність всього інструментарію для імплементування (реалізації) навчального процесу;

– налаштована та зручна в застосуванні база даних для зберігання та використання всій інформації що необхідна для даної ОП;

– перегляд всіх курсів, всіх завдань лекцій та практичних занять, методичних матеріалів, та перегляд оцінки за ці зайняття;

– обробка інформації, наприклад відправлення готового завдання на оцінку та її отримання. Можливість додавання зручного опису недоліків та помилок готового завдання;

– можливість для студента переглядати всю необхідну інформацію о курсах та завданнях, можливість самореєстрації та самовидалення якщо це потрібно;

– інструменти які потрібні для умовних позначок для студента якщо потрібно занотувати якусь інформацію, або пройдений матеріал.

За висновком пункту, є реалізація та обґрунтування необхідного пакету функцій навчальної ОП для організації навчального процесу з застосування мови програмування Java:

– головна сторінка ОП, з переліком всіх доступних курсів;

– кабінет користувача, з наявною інформації, оцінок, перелік реєстрованих курсів та тощо. Кабінет користувача може бути різним якщо це студент або викладач або навіть адміністрація;

– детальні сторінки курсів;

– детальні сторінки завдань, лекцій, методичних матеріалів, та практичних завдань;

– детальні сторінки з оцінками курсів та відгуками;

– можливість перегляду всіх завдань за хронологічною послідовністю;

– можливість редагувати видаляти та добавляти курси та завдання, якщо користувач є викладачем;

– інструменти для керування базою даних та всім сайтом.

1.3. Підстава для розробки

Підставами для розробки (виконання кваліфікаційної роботи) є:

– освітня програма спеціальності 122 «Комп’ютерні науки»;

– навчальний план та графік навчального процесу;

– наказ ректора Національного технічного університету «Дніпровська політехніка» №469-с від 23.05.2024 р.;

– завдання на кваліфікаційну роботу на тему «Розробка навчальної онлайн-платформи для організації навчального процесу з використанням мови програмування Java».

1.4. Постановка завдання

В кваліфікаційній роботі розроблено проект під назвою «Розробка навчальної онлайн-платформи для організації навчального процесу з використанням мови програмування Java».

Мета кваліфікаційної роботи: розробка навчальної онлайн-платформи для організації навчального процесу з використанням мови програмування Java, яка буде зручна та зрозуміла для використання користувачам.

Ця онлайн-платформа для організації навчання надає інструментарій та всі необхідні програми та сам сайт для ефективної роботи навчального процесу закладу середньої або вищої освіти.

Окремі вимоги, які ОП повинна реалізовувати в своїй роботі:

– зрозумілість та зручність;

– швидкість в відклику від сервера до користувача;

– швидкість в роботі системи;

– швидкість сприйняття програмою щодо інформації яку потрібно добавити видалити або редагувати до бази даних;

– наявність всіх інструментів для роботи з інформацією.

Створення навчальної онлайн-платформи для організації навчального процесу з використанням мови програмування Java повинна бути містити такі етапи, як:

– аналіз всіх існуючих рішень та використання їх ідей для знаходження більш зручних рішень;

– планування всіх етапів щодо поступової розробки системи, баз даних та їх відносин;

– реалізація всіх задуманих ідей, фреймворків, програм, мов;

– програмування тощо, задля досягнення кінцевої цілі;

– фінальне тестування програми для виявлення її недоліків та подальшого видалення з програми.

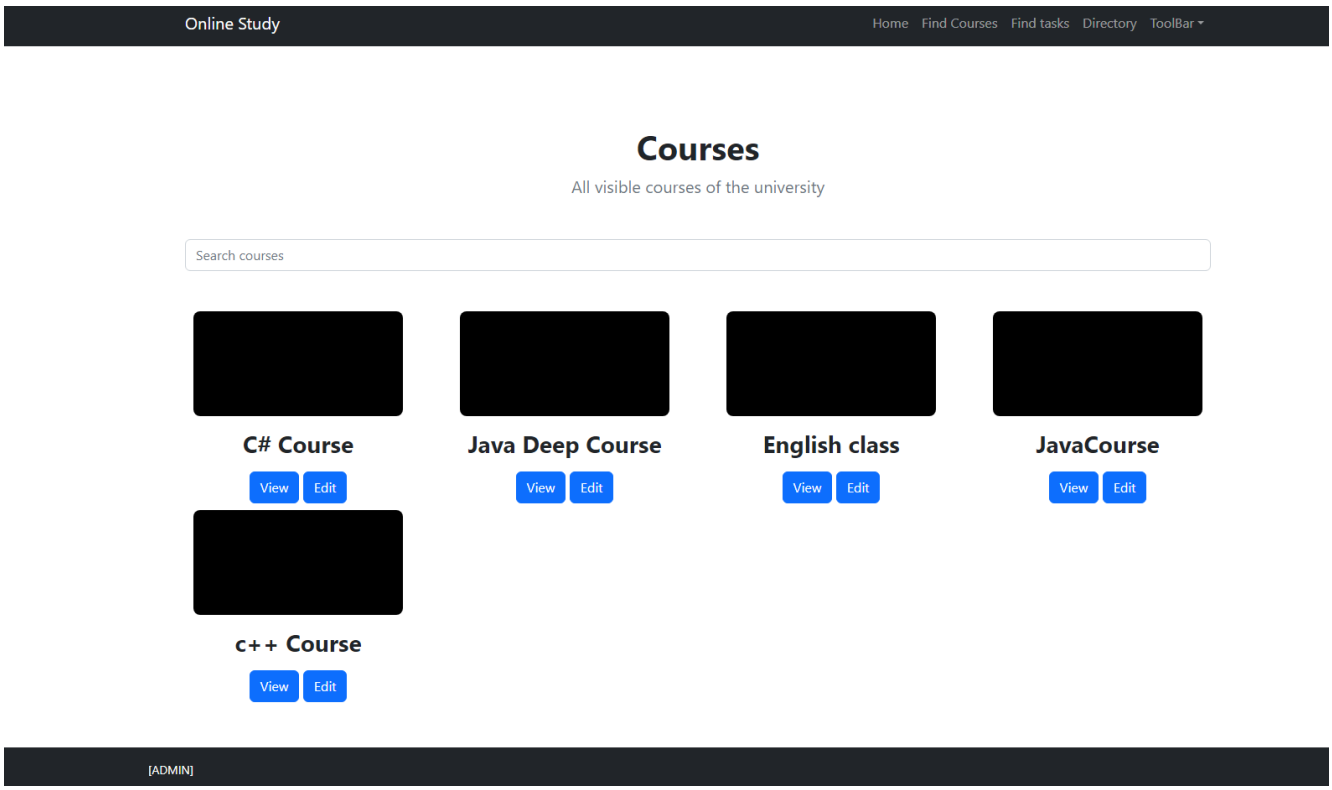


Рис.1.1. Головна сторінка навчальної ОП

Головна сторінка містить:

– thymeleaf layout (хедери), які містять назву сайту, назву сайту, інструментарій для пошуку інформації(курси, завдання, тощо), та toolbar, котрий містить можливість додавання курсів, завдань, та їх редагування та видалення. Та можливість(якщо користувач не ввійшов до сайту) зареєструватися або ввійти до сайту, та якщо він все ж таки ввійшов, то можливість вийти з аккаунту;

– назва самої сторінки Courses тобто курси університету та опис цієї сторінки;

– пошуковий рядок для пошуку необхідній інформації для користувача;

– весь перелік курсів разом з кнопкою VIEW (перегляд) та кнопкою EDIT (редагування, для викладачів та адміністрації сайту);

– та нижній footer, де написано який користувач увійшов до сайту;

Сайт був розроблений за допомогою мови програмування Java, та деяких фреймворків, такі як:

- Spring framework;
- Spring boot;
- Spring boot data jpa;
- Spring security;
- Spring MVC;
- Lombok;
- Thymeleaf;
- PostgreSQL;
- HTML, CSS та Bootstrap.

1.5. Вимоги до програмного виробу

1.5.1. Вимоги до функціональних характеристик

Навчальна ОП для організації навчального процесу яка була розроблена на мові програмування Java, повинна містити та підтримувати такі дії:

- повинна бути в будь який час зручна та зрозуміла в застосуванні;
- повинна надавати всі необхідні інструменти для надання необхідних послуг;
- повинна містити довгострокову функціональну та програмну підтримку;
- повинна містити системи та потрібний функціонал для пошуку інформації на сайті в цілому;
- повинна містити захист від інших осіб які ніяк не відносяться до закладу освіти;
- повинна містити шифрування даних, наприклад паролю від кабінету користувача;

– повинна містити швидке оновлення інформації, якщо її хтось редагує або додає;

– повинна мати функціонал який потрібен тільки для певної області користувачів (адміністрації та викладачів);

– повинна містити кабінет користувача, з переглядом всій інформації о користувачі;

– повинна містити отримання та відправку всіх даних на більш швидкому та прозорому рівні;

– повинна містити можливості для зв’язку.

1.5.2. Вимоги до інформаційної безпеки

Щодо інформаційної системи в Spring framework встановлено фреймворк під назвою Spring Security який допомагає налаштовувати систему для захисту всій інформації яка надходить до користувачів та відходить до сайту від них:

– шифрування даних за допомогою BCryptPasswordEncoder();

– неможливість використання послугами сайту без входу до системи;

– відокремлення реєстрування нових користувачів від ролі ADMIN;

– можливість для системи сховати інформацію та функціонал від користувача який йому він не потрібен;

– можливість більш зручного застосування бази даних та подальшого шифрування інформації.

1.5.3. Вимоги до складу технічних засобів

Програма потребує з боку користувача:

– персональний ПК який має всі технічні обладнання для доступу;

– можливість доступу ПК до мережі інтернет;

– встановлену та налаштовану ОС: Windows 7/8/10/11, Linux;

– веб-браузер: Google Chrome, Microsoft Edge, Opera та інші;

– мінімальна швидкість активного підключення до Інтернету 512 Кбіт/с і вище.

Та деякі технічні засоби ПК користувача які задовольняють мінімальним вимогам ОП:

– процесор: Intel Pentium E2180;
– оперативна пам’ять: 3гб;
– відеоадаптер: NVIDIA GeForce 8600;
– вільне місце на диску: 1гб;
– інтернет-підключення: 512 кбіт/с.

Список рекомендованих технічним вимог:

– процесор: Intel Core i5 (версія для ПК);
– оперативна пам’ять: 6+гб;
– відеоадаптер: NVIDIA GeForce GTX 660 (2гб відеопам’яті);
– вільне місце на диску: 1гб;
– інтернет-підключення: 1 мбіт/с.

1.5.4. Вимоги до інформаційної та програмної сумісності

Для запуску та функціонування програми з середовища розробки потрібні такі програми та характеристики:

– встановлена та налаштована ОС: Windows 7/8/10/11, Linux;
– встановлений та налаштований веб-браузер: Google Chrome, Microsoft Edge, Opera та інші.

Та список мінімальних технічних вимог:

– процесор: Intel Pentium E2180;
– оперативна пам’ять: 3гб;
– відеоадаптер: NVIDIA GeForce 8600;
– вільне місце на диску: 1гб;
– інтернет-підключення: 512 кбіт/с.

Для розробки цієї ОП знадобились такі програми та фреймворки:

– Intellij IDEA;
– PostgreSql;
– Java;
– Google Chrome;
– Та встановлені та вбудовані фреймворки до проекту(Spring boot,
– Spring Security, Spring boot data jpa, Spring mvc, Lombok, thymeleaf, postgresql, html та css).

Для створення системи на основі Spring framework конфігурація складає:

– Java 17 або вище;
– Spring Framework 3.2.4. або вище;
– підтримка збірки надається для Maven 3.2.

Для запуску серверу на якому знаходиться ОП є вбудований до Spring framework Tomcat 10.1.9.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

Навчальна ОП для організації навчального процесу з використанням мови програмування Java призначена для:

– ефективної реалізації навчальних планів закладу освіти;

– швидкої та ефективної логістики в обміну інформації;

– швидкої та зрозумілої навігації по сайту;

– отримання повної інформації щодо навчання.

Виходячи з цього функціональне призначення навчальної ОП для організації навчального процесу:

– відображення всіх існуючих курсів на сайті;

– відображення всіх існуючих завдань на які можна подивитись зі сторінки курсу або особистого кабінету користувача;

– відображення оцінок студента, які можна переглянути на сторінці користувача разом із відгуком на завдання;

– зручні форми для додавання, редагування, видалення курсів, завдань тощо;

– система перегляду всіх студентів зареєстрованих на курсі, та редагування їх оцінок;

– можливість реєстрування нових акаунтів, та можливість входу до старого акаунта;

– можливість реєстрування на курси;

– можливість (для користувачів з привілеями ADMIN) переглянути всіх користувачів на курсах.

Основні підпункти які реалізовані в навчальній ОП для організації навчання:

– ефективна робота бази даних, зручне додавання, редагування, видалення даних з них;

– зручний та зрозумілий інтерфейс, легкий для застосування та перебігу по сайту;

– автоматизоване додавання до всіх сторінок сайту даних, які нещодавно були додані до бази даних сайту;

– використання REST API;

– використання Spring framework для високої ефективності роботи сайту.

2.2. Опис застосованих математичних методів

Оскільки особливості предметної області розв'язуваної задачі не передбачають застосування математичних методів, при розробці онлайн-платформи для організації навчального процесу математичні методи не використовувалися.

2.3. Опис використаних технологій та мов програмування

Для реалізації навчальної ОП були використовувані такі мови програмування та технології:

– мова програмування Java - високотехнологічна мова програмування, яка застосовується для вирішення та написання складних програм, проблем автоматизації, та написання веб-сайтів. Java являє собою об’єктно-орієнтовану та багатоплатформну мову програмування, за допомогою якої можна написати легкі для розуміння розробникам програму, та її подальше розширення та модернізацію [1-2] ;

– Spring framework - найпопулярніший фреймворк Java призначений для розробки веб додатків. Складові Spring framework [1, 3]:

- Spring core container: набір бібліотек та класів які описують роботу фреймворку та інших його компонентів [1];
- Spring Data: модуль для роботи з найпопулярнішими базами даних [15];
- Spring Security: модуль призначений для розробки аутентифікації, авторизації та контролю доступу [14];
- Spring MVC: модуль який призначений для розробки веб-орієнтованих застосунків за допомогою архітектури Model-View-Controller [13].

– Hibernate - фреймворк мови програмування Java, який призначений для вирішення складних задач без написання безліч низькорівневого коду який описує сам фреймворк, реалізований та написаний на Java для вирішення проблем зв’язаних з базами даних [7];

– PostgreSQL - СКБД (система керування базами даних),реалізований та розроблений для ефективного додавання, редагування, та видалення з даних з баз даних [17];

– HTML - Мова гіпертекстової розмітки (HyperText Markup Language) для написання візуальної частини веб-сайтів, ОП отримують html файли за допомогою протоколів http та https або відкриваються з носія користувача, де розробляється сама ОП [11] ;

– CSS (Cascading Style Sheets) - мова, за допомогою якої розробники роблять зовнішній вид html файлів більш привабливим для користувачів. Призначений для HTML та легко імпортується в любі проекти зав’язані на проектуванні веб-сайтів [12];

– Thymeleaf - фреймворк написаний на мові програмування Java, призначений для зручної обробки даних html, css, text, javascript, xml файлів для скорочення необхідної роботи для реалізації функцій [4];

– Lombok - фреймворк призначений для скорочення написання однотипного коду, за допомогою Java анотацій, за допомогою нього Java-розробники можуть не писати getter/setter методи, конструктори та інше;

– Maven - інструмент для створення автоматизованої архітектури проектів написаних на мові програмування Java та описання їх структури (додавання фреймворків) за допомогою xml файлу [6].

2.4. Опис структури системи та алгоритмів її функціонування

Структура навчальної ОП для організації навчального процесу за використанням мови програмування Java.

Структура проекту навчальної ОП складається з:

– пакету Config, за допомогою файлів якого проект набуває необхідного функціоналу для роботи системи:

SecurityConfig.java: Java файл, в якому описується робота авторизації за допомогою метода filterChain (HttpSecurity http) (рис.2.1), шифрування паролю в базі даних за допомогою методу passwordEncoder() та методу налаштування configure (AuthenticationManagerBuilder builder) (рис.2.2).

```
@Bean
public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
    http.csrf().disable() HttpSecurity
            .authorizeRequests() ExpressionInterceptUrlRegistry
            .requestMatchers("/register", "register/save", "/courses" ,"/login", "/css/**", "/js/**") AuthorizedUrl
            .permitAll() ExpressionInterceptUrlRegistry
            .anyRequest().authenticated()
            .and() HttpSecurity
            .formLogin(form -> form
                    .loginPage("/register")
                    .defaultSuccessUrl("/courses")
                    .loginPage("/login")
                    .defaultSuccessUrl("/courses")
                    .loginProcessingUrl("/login")
                    .failureUrl( authenticationFailureUrl: "/login?error=true")
                    .permitAll()
            ).logout(
                    logout -> logout
                            .logoutRequestMatcher(new AntPathRequestMatcher("/logout"))
                            .logoutSuccessUrl("/login?logout")
                            .permitAll());

    return http.build();
}
```

Рис.2.1. Метод filterChain файлу SecurityConfig.java

```java
@Bean
public static PasswordEncoder passwordEncoder() {
    return new BCryptPasswordEncoder();
}

no usages
public void configure(AuthenticationManagerBuilder builder) throws Exception {
    builder.userDetailsService(myUserDetailsService).passwordEncoder(passwordEncoder());
}
```

Рис.2.2. Метод passwordEncoder файлу SecurityConfig.java та його метод налаштування

SecurityUtil.java: Java файл, в якому описуються кастомні методи для отримання інформації за допомогою SpringSecurity, метод getSessionUser() дозволяє програмі отримувати ім’я користувача який знаходиться на сторінці (рис.2.3).

```java
3 usages
public static String getSessionUser(){
    Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
    if(!(authentication instanceof AnonymousAuthenticationToken)){
        String currentUsername = authentication.getName();
        return currentUsername;
    }
    return null;
}
```

Рис.2.3. Метод getSessionUser файлу SecurityUtil.java

– пакету Controller, за допомогою файлів якого програма набуває необхідного функціоналу для відкривання сторінок сайту:

AuthController.java: Java файл, в якому описуються методи для успішної реєстрації та входу до сайту користувачів, get-метод loginPage() який видає html сторінки для входу до системи та post-метод loginSubmit (String username, String password, Model model) який відправляє дані введені користувачем у формі після натискання кнопки login (рис.2.4), також, get-методу getRegisterForm (Model model) який відкриває сторінку сайту з формою для реєстрації нового користувача та post-методу register (RegistrationDto user, BindingResult result,

Model model) який відправляє дані для реєстрації нового користувача після натискання кнопки register (рис.2.5), та методу logoutPage() для успішного виходу з системи користувачу (рис.2.6).

```
@GetMapping("/login")
public String loginPage() { return "login"; }

@PostMapping("/login")
public String loginSubmit(@RequestParam String username, @RequestParam String password, Model model) {
    UserDetails userDetails = myUserDetailsService.loadUserByUsername(username);

    if (userDetails == null) {
        return "redirect:/error";
    }

    if ("validUsername".equals(username) && "validPassword".equals(password)) {
        model.addAttribute( attributeName: "user", userDetails);
        return "redirect:/courses";
    }
    model.addAttribute( attributeName: "user", userDetails);
    return "redirect:/courses";
}
```

Рис.2.4. Методи loginPage та loginSubmit файлу AuthController.java

```
@GetMapping("/register")
public String getRegisterForm(Model model) {
    RegistrationDto user = new RegistrationDto();
    model.addAttribute( attributeName: "user", user);
    return "register";
}

@PostMapping("/register/save")
public String register(@Valid @ModelAttribute("user") RegistrationDto user,
                       BindingResult result, Model model) {
    UserEntity existingUserEmail = userService.findByEmail(user.getEmail());
    if (existingUserEmail != null && existingUserEmail.getEmail() != null && !existingUserEmail.getEmail().isEmpty()) {
        return "redirect:/register?fail";
    }
    UserEntity existingUserUsername = userService.findByUsername(user.getUsername());
    if (existingUserUsername != null && existingUserUsername.getUsername() != null && !existingUserUsername.getUsername().isEmpty()) {
        return "redirect:/register?fail";
    }
    if (result.hasErrors()) {
        model.addAttribute( attributeName: "user", user);
        return "register";
    }
    userService.saveUser(user);
    return "redirect:/login?success";
}
}
```

Рис.2.5. Методи getRegisterForm та register файлу AuthController.java

Рис.2.6. Метод logoutPage файлу AuthController.java

CourseController.java: Java файл, в якому описуються методи для відображення сторінок пов'язаних з entity Course. Get-метод editCourseForm (Long courseId, Model model) та його post-метод updateCourse (Long courseId, CourseDto courseDto, BindingResult result, Model model) призначені для відкривання сторінки зі формою для редагування Course та відправляння даних до проекту для успішного додавання виправлених даних до бази даних (рис.2.7). Get-метод searchCourse (String query, Model model) призначений для відображення всіх Course за допомогою ключового слова (рис.2.8). Post-метод deleteCourse (Long courseId) призначений для того щоб видаляти Course за ідентифікатором coursed (рис.2.9). Get-метод createCourseForm (Model model) та Post-метод saveCourse (CourseDto courseDto, BindingResult result, Model model) призначені для відображення сторінки з формою для додавання нового Course та відправляння усіх даних до бази даних після натискання кнопки Create (рис.2.10). Get-метод courseDetail (Long courseId, Model model) призначений для відображення детальної інформації щодо Course (рис.2.11). Get-метод listCourse (Model model) призначений для відображення сторінки з усіма Course в базі даних (рис.2.12). Get-метод userRegistration (Long courseId, Long userId, Model model) та post-метод registerUserOnCourse (Long courseId, Long userId) призначені для успішного реєстрування користувача на Course (рис.2.13). Get-метод allRegisteredUsers (Long courseId, Model model) призначенний для відображення сторінки з усіма користувачами які зареєстровані на Course (рис.2.14).

```
@GetMapping("/courses/{courseId}/edit")
public String editCourseForm(@PathVariable("courseId") Long courseId, Model model){
    CourseDto courseDto = courseService.findCourseById(courseId);
    model.addAttribute( attributeName: "course", courseDto);
    return "courses-edit";
}
@PostMapping("/courses/{courseId}/edit")
public String updateCourse(@PathVariable("courseId") Long courseId,
                           @Valid @ModelAttribute("course") CourseDto courseDto,
                           BindingResult result, Model model){
    if(result.hasErrors()){
        model.addAttribute( attributeName: "course", courseDto);
        return "courses-edit";
    }
    courseDto.setId(courseId);
    courseService.updateCourse(courseDto);
    return "redirect:/courses";
}
```

Рис.2.7. Методи editCourseForm та updateCourse файлу CourseController.java

```
@GetMapping("/courses/search")
public String searchCourse(@RequestParam(value = "query")String query,
                           Model model){
    List<CourseDto> courses = courseService.searchCourses(query);
    model.addAttribute( attributeName: "courses", courses);
    return "courses-list";
}
```

Рис.2.8. Метод searchCourse файлу CourseController.java

```
@PostMapping("/courses/{courseId}/delete")
public String deleteCourse(@PathVariable("courseId") Long courseId){
    courseService.delete(courseId);
    return "redirect:/courses";
}
```

Рис.2.9. Метод deleteCourse файлу CourseController.java

```
@GetMapping("/courses/new")
public String createCourseForm(Model model){
    Course course = new Course();
    model.addAttribute( attributeName: "course", course);
    return "course-create";
}
@PostMapping("/courses/new")
public String saveCourse(@Valid @ModelAttribute("course")CourseDto courseDto,
                         BindingResult result, Model model){
    if(result.hasErrors()){
        model.addAttribute( attributeName: "course", courseDto);
        return "course-create";
    }
    courseService.saveCourse(courseDto);
    return "redirect:/courses";
}
```

Рис.2.10. Методи createCourseForm та saveCourse файлу CourseController.java

```
@GetMapping("/courses/{courseId}")
public String courseDetail(@PathVariable("courseId") Long courseId,
                           Model model){
    Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
    String username = authentication.getName();
    UserEntity userEntity = userService.findByUsername(username);
    CourseDto courseDto = courseService.findCourseById(courseId);
    boolean isRegistered = courseService.isRegisteredOnCourse(userEntity.getId(), courseId);
    model.addAttribute( attributeName: "isRegistered", isRegistered);
    model.addAttribute( attributeName: "user", userEntity);
    model.addAttribute( attributeName: "course", courseDto);
    return "courses-detail";
}
```

Рис.2.11. Метод courseDetail файлу CourseController.java

```java
@GetMapping("/courses")
public String listCourses(Model model){
    UserEntity userEntity = new UserEntity();
    List<CourseDto> courses = courseService.findAllCourses();
    String username = SecurityUtil.getSessionUser();
    if(username != null){
        userEntity = userService.findByUsername(username);
        model.addAttribute( attributeName: "user", userEntity);
    }
    model.addAttribute( attributeName: "user", userEntity);
    model.addAttribute( attributeName: "courses", courses);
    return "courses-list";
}
```

Рис.2.12. Метод listCourses файлу CourseController.java

```java
@GetMapping("/courses/{courseId}/{userId}/registrationOnCourse")
public String userRegistration(@PathVariable("courseId") Long courseId,
                               @PathVariable("userId")Long userId,
                               Model model){
    String username = userService.findUserById(userId).getUsername();
    UserEntity userEntity = userService.findByUsername(username);
    CourseDto courseDto = courseService.findCourseById(courseId);
    model.addAttribute( attributeName: "user", userEntity);
    model.addAttribute( attributeName: "course", courseDto);
    return "register-on-course";
}
@PostMapping("/courses/{courseId}/{userId}/save")
public String registerUserOnCourse(@PathVariable("courseId")Long courseId,
                                   @PathVariable("userId")Long userId
                                   ) {
    if(userService.isAlreadyUserOnCourse(userId, courseId)){
        return "redirect:/error";
    }
    userService.updateUser(userId ,courseId);
    return "success";
}
```

Рис.2.13. Методи userRegistration та registerUserOnCourse файлу CourseController.java

Рис.2.14. Метод allRegisterUsers файлу CourseController.java

MyErrorController.java: java-файл який додає можливості при помилках на боці системи видавати сторінки з описом помилки. Get-метод handleError (HttpServletRequest request) провіряє після запиту на наявність помилок, та якщо вони є відправляє користувача на сторінку з помилкою (рис.2.15).

Рис.2.15. Метод handleError файлу MyErrorController.java

TaskController.java: java-файл, в якому описуються методи для відображення сторінок пов’язаних з entity Task. Get та Post методи editTaskForm

(Long taskId, Long courseId, Model model) та updateTask (Long taskId, Long courseId, TaskDto taskDto, BindingResult result, Model model) додають сайту можливості надання сторінки користувачеві з формою для редагування Task (завдання) та за допомогою post-методу updateTask відправляння даних на обробку та додавання змін у базу даних (рис.2.16). Get та post методи createTaskForm (Long courseId, Model model) та createTask (Long courseId, TaskDto taskDto, BindingResult result, Model model) додають сайту можливості надання користувачеві з формою для додавання Task та за допомогою post-методу createTask відправляння даних для додавання нового об’єкту до бази даних (рис.2.17). Get-метод searchTasks (String query, Model model) відправляє на сторінку всі Task які підходять по назві за ключовим словом (String query) (рис.2.18). Get-метод deleteTask (Long taskId) додає можливості видаляти Task за ідентифікатором taskId (рис.2.19). Get-метод viewTask (Long taskId, Long courseId, Model model) додає можливості сайту за допомогою taskId та courseId відкривати особисту сторінку Task (рис.2.20). Та get-метод taskList (Model model) додає можливості надання сторінки з усіма Task які існують у базі даних (рис.2.21).

```java
@GetMapping("/tasks/{taskId}/{courseId}/edit")
public String editTaskForm(@PathVariable("taskId") Long taskId,
                           @PathVariable("courseId") Long courseId,
                           Model model){
    TaskDto taskDto = taskService.findTaskById(taskId);
    model.addAttribute( attributeName: "courseId",courseId);
    model.addAttribute( attributeName: "task", taskDto);
    return "tasks-edit";
}
@PostMapping("/tasks/{taskId}/{courseId}/edit")
public String updateTask(@PathVariable("taskId") Long taskId,
                         @PathVariable("courseId") Long courseId,
                         @Valid @ModelAttribute("task") TaskDto taskDto,
                         BindingResult result, Model model){
    if(result.hasErrors()){
        model.addAttribute( attributeName: "task", taskDto);
        return "tasks-edit";
    }
    if (courseId == null) {
        model.addAttribute( attributeName: "errorMessage",  attributeValue: "Course ID is required.");
        model.addAttribute( attributeName: "task", taskDto);
        return "tasks-edit";
    }

    taskDto.setId(taskId);

    Course course = Mappers.mapToCourse(courseService.findCourseById(courseId));
    taskDto.setCourse(course);

    taskService.updateTask(taskDto);

    return "redirect:/tasks";
}
```

Рис.2.16. Методи editTaskForm та upadteTask файлу TaskController.java

```java
@GetMapping("/tasks/{courseId}/new")
public String createTaskForm(@PathVariable("courseId") Long courseId,
                             Model model){
    Task task = new Task();
    model.addAttribute( attributeName: "courseId", courseId);
    model.addAttribute( attributeName: "task", task);
    return "tasks-create";
}
@PostMapping("/tasks/{courseId}")
public String createTask(@PathVariable("courseId") Long courseId,
                         @ModelAttribute("task")TaskDto taskDto,
                         BindingResult result,Model model){
    if(result.hasErrors()){
        model.addAttribute( attributeName: "task", taskDto);
        return "tasks-create";
    }
    taskService.createTask(courseId,taskDto);
    return "redirect:/courses";
}
```

Рис.2.17. Методи createTaskForm та createTask файлу TaskController.java

```java
@GetMapping("/tasks/search")
public String searchTasks(@RequestParam(value = "query")String query,
                          Model model){
    List<CourseDto> courses = courseService.findAllCourses();
    List<TaskDto> tasks = taskService.searchTasks(query);
    model.addAttribute( attributeName: "courses", courses);
    model.addAttribute( attributeName: "tasks", tasks);
    return "tasks-list";
}
```

Рис.2.18. Метод searchTasks файлу TaskController.java

```java
@GetMapping("/tasks/{taskId}/delete")
public String deleteTask(@PathVariable("taskId") Long taskId){
    taskService.delete(taskId);
    return "redirect:/tasks";
}
```

Рис.2.19. Метод deleteTask файлу TaskController.java

```
@GetMapping("/tasks/{taskId}/{courseId}")
public String viewTask(@PathVariable("taskId") Long taskId,
                       @PathVariable("courseId") Long courseId,
                       Model model){
    TaskDto taskDto = taskService.findTaskById(taskId);
    model.addAttribute( attributeName: "courseId", courseId);
    model.addAttribute( attributeName: "task", taskDto);
    return "tasks-detail";
}
```

Рис.2.20. Метод viewTask файлу TaskController.java

```
@GetMapping("/tasks")
public String taskList(Model model){
    List<CourseDto> courses = courseService.findAllCourses();
    List<TaskDto> tasks = taskService.findAllTasks();
    model.addAttribute( attributeName: "tasks", tasks);
    model.addAttribute( attributeName: "courses", courses);
    return "tasks-list";
}
```

Рис.2.21. Метод taskList файлу TaskController.java

UserDetailController.java: java-файл, в якому описуються методи для роботи кабінету користувача, додавання, редагування, видалення оцінок та їх перегляду. Get-метод editLogOfCoursePage (Long courseId, Model model) додає можливості сайту для надання користувачу (тільки вчителям) редагувати створені ними курси, оцінки, та завдання (рис.2.22). Get-метод assessmentsOfCourse (Long courseId, Model model) додає можливості надання користувачу переглянути оцінки на курсах на які він зареєстрований (рис.2.23). Get-метод userDetails (Model model) надає можливості переглянути особистий кабінет користувача (рис.2.24). Get та Post методи addUserAssessment (Long userId, Long taskId, Model model) та saveUserAssessment (Long userId, Long taskId, AssessmentDto assessmentDto, BindingResult result, Model model) надають можливості відкривати форму для додавання нової оцінки, та відправляють дані до бази даних за допомогою userId та taskId прив’язують оцінку до користувача

та завдання (рис.2.25). Get та Post методи editUserAssessment (Long userId, Long taskId, Long assessmentId, Model model) та saveEditedUserAssessment (Long userId, Long taskId, Long assessmentId, AssessmentDto assessmentDto, BindingResult result, Model model) надають можливості користувачу відкривати сторінки з формою для редагування оцінки та відправляння редагованої інформації до бази даних (рис.2.26). Get-метод deleteAssessment додає можливості видаляти оцінку з бази даних (рис.2.27).

```
@GetMapping("/editLogOfCourse/{courseId}")
public String editLogOfCoursePage(@PathVariable(value = "courseId")Long courseId,
                                  Model model){
    CourseDto courseDto = courseService.findCourseById(courseId);
    List<TaskDto> tasks = taskService.findTaskByCourse(courseDto);
    List<UserEntity> users = userService.findUsersByCourse(courseId);
    List<AssessmentDto> assessments = assessmentService.findAllAssessments();
    model.addAttribute( attributeName: "assessments", assessments);
    model.addAttribute( attributeName: "users", users);
    model.addAttribute( attributeName: "course", courseDto);
    model.addAttribute( attributeName: "tasks", tasks);
    return "log-of-course";
}
```

Рис.2.22. Метод editLogOfCoursePage файлу UserDetailController.java

```
@GetMapping("/assessments/{courseId}")
public String assessmentsOfCourse(@PathVariable(value = "courseId") Long courseId,
                                  Model model){
    Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
    String username = authentication.getName();
    UserEntity userEntity = userService.findByUsername(username);
    CourseDto courseDto = courseService.findCourseById(courseId);
    List<TaskDto> tasks = taskService.findTaskByCourse(courseDto);
    List<AssessmentDto> assessments = assessmentService.findAllAssessments();
    model.addAttribute( attributeName: "user", userEntity);
    model.addAttribute( attributeName: "assessments", assessments);
    model.addAttribute( attributeName: "course", courseDto);
    model.addAttribute( attributeName: "tasks", tasks);
    return "assessments";
}
```

Рис.2.23. Метод assessmentsOfCourse файлу UserDetailController.java

```
@GetMapping("/userDetails")
public String userDetails(Model model){
    Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
    String userName = authentication.getName();
    UserEntity userEntity = userService.findByUsername(userName);
    model.addAttribute( attributeName: "user", userEntity);
    return "user-detail";
}
```

Рис.2.24. Метод userDetails файлу UserDetailController.java

```
@GetMapping("/addAssessment/{userId}/{taskId}/new")
public String addUserAssessment(@PathVariable(value = "userId") Long userId,
                                @PathVariable(value = "taskId")Long taskId,
                                Model model){
    Assessment assessment = new Assessment();
    model.addAttribute( attributeName: "assessment", assessment);
    model.addAttribute( attributeName: "userId", userId);
    model.addAttribute( attributeName: "taskId", taskId);
    return "assessment-create";
}
@PostMapping("/addAssessment/{userId}/{taskId}")
public String saveUserAssessment(@PathVariable(value = "userId") Long userId,
                                @PathVariable(value = "taskId")Long taskId,
                                @ModelAttribute("assessment") AssessmentDto assessmentDto,
                                BindingResult result,
                                Model model){
    if(result.hasErrors()){
        model.addAttribute( attributeName: "assessment", assessmentDto);
        return "assessment-create";
    }
    assessmentService.createAssessment(userId, taskId, assessmentDto);
    return "redirect:/userDetails";
}
```

Рис.2.25. Методи addUserAssessment та saveUserAssessment файлу UserDetailController.java

```
@GetMapping("/addAssessment/{userId}/{taskId}/edit/{assessmentId}")
public String editUserAssessment(@PathVariable(value = "userId") Long userId,
                                 @PathVariable(value = "taskId")Long taskId,
                                 @PathVariable(value = "assessmentId")Long assessmentId,
                                 Model model){
    AssessmentDto assessmentDto = assessmentService.findAssessmentById(assessmentId);
    model.addAttribute( attributeName: "assessment", assessmentDto);
    return "assessment-edit";
}
@PostMapping("/addAssessment/{userId}/{taskId}/edit/{assessmentId}/save")
public String saveEditedUserAssessment(@PathVariable(value = "userId") Long userId,
                                       @PathVariable(value = "taskId")Long taskId,
                                       @PathVariable(value = "assessmentId")Long assessmentId,
                                       @Valid @ModelAttribute("assessment") AssessmentDto assessmentDto,
                                       BindingResult result, Model model){
    if(result.hasErrors()){
        model.addAttribute( attributeName: "assessment", assessmentDto);
        return "assessment-edit";
    }
    AssessmentDto assessmentDto1 = assessmentService.findAssessmentById(assessmentId);
    assessmentDto.setId(assessmentId);
    assessmentDto.setUser(userService.findUserById(userId));
    assessmentDto.setTask(Mappers.mapToTask(taskService.findTaskById(taskId)));
    assessmentService.updateAssessment(assessmentDto);
    return "redirect:/editLogOfCourse/" + assessmentDto.getTask().getCourse().getId();
}
```

Рис.2.26. Методи editUserAssessment та saveEditedUserAssessment файлу UserDetailController.java

```
@GetMapping("/addAssessment/{userId}/{taskId}/delete/{assessmentId}")
public String deleteUserAssessment(@PathVariable(value = "userId") Long userId,
                                   @PathVariable(value = "taskId")Long taskId,
                                   @PathVariable(value = "assessmentId")Long assessmentId,
                                   Model model){
    assessmentService.deleteAssessment(assessmentId);
    return "redirect:/userDetails";
}
```

Рис.2.27. Метод deleteUserAssessment файлу UserDetailController.java

– пакету Models, за допомогою файлів якого до програми надають зрозуміння сутності Task, Course, Asessment, UserEntity, UserCourse та Role. У кожної з цих сутностей існує унікальний ідентифікатор Id за допомогою якого легко та зручно діставати дані з бази даних та використовувати їх для потреб програми (рис.2.28):

Assessment.java: java-файл сутності оцінок в якому є зміні для роботи самої оцінки:

1. Long id – унікальний ідентифікатор.
2. int assessment – числове значення оцінки.
3. String comment – коментар про оцінку.
4. LocalDateTime createdOn – час коли була створена оцінка.
5. LocalDateTime updatedOn – час коли оцінка була відредагована.
6. Task task – завдання к якому відноситься оцінка.
7. UserEntity userEntity – користувач к якому відноситься оцінка.

```
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
@Entity
public class Assessment {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private int assessment;
    private String comment;
    @CreationTimestamp
    private LocalDateTime createdOn;
    @UpdateTimestamp
    private LocalDateTime updatedOn;
    @ManyToOne
    @JoinColumn(name = "task_id", nullable = false)
    private Task task;

    @ManyToOne
    @JoinColumn(name = "user_id", nullable = false)
    private UserEntity user;
}
```

Рис.2.28. Клас сутності Assessment (оцінка) файлу Assessment.java

Course.java: java-файл сутності курсів в якому є зміні для роботи курсів (рис.2.29):

1. Long id – унікальний ідентифікатор.

2. String title – назва курсу.
3. String photoUrl – посилання на логотип курсу.
4. String content – короткий коментар про курс.
5. LocalDateTime createdOn – час коли був створений курс.
6. LocalDateTime updatedOn – час коли курс був відредагований.
7. UserEntity createdBy – користувач який створив курс.
8. List<Task> tasks – завдання які належать курсу.
9. List<UserEntity> users – користувачі які були зареєстровані на курс.

```java
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
@Entity
@Table(name = "courses")
public class Course {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String title;
    private String photoUrl;
    private String content;
    @CreationTimestamp
    private LocalDateTime createOn;
    @UpdateTimestamp
    private LocalDateTime updatedOn;
    @ManyToOne
    @JoinColumn(name = "created_by")
    private UserEntity createdBy;
    @OneToMany(mappedBy = "course", cascade = CascadeType.REMOVE)
    private List<Task> tasks = new ArrayList<>();
    @ManyToMany(mappedBy = "courses")
    private List<UserEntity> users;
}
```

Рис.2.29. Клас сутності Course (курс) файлу Course.java

Role.java: java-файл сутності привілей користувачів за допомогою яких користувачі набувають необхідних функцій на сайті (рис.2.30):

1. Long id – унікальний ідентифікатор.
2. String name – назва привілей.
3. List<UserEntity> users – список користувачів за допомогою якого до них відносяться ролі у базі даних.

```
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Entity(name = "roles")
public class Role {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;

    @ManyToMany(mappedBy = "roles")
    private List<UserEntity> users = new ArrayList<>();
}
```

Рис.2.30. Клас сутності Role (привілеї) файлу Role.java

Task.java: java-файл сутності завдань в якому є зміні для їх роботи (рис.2.31):

1. Long id – унікальний ідентифікатор.
2. String name – назва завдання.
3. String content – короткий коментар про завдання.
4. LocalDateTime startTime – час коли завдання починає приймати роботи студентів.
5. LocalDateTime endTime – час коли завдання закінчує приймати роботи студентів.
6. String photoUrl – посилання на логотип завдання.
7. boolean isDone – зміна яка відображає чи зроблене користувачем завдання.
8. LocalDateTime createdOn – час коли було створено завдання.
9. LocalDateTime updatedOn – час коли завдання було відредаговано.

10. Course course – курс до якого відносяться завдання.
11. List<Assessment> assessments – список оцінок користувачів які відносяться до завдання.

```java
@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
@Entity
public class Task {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    private String content;
    private LocalDateTime startTime;
    private LocalDateTime endTime;
    private String photoUrl;
    private boolean isDone;
    @CreationTimestamp
    private LocalDateTime createdOn;
    @UpdateTimestamp
    private LocalDateTime updatedOn;

    @ManyToOne
    @JoinColumn(name = "course_id", nullable = true)
    private Course course;
    @OneToMany(mappedBy = "task", cascade = CascadeType.ALL, orphanRemoval = true)
    private List<Assessment> assessments = new ArrayList<>();

}
```

Рис.2.31. Клас сутності Task (завдання) файлу Task.java

UserCourse.java: java-файл в якому користувач відноситься до курсу для зручної обробки даних по цим показниками(рис.2.32):

1. Long id – унікальний ідентифікатор.
2. UserEntity user – користувач.
3. Course course – курс.

Рис.2.32. Клас сутності UserCourse (користувач до курсу) файлу UserCourse.java

UserEntity.java: java-файл в якому надаються всі зміні для створення таблиць в базах даних та для зручного функціонування користувача(рис.2.33):

1. Long id – унікальний ідентифікатор.
2. String username – і’мя користувача.
3. String email – корпоративна пошта користувача.
4. String password – пароль користувача.
5. List<Role> roles – список усіх ролей які існують в ОП, для того щоб користувачу змогли їх змінити в будь який момент.
6. List<Course> course – список курсів на які користувач зареєстрований.
7. List<Assessment> assessments – список оцінок які відносяться до користувача.

```java
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Entity(name = "users")
public class UserEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String username;
    private String email;
    private String password;
    @ManyToMany(fetch = FetchType.LAZY, cascade = CascadeType.ALL)
    @JoinTable(name = "users_roles",
            joinColumns = {@JoinColumn(name = "user_id", referencedColumnName = "id")},
            inverseJoinColumns = {@JoinColumn(name = "role_id", referencedColumnName = "id")})
    private List<Role> roles = new ArrayList<>();
    @ManyToMany(fetch = FetchType.LAZY, cascade = CascadeType.ALL)
    @JoinTable(name = "users_courses",
                joinColumns = {@JoinColumn(name="user_id", referencedColumnName = "id")},
                inverseJoinColumns = {@JoinColumn(name = "course_id", referencedColumnName = "id")})
    private List<Course> courses = new ArrayList<>();
    @OneToMany(mappedBy = "user", cascade = CascadeType.ALL, orphanRemoval = true)
    private List<Assessment> assessments = new ArrayList<>();
}
```

Рис.2.33. Клас сутності UserEntity (користувач) файлу UserEntity.java

– пакету Repository, за допомогою якого та фреймворка Hibernate розробник може використовувати дефолтні методи писати кастомні для зручного використання даних з бази даних та реалізовувати їх в класах ServiceImpl (клас реалізації):

AssessmentsRepository.java – файл репозиторію за допомогою якого програмі надається функціоналу для отримання та подальшого додавання, редагування, видалення даних оцінок з таблиць баз даних (рис.2.33).

```java
public interface  AssessmentRepository extends JpaRepository<Assessment,Long> {

}
```

Рис.2.33. Інтерфейс репозиторію сутності Assessment (оцінка) файлу AssessmentRepository.java

CourseRepository.java – файл репозиторію за допомогою якого програмі надається функціонал для отримання та подальшого додавання, редагування, видалення даних курсів з таблиць баз даних. Метод findByTitle (String url) надає функціоналу пошуку в базі даних курсів по змінній Title. Метод searchCourses (String query) за допомогою анотації яка надається фреймворком Hibernate надається функціоналу пошуку в базі даних за ключовим словом який відноситься до Title (рис.2.34).

```
9 usages
public interface CourseRepository extends JpaRepository<Course, Long> {
    no usages
    Optional<Course> findByTitle(String url);
    1 usage
    @Query("SELECT c from Course c WHERE lower(c.title) LIKE concat('%', lower(:query), '%') ")
    List<Course> searchCourses(String query);
}
```

Рис.2.34. Інтерфейс репозиторію сутності Course (курс) файлу CourseRepository.java

RoleRepository.java – файл репозиторію за допомогою якого програмі надається функціонал для отримання та подальшого додавання, редагування, видалення даних привілей з таблиць баз даних. Метод findByName (String name) додає можливості шукати Role по змінній name (рис.2.35).

```
3 usages
public interface RoleRepository extends JpaRepository<Role, Long> {
    2 usages
    Role findByName(String name);

}
```

Рис.2.35. Інтерфейс репозиторію сутності Role (привілеї) файлу RoleRepository.java

TaskRepository.java – файл репозиторію за допомогою якого програмі надається функціонал для отримання та подальшого додавання, редагування, видалення даних завдань з таблиць баз даних. Метод findByName (String name)

додає можливості шукати Task по змінній name. Метод searchTasks (String query) за допомогою анотації яка надається фреймворком Hibernate надається функціоналу пошуку в базі даних за ключовим словом який відноситься до Name. Метод findTaskByCourse (Course course) надає можливості шукати завдання за курсом (рис.2.36).

```
public interface TaskRepository extends JpaRepository<Task, Long> {
    no usages
    Optional<Task> findByName(String url);
    1 usage
    @Query("SELECT c from Task c WHERE lower(c.name) LIKE concat('%', lower(:query), '%') ")
    List<Task> searchTasks(String query);

    1 usage
    List<Task> findTaskByCourse(Course course);
}
```

Рис.2.36. Інтерфейс репозиторію сутності Task (завдання) файлу TaskRepository.java

UserCourse.java – файл репозиторію за допомогою якого програмі надається функціонал для отримання та подальшого додавання, редагування, видалення даних відношень користувача до курсу. Метод deleteByCourseId (Long courseId) видаляє це відношення по ключовому ідентифікатору coursed (рис.2.37).

```
public interface UserCourseRepository extends JpaRepository<UserCourse, Long> {
    1 usage
    void deleteByCourseId(Long courseId);
}
```

Рис.2.37. Інтерфейс репозиторію сутності UserCourse (відношення користувача до курсу) файлу UserCourseRepository.java

UserCourse.java – файл репозиторію за допомогою якого програмі надається функціонал для отримання та подальшого додавання, редагування, видалення даних користувача з баз даних. Метод findByEmail (Long courseId) шукає користувача за корпоративною поштою. Метод findByUsername (StringUsername) шукає користувача за його нікнеймом. Метод findById (Long

userId) шукає користувача за ідентифікатором. Метод existsByIdAndCourses_id (Long userId, Long courseId) перевіряє чи існує користувач за двома ідентифікаторами. Метод findUserEntitiesByCourses_id (Long courseId) шукає усіх користувачів які зареєстровані на курс (рис.2.38).

```
public interface UserRepository extends JpaRepository<UserEntity, Long>{
    2 usages
    UserEntity findByEmail(String email);
    4 usages
    UserEntity findByUsername(String username);
    2 usages
    UserEntity findFirstByUsername(String username);
    8 usages
    Optional<UserEntity> findById(Long userId);
    1 usage
    boolean existsByIdAndCourses_id(Long userId, Long courseId);

    1 usage
    List<UserEntity> findUserEntitiesByCourses_id(Long courseId);
}
```

Рис.2.38. Інтерфейс репозиторію сутності UserEntity (користувач) файлу UserRepository.java

Фінальна структура проекту:

Пакет config:

1. SecurityConfig.
2. SecurityUtil.

Пакет controller:

1. AuthController.
2. CourseController.
3. MyErrorController.
4. TaskController.
5. UserDetailController.

Пакет dto:

1. AssessmentDto.

2. CourseDto.
3. RegistrationDto.
4. TaskDto.

Пакет models:

1. Assessment.
2. Course.
3. Role.
4. Task.
5. UserCourse.
6. UserEntity.

Пакет repository:

1. AssessmentRepository.
2. CourseRepository.
3. RoleRepository.
4. TaskRepository.
5. UserCourseRepository.
6. UserRepository.

Пакет service:

1. Пакет impl (реалізація):
   - AssessmentServiceImpl.
   - CourseServiceImpl.
   - Mappers.
   - MyUserDetailService.
   - TaskServiceImpl.
   - UserServiceImpl.
2. AssessmentService.
3. CourseService.
4. TaskService.
5. UserService.

2.5. Обґрунтування та організація вхідних та вихідних даних програми

Програма навчальної ОП для організації навчального процесу з використанням мови програмування Java може отримувати дані з баз даних, з форм для введення даних, та юрл-посилань на сторінках у вигляді індентифікаторів (userId, courseId, taskId тощо).

Вхідні дані поділяються на:

– список курсів;

– список завдань;

– список оцінок;

– список привілей;

– список вкладених робіт користувача;

– нікнейм, пароль та корпоративна пошта для входу/реєстрації користувача;

– список курсів на які зареєстрований користувач.

Система ОП отримує від користувача дані для входу та реєстрації через спеціальні форми (рис 2.39 - 2.40):

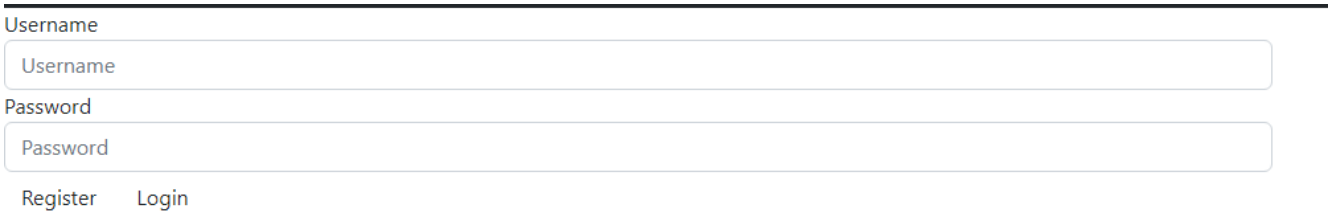


Рис.2.39. Форма входу до системи

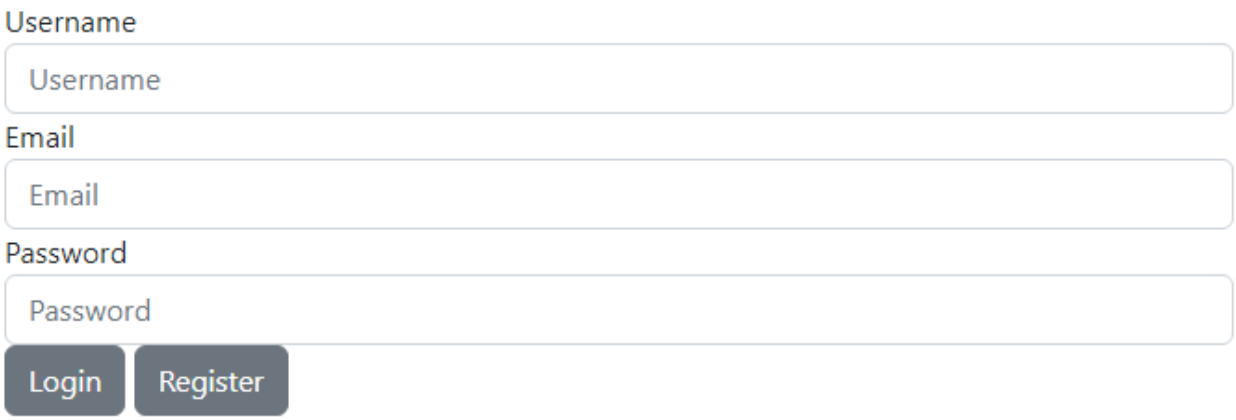


Рис.2.40. Форма реєстрації в системі

Після успішної реєстрації або входу до системи користувача, вихідні дані включать в себе всі сторінки сайту:

– сторінка з усіма курсами;
– сторінка з усіма завданнями;
– сторінка особистого кабінету користувача.

2.6. Опис розробленої системи

2.6.1. Використані технічні засоби

Розроблена навчальна ОП для організації навчального процесу з використанням мови програмування Java була розроблена та протестована на ПК з встановленим Windows 11 та деякими програмними засобами (детальніше пункт 2.6.2).

Характеристики системи:

– процесор: AMD Ryzen 5 5600 6-Core Processor;
– відеоадаптер: NVIDIA GeForce GTX 1650;
– накопичувачі: Apacer AS340 240GB та TOSHIBA HDWD110;
– аудіо картка: Speakers(Razer USB Sound Card);
– оперативна пам’ять: 16 GB DDR4;
– монітор: QUBE Overlord (G24F144PLUS) IPS FullHD 144Hz.

2.6.2. Використані програмні засоби

Розроблена навчальна ОП для організації навчального процесу була розроблена завдяки програмі Intellij IDEA Ultimate version 2023.3.2.

Intellij IDEA – програмний засіб для програмування мовою Java розроблений компанією JetBrains. Має зручний та зрозумілий інтерфейс, підтримку безліч плагінів для облегшення роботи розробника та підтримку багатьох фреймворків, наприклад такі як Spring boot, hibernate та підтримує

підключення до проекту таких найпопулярніших СКБД такі як MySQL, PostgreSQL та інші (рис 2.41). Має зручну систему керування базою даних не виходячи до спеціалізованих програм (рис 2.42).

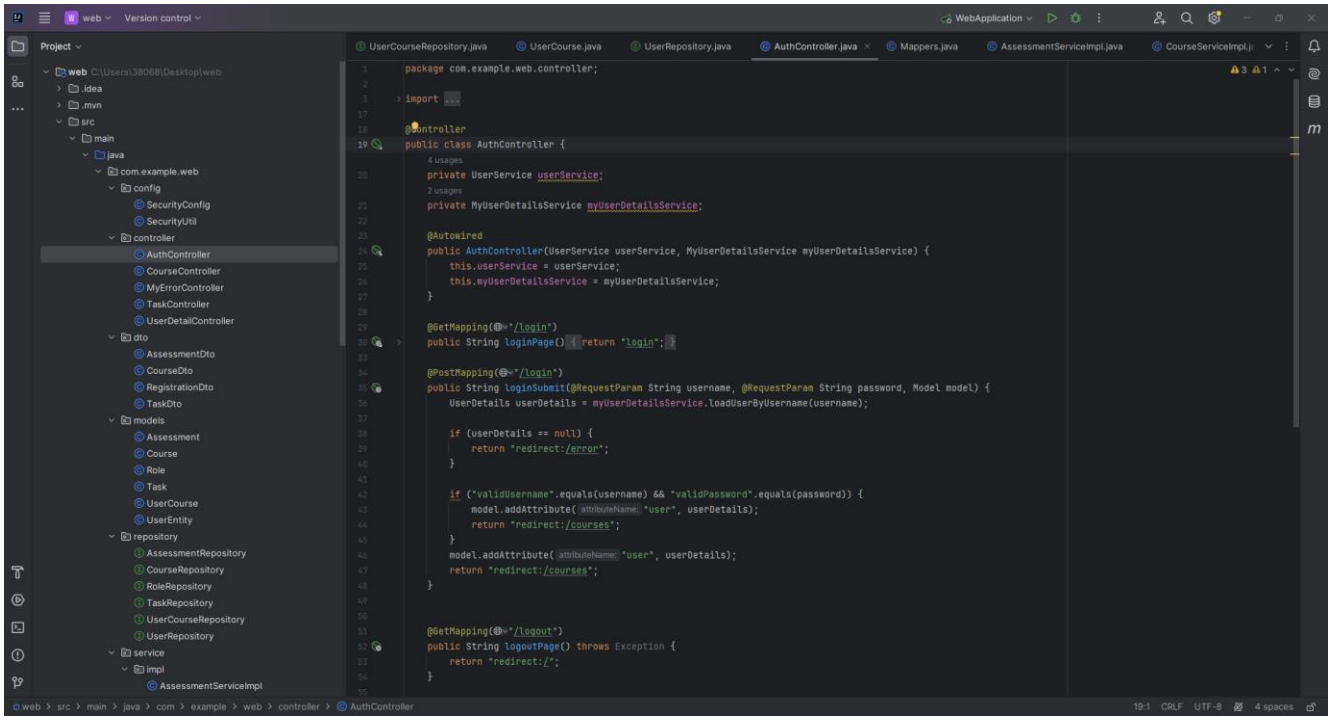


Рис.2.41. Інтерфейс керування Intellij IDEA

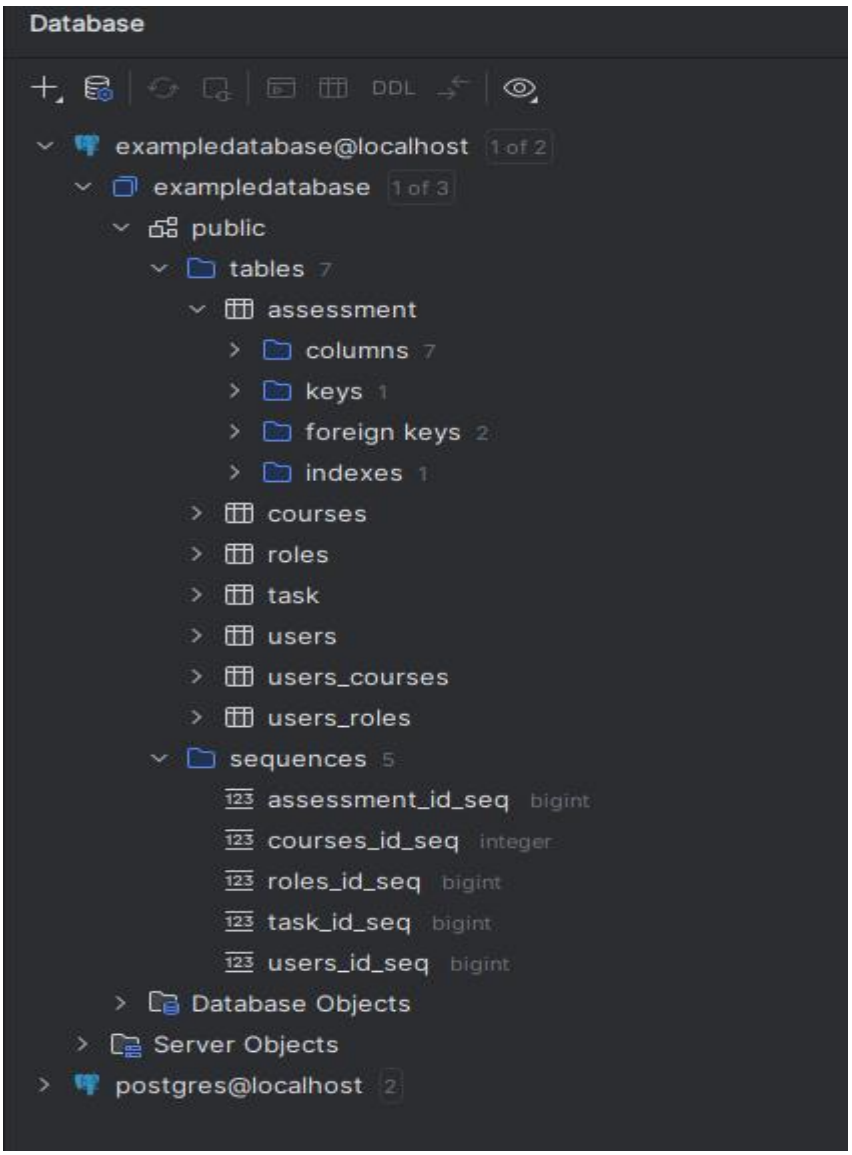


Рис.2.42. Інтерфейс керування базою даних підключеної до проекту в Intellij IDEA

Також має підтримку гіпертекстової мови розмітки HTML (рис 2.42) та мовою деталізації CSS (рис 2.43):

```
sub,
sup {
  position: relative;
  font-size: 0.75em;
  line-height: 0;
  vertical-align: baseline;
}
```

Рис.2.43. Приклад CSS файлу в Intellij IDEA

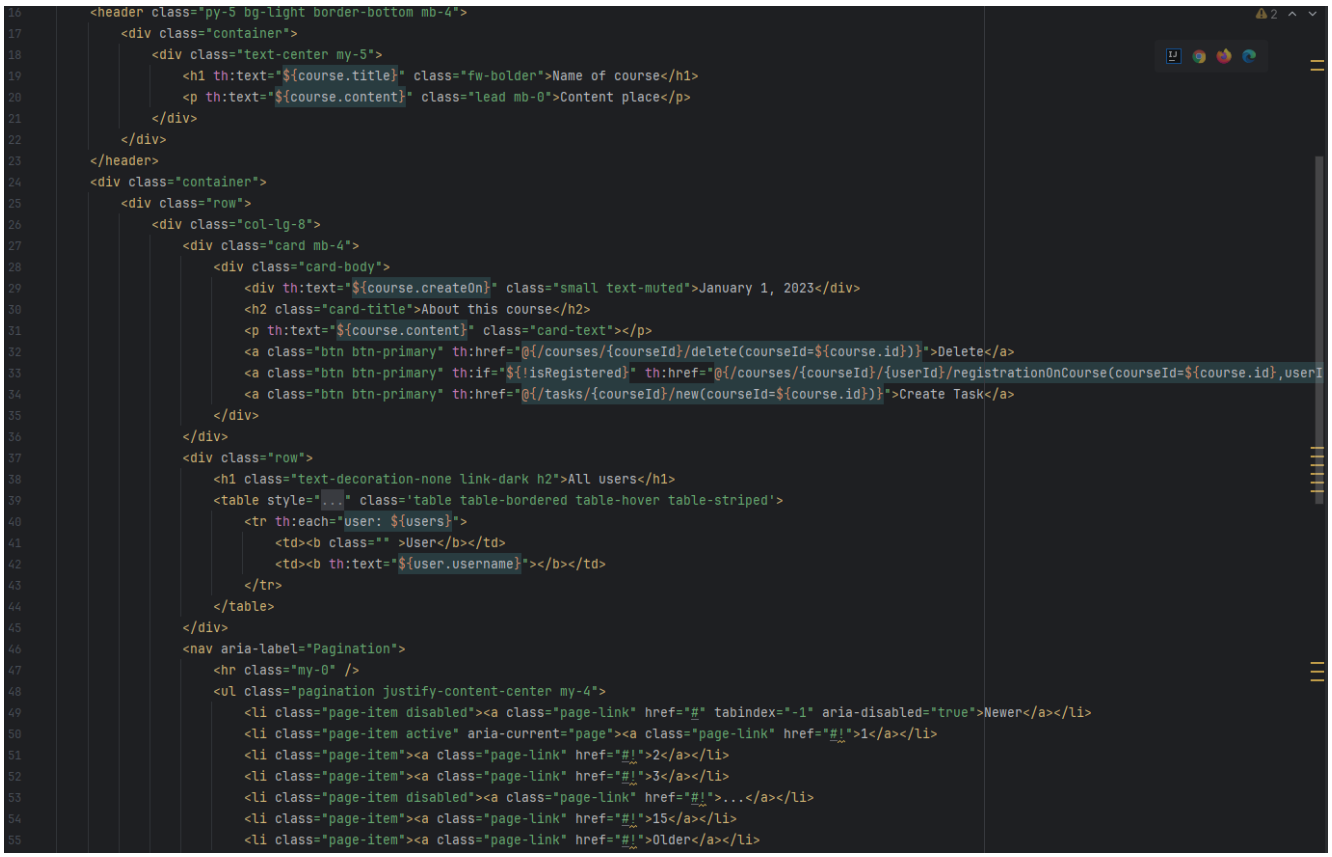


Рис.2.44. Приклад HTML файлу в Intellij IDEA

2.6.3. Виклик та завантаження програми

Для коректної роботи проекту розробленої навчальної ОП для організації навчального процесу потрібно мати такі встановлені програмні засоби:

– Java JDK 17+;

– Intellij IDEA;

– PostgreSQL.

Все інше, наприклад Spring boot або Hibernate встановлені до проекту та не потребують додаткового скачування.

Після успішного скачування програмних засобів потрібно запустити Intellij IDEA та імпортувати проект наданий в цій кваліфікаційній роботі. Це можна зробити завдяки цьому шляху в інтерфейсу Intellij IDEA: File → Open… → “Та вибрати папку з файлами проекту”.

Після чого інтерфейс керування Intellij IDEA буде мати вигляд зазначений на рис 2.45.

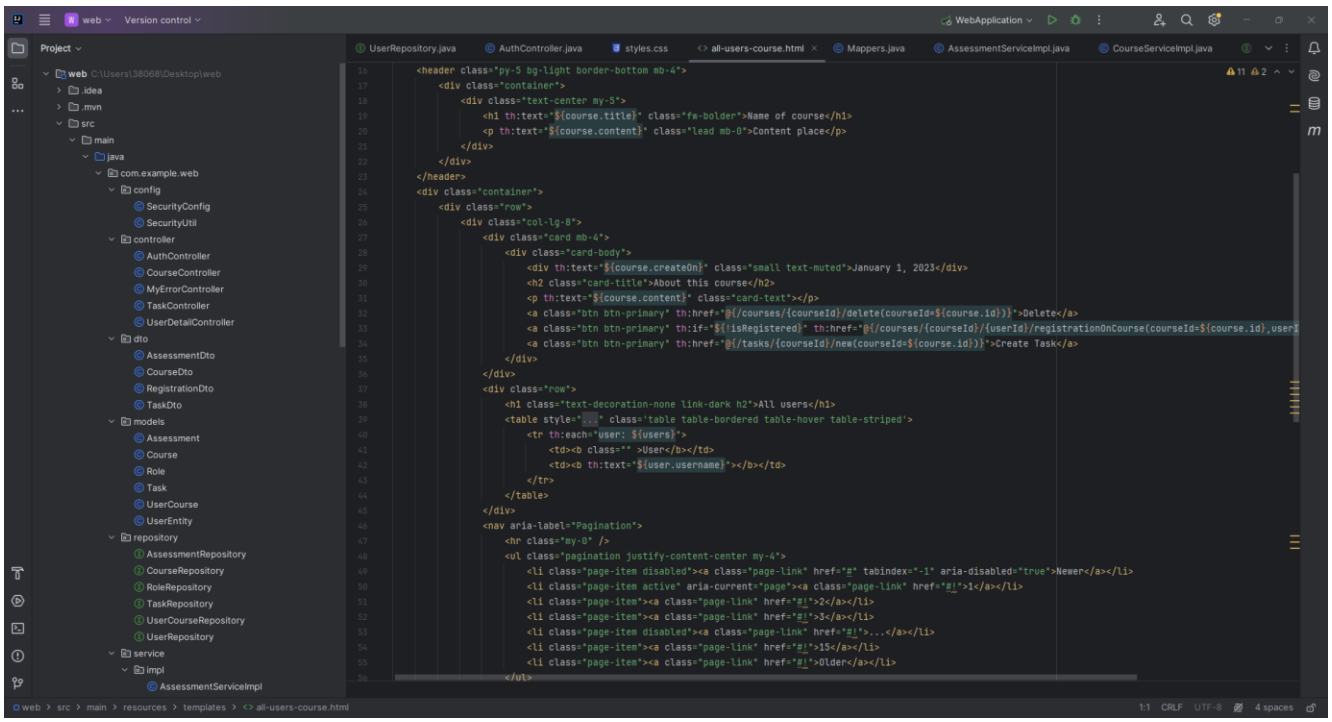


Рис.2.45. Вигляд Intellij IDEA після успішного імпортування проекту

Після всіх необхідних етапів для запуску програми потрібно натиснути → в правому верхньому куточку екрану (рис 2.46):

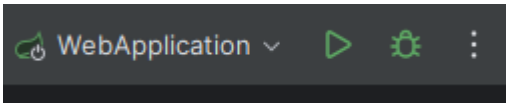


Рис.2.46. Вигляд кнопки для запуску програми в Intellij IDEA

Програма почне запускати проект та напише в консолі інформацію щодо успішного запуску проекту (рис 2.47):

```
Started WebApplication in 4.519 seconds (process running for 5.175)
```

Рис.2.47. Вигляд інформації щодо успішного запуску проекту

Далі потрібно запустити будь-який веб-браузер, в цьому випадку це буде Google Chrome та написати в пошуковому рядку localhost:8080/courses та відправити запит, після чого в нас успішно буде завантажена головна сторінка навчальної ОП для організації навчального процесу (рис 2.48):

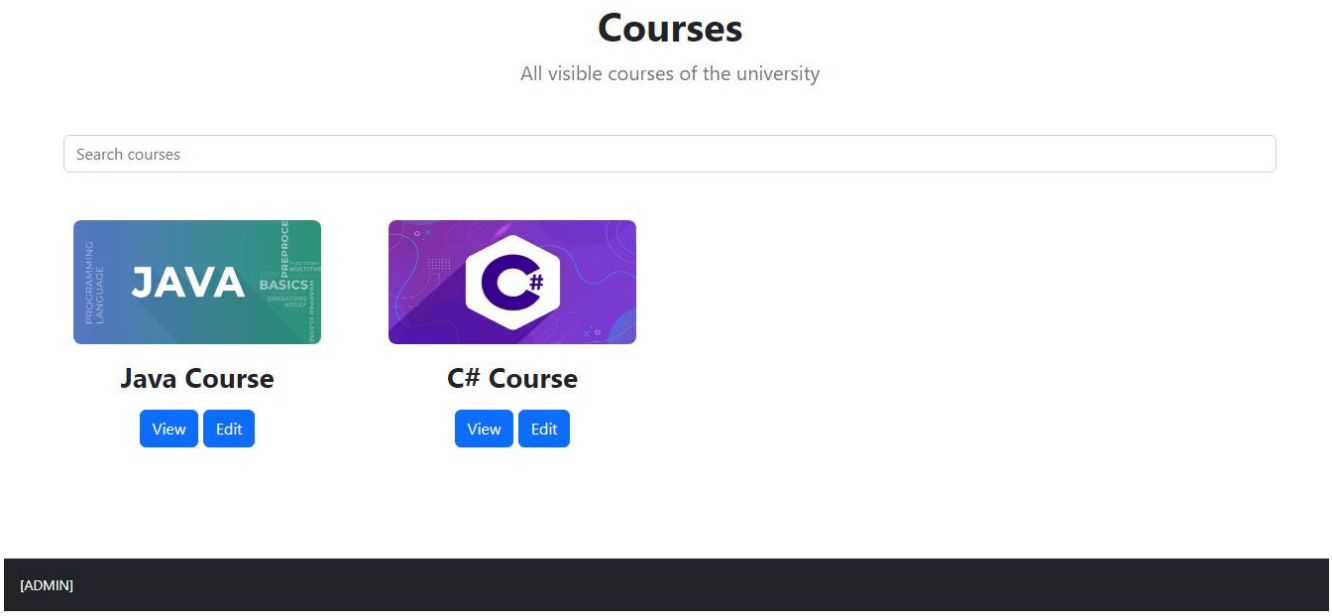


Рис.2.48. Головна сторінка навчальної ОП

2.6.4. Опис інтерфейсу користувача

Навчальна ОП для організації навчального процесу з використанням мови програмування Java має головну сторінку localhost:8080/courses зазначену в пункті 2.6.3 (рис 2.48).

Без реєстрації користувачу доступні тільки головна сторінка сайту з усіма курсами навчальної ОП та сторінки з формами реєстрації та входу до системи зазначені в пункті 2.5 (рис 2.39 - 2.40).

Для доступу користувачеві потрібно здійснити вхід (якщо у користувача вже є аккаунт) або зареєструватися до ОП. Для цього користувачеві потрібно на головній панелі сайту (хедері) знайти ToolBar та в ньому натиснути або Register для реєстрації або Login для входу до системи (рис 2.49):

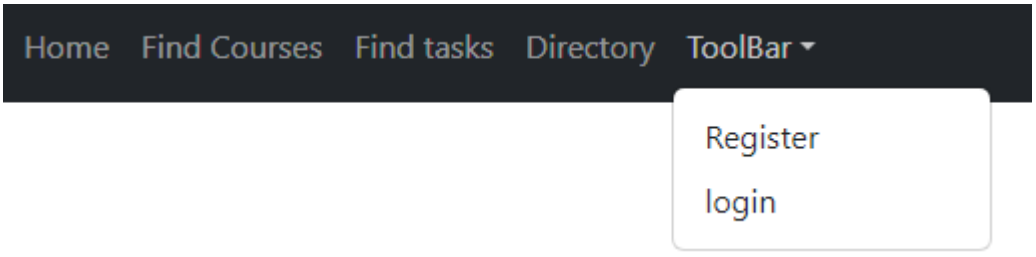


Рис.2.49. Перелік можливостей користувача в ToolBar

Вигляд заповненої форми для входу до системи (рис 2.50):

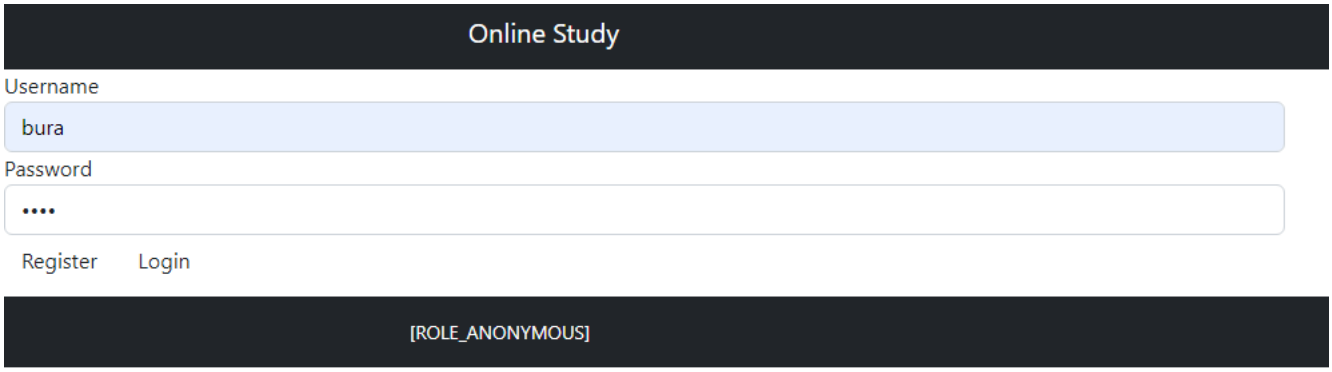


Рис.2.50. Заповнена форма для входу до системи

Після натискання кнопки Login якщо дані об користувачеві були введені вірно, то він потрапить на головну сторінку ОП та побачить що він успішно ввійшов до ОП.

Після чого користувачеві відкриваються усі можливості перебігу по сайту:

– можливість перегляду усіх курсів (рис 2.51), їх детальної сторінки (рис 2.52), реєстрації на курси (рис.2.53), та перегляду завдань які відносяться до курсів (рис 2.54);

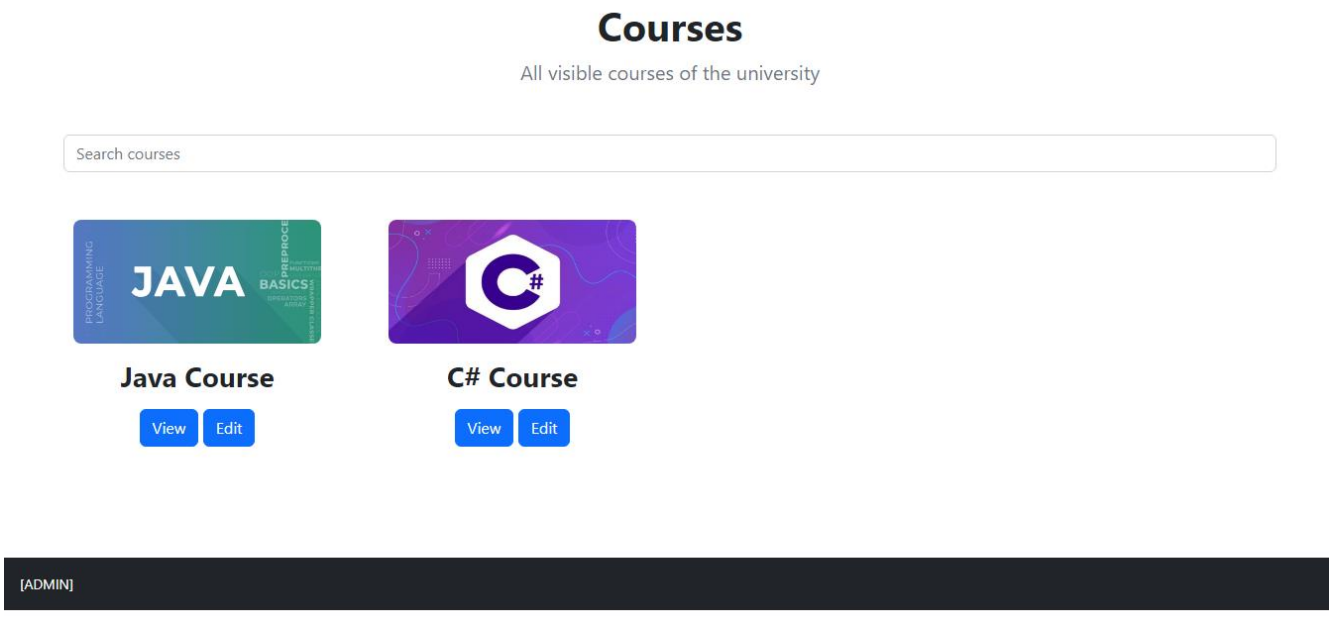


Рис.2.51. Головна сторінка з переліком всіх курсів

Рис.2.52. Сторінка детальної інформації курсу

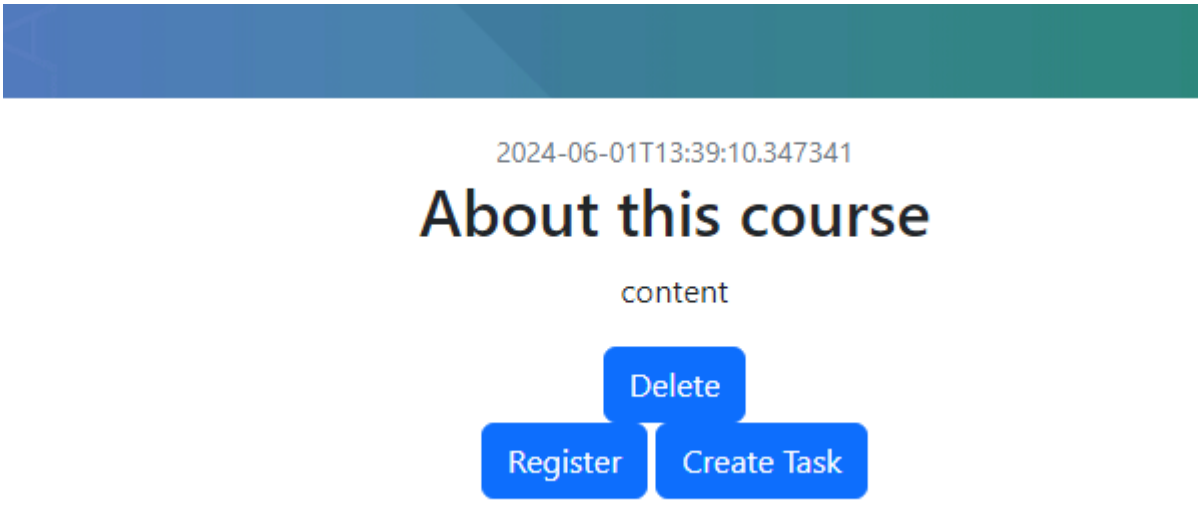


Рис.2.53. Кнопки реєстрації на курс, видалення курсу, та додавання нового завдання

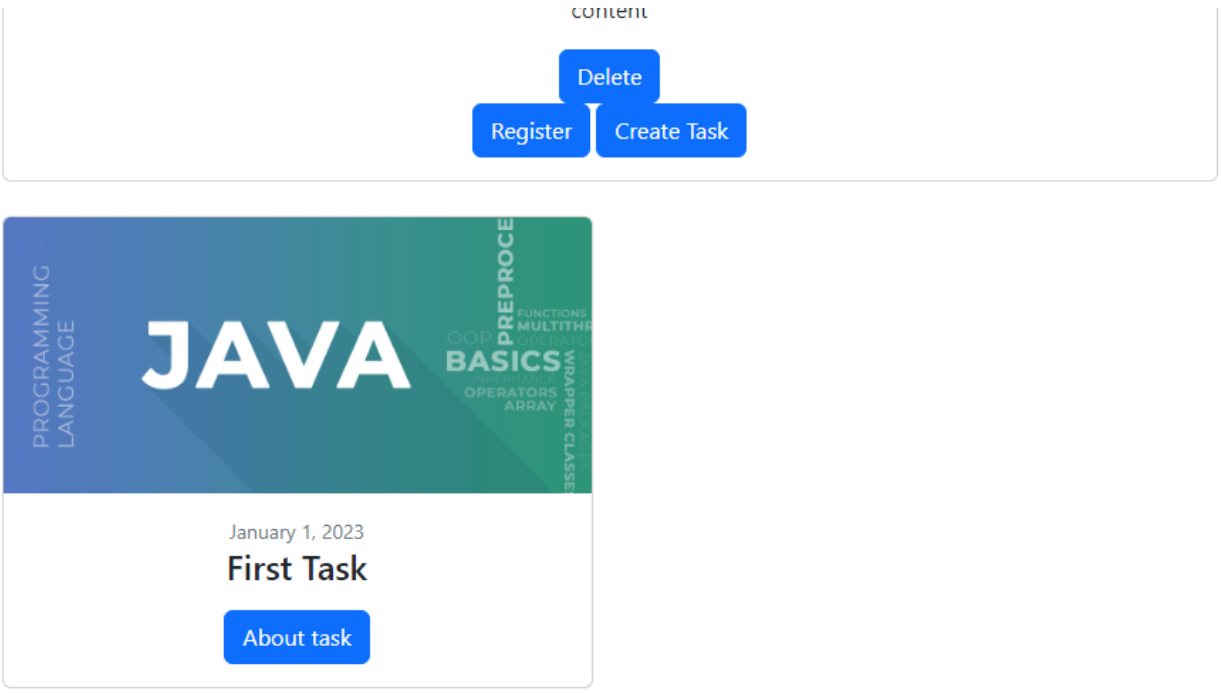


Рис.2.54. Перегляд завдань які відносяться до курсу

– можливість перегляду сторінки з завданнями (рис 2.55), їх детальної сторінки (рис 2.56), та можливість функціонувати з ними (рис 2.57);

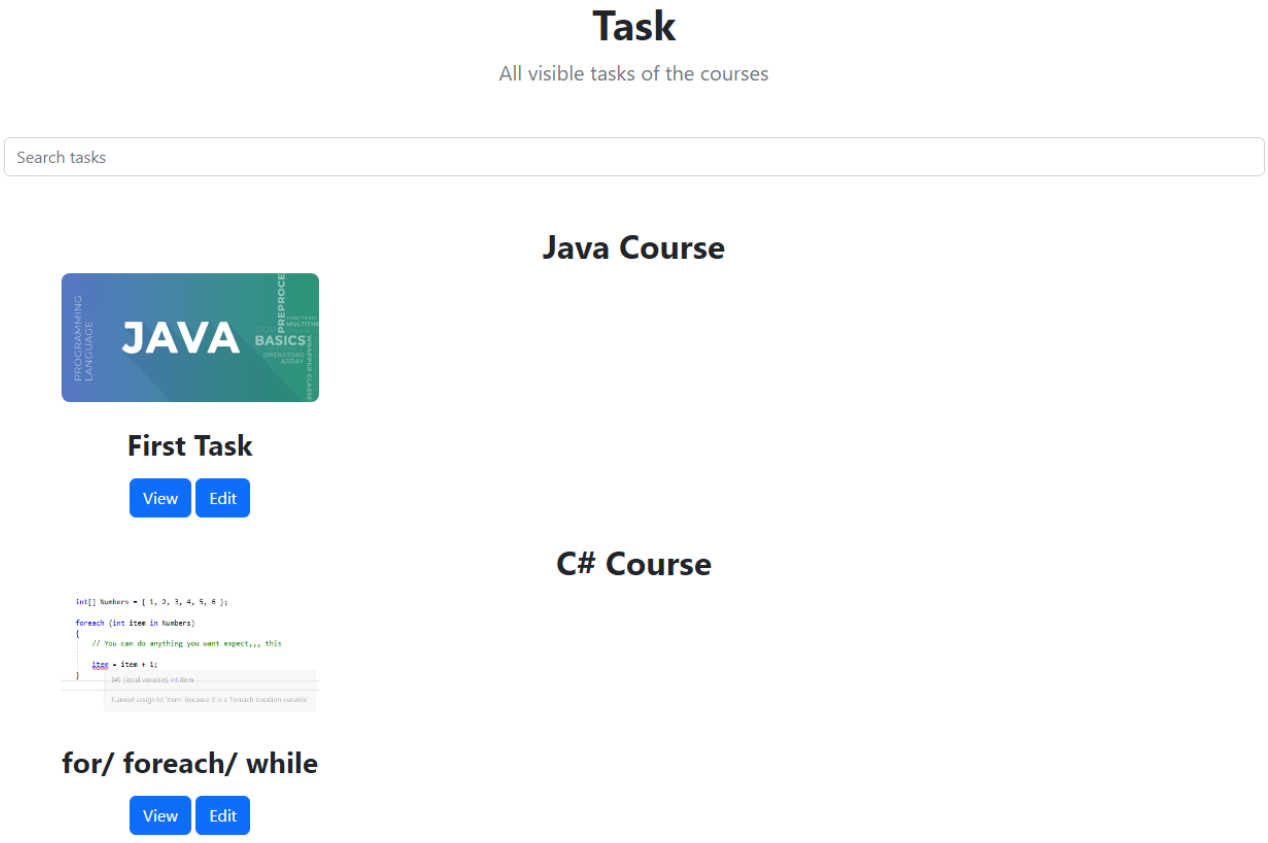


Рис.2.55. Сторінка з усіма завданнями ОП

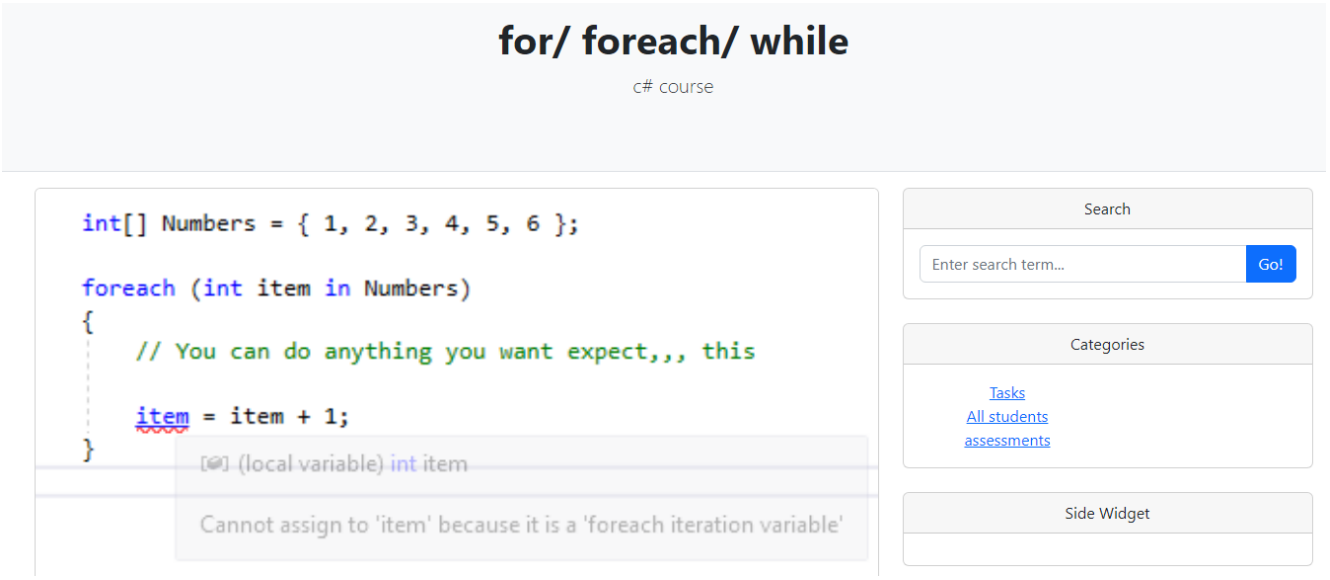


Рис.2.56. Детальна сторінка завдання

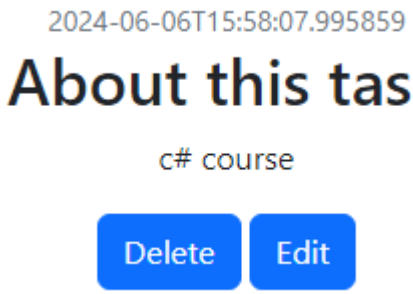


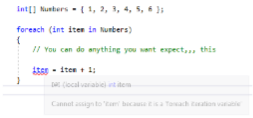
Рис.2.57. Можливість видалення та редагування завдання

– детальна сторінка особистого кабінету користувача(рис 2.58), та його функції;

Username	bura
Email	bura
Role	ADMIN
Courses	• C# Course
Assessments	• C# Course
Log of created courses	• C# Course

user tasks

C# Course



for/

foreach/ while

View Edit

Рис.2.58. Сторінка особистого кабінету користувача

– та форми для редагування (рис 2.59 - 2.60), видалення (зазначені на рис 2.53 та рис 2.57) курсів та завдань, та додавання нових прикладів (рис 2.61 - 2.62) тільки для викладачів та адміністрації сайту;

Title

Java Course

Photo Url

https://media.geeksforgeeks.org/wp-content/cdn-uploads/20230305131111/Java-progr

Content

content

Edit

Рис.2.59. Сторінка форми для редагування курсу

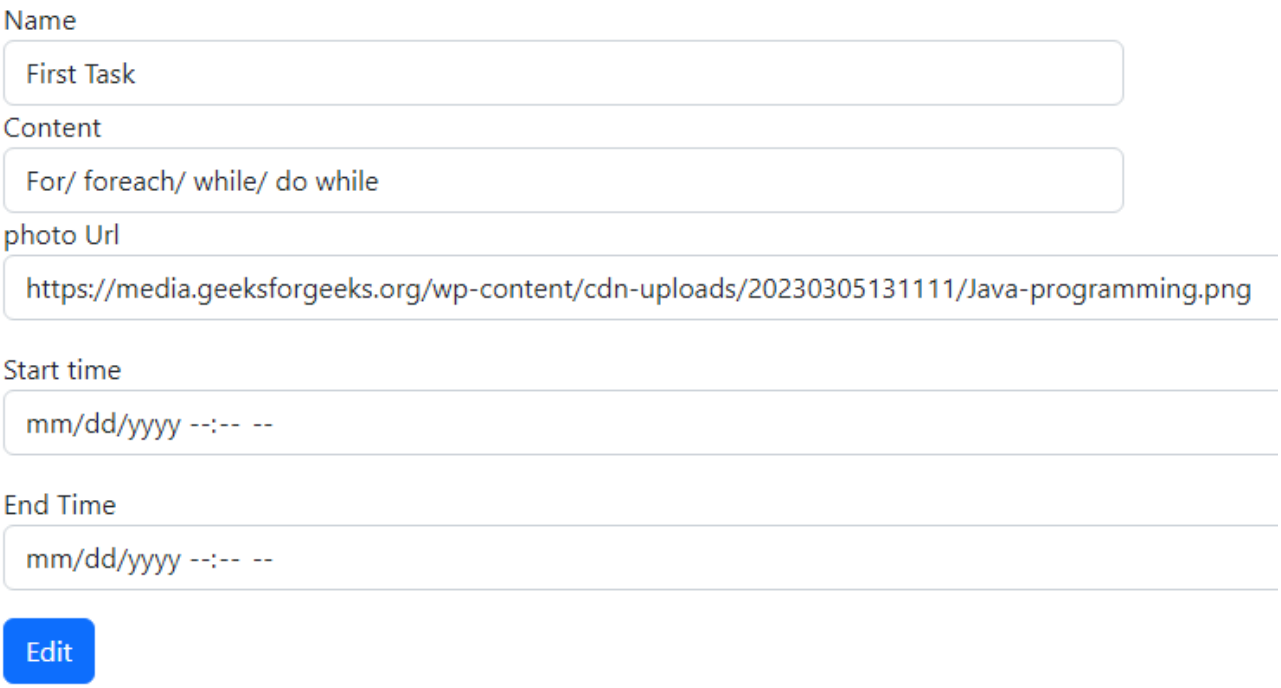


Рис.2.60. Сторінка форми для редагування завдання

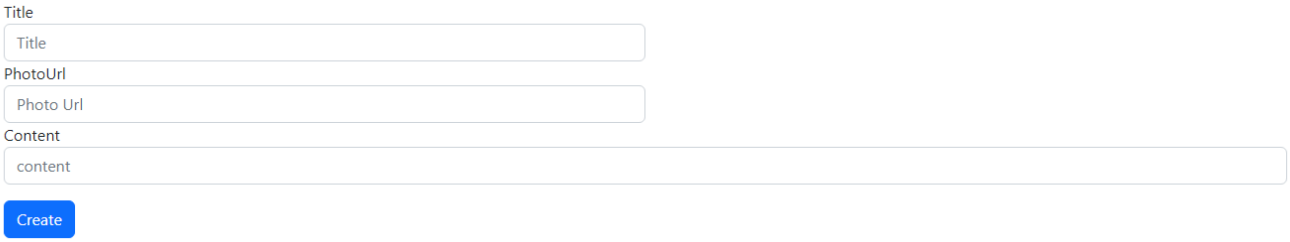


Рис.2.61. Сторінка форми для додавання нового курсу

Рис.2.62. Сторінка форми для додавання нового завдання

– перегляд оцінок на зареєстрованих курсів за допомогою UserDetails → Assessments → “Будь який курс” (рис 2.63).

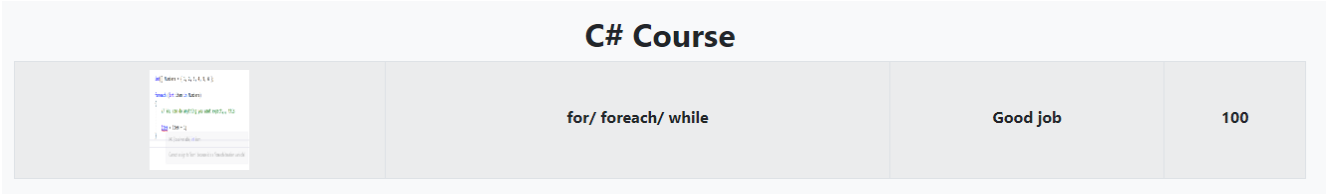


Рис.2.63. Сторінка з оцінками за завдання курсу C# Course

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Визначення трудомісткості розробки програмного забезпечення

Вхідні дані:

– передбачуване число операторів програми – 2950;

– коефіцієнт складності програми – 1.4;

– коефіцієнт корекції програми в ході її розробки – 0.1;

– годинна заробітна плата програміста – 137.06 грн/год.

Середня годинна заробітна плата програміста була отримана за допомогою сайту вакансій в IT спільноті. На 2024 рік, стандартна заробітна плата програміста становить 600 доларів на місяць (24123 грн), та після цього було узято 176 годин на місяць (22 днів по 8 часів). Формула наведена нижче (3.1):

$$\frac{A}{B} = C \quad (3.1)$$

де:

A - це заробітна плата програміста на місяць,

B - кількість годин на місяць,

C - годинна заробітна плата.

– коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1.3;

– коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1.2;

– вартість машино-годин ЕОМ – 49.58 (розрахунок наведено нижче).

Вартість години роботи комп’ютера буде дорівнювати:

вартість роботи комп’ютера на годину * ціну за 1 кВт/год.

Вартість 1 кВт/год енергії на 1 червня 2024 року становиться 4.32 грн/год, та споживання комп’ютера на одну годину буде становити 250/год. Тобто вартість години роботи комп’ютера буде становити 0.25*4.32 = 1.08 грн/год.

Вартість ЕОМ, машино-годин на годину буде визначатися за формулою:

$$E+A+r \quad (3.2)$$

де:

E - вартість роботи комп'ютера на годину,

A - амортизація комп'ютерного обладнання (детальніше в формулі 3.3),

r - ремонт та обслуговування ЕОМ (детальніше в формулі 3.4).

Амортизація комп'ютерного обладнання вираховується за формулою:

$$\frac{C-L}{i} * 100\% \quad (3.3)$$

де:

C - вартість обладнання на якому реалізована навчальна ОП (25600 грн),

L - вартість ліквідації сміття – 134.2 грн

I - термін, за який було виконано кваліфікаційну роботу (3 місяці по 176 годин (528)).

Отже амортизація комп'ютерного обладнання дорівнює 48.23 грн.

Ремонт та обслуговування ЕОМ буде розраховуватись за формулою:

$$\frac{A}{i} \quad (3.4)$$

де:

A- амортизація (48.23 * 3),

i - термін (528).

Отже ремонт та обслуговування ЕОМ буде дорівнювати приблизно 0.27 грн/год.

Та фінальний розрахунок вартості машино-години ЕОМ:

$$1.08 + 48.23 + 0.27 = 49.58 \text{ грн/год.}$$

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці розробника. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ визначається за формулою:

$$t = t_0 + t_u + t_a + t_n + t_{отл} + t_д \text{ , людино-годин ,} \quad (3.5)$$

де:

t_0 - витрати праці на підготовку й опис поставленої задачі (приймається 50 людино-годин),

t_u - витрати праці на дослідження алгоритму рішення задачі,

t_a - витрати праці на розробку блок-схеми алгоритму,

t_n - витрати праці на програмування по готовій блок-схемі,

$t_{отл}$ - витрати праці на налагодження програми та ЕОМ,

$t_д$ - витрати праці на підготовку документації.

Для подальшого розрахунку потрібно визначати умовне число операторів програми, яке визначається за формулою:

$$Q = q * C * (1 + p) \tag{3.6}$$

де:

q - передбачуване число операторів (2950),

C - коефіцієнт складності програми (1.4),

p - коефіцієнт корекції програми (0.1).

Отже умовне число операторів буде дорівнювати:

$$Q = 2950 * 1.4 * (1 + 0.1) = 4131.1$$

Витрати праці на вивчення алгоритму рішення задачі (t_u) визначається за формулою:

$$t_u = \frac{Q*B}{(75..85)*k} \tag{3.7}$$

де:

B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі (1.3),

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності (1.2).

Отже витрати праці на вивчення алгоритму рішення задачі будуть дорівнювати:

$$t_u = \frac{4131.1 * 1.3}{80 * 1.2} = 55.94, \text{людино години}$$

Витрати праці на розробку алгоритму рішення задачі (t_a):

$$t_a = \frac{Q}{(20..25)*k} \quad (3.8)$$

Отже витрати праці на розробку алгоритму рішення задачі будуть дорівнювати:

$$t_a = \frac{4131.1}{20 * 1.2} = 172.12\text{ , людино} - \text{годин}$$

Витрати праці на складання програми по готовій блок-схемі (t_n):

$$t_n = \frac{Q}{(20..25)*k} \quad (3.9)$$

Отже витрати праці на складання програми по готовій блок-схемі будуть дорівнювати:

$$t_n = \frac{4131.1}{20 * 1.2} = 172.12\text{ , людино} - \text{годин}$$

Витрати праці на налагодження ЕОМ ($t_{\text{отл}}$ та $t_{\text{отл}}^k$):

– за умови автономного налагодження одного завдання:

$$t_{\text{отл}} = \frac{Q}{(4..5)*k} \quad (3.10)$$

та буде дорівнювати:

$$t_{\text{отл}} = \frac{4131.1}{4 * 1.2} = 2295.05\text{ , людино} - \text{годин}$$

– за умови автономного налагодження одного завдання:

$$t_{\text{отл}}^k = 1.5 * t_{\text{отл}} \quad (3.11)$$

та буде дорівнювати:

$$t_{\text{отл}}^k = 1.5 * 2295.05 = 3442.58\text{ , людино} - \text{годин}$$

Витрати праці на підготовку документації ($t_{\text{д}}$):

$$t_{\text{д}} = t_{\text{др}} + t_{\text{до}} \quad (3.12)$$

та буде дорівнювати:

$$t_{\text{д}} = 229.5 + 172.125 = 401.625\text{ , людино} - \text{годин}$$

де:

$t_{\text{др}}$ – трудомісткість підготовки матеріалів і рукопису.

$$t_{\text{др}} = \frac{Q}{(15..20)*k} \quad (3.13)$$

та буде дорівнювати:

$$t_{\text{др}} = \frac{4131.1}{15 * 1.2} = 229.5\text{ , людино} - \text{годин}$$

$t_{\text{до}}$ – трудомісткість редагування, печатки й оформлення документації

$$t_{\text{до}} = 0.75 * t_{\text{др}} \quad (3.14)$$

та буде дорівнювати:

$$t_{\text{до}} = 0.75 * 229.5 = 172.125\text{ , людино} - \text{годин}$$

Підготувавши всі показники які були надані вище, то все готово для розрахування трудомісткості розробки програмного забезпечення за формулою (3.5):

$$t = 50 + 55.94 + 172.12 + 172.12 + 2295.05 + 401.625 = 3146.855,$$

людино-годин.

3.2. Розрахунок витрат на створення програмного забезпечення

Для розрахунку витрат на створення ПЗ ($К_{\text{по}}$) потрібно використати заробітну плату програміста ($З_{\text{зп}}$) та витрати машинного часу:

$$К_{\text{по}} = З_{\text{зп}} + З_{\text{мв}} \quad (3.15)$$

А заробітна плата програміста:

$$З_{\text{зп}} = t * С_{\text{пр}} \quad (3.16)$$

де:

t – трудомісткість розробки програмного забезпечення,

$С_{\text{пр}}$ – середня заробітна плата програміста.

Отже буде дорівнювати:

$$З_{\text{зп}} = 3146.855 * 137.05 = 431276, \text{грн}$$

Вартість машинного часу, необхідного для налагодження програми:

$$З_{\text{мв}} = t_{\text{отл}} * С_{\text{мч}} \quad (3.16)$$

де:

$t_{\text{отл}}$ – трудомісткість налагодження програми на ЕОМ,

$С_{\text{мч}}$ – вартість машино-години ЕОМ (49.58).

Та буде дорівнювати:

$$З_{мв} = 2295.05 * 49.58 = 113788, грн$$

Отже ми можемо порахувати $К_{по}$:

$$К_{по} = 431276 + 113788 = 545064, грн$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k * F_p} \quad (3.16)$$

де:

B_k – число виконавців(1),

F_p – місячний фонд робочого часу (176).

Та буде дорівнювати:

$$T = \frac{3146.855}{1 * 176} = 17.87(міс)$$

ВИСНОВКИ

В рамках цієї кваліфікаційної роботи було розглянуто та аналізовано процес розробки навчальної ОП для організації навчального процесу з використанням мови програмування Java та її реалізацію.

Метою було поставлено розробити веб-застосунок навчальної ОП для організації навчального процесу.

Призначення цієї ОП полягає в тому, що студенти, викладачі та інший персонал зміг організовувати весь потенціал навчального процесу не виходячи з домівок та зміг зручно та ефективно контактувати одні з одними. У цьому випадку було розроблено веб-застосунок, на якому користувачі можуть проглядати курси, виконувати завдання, переглядати оцінки та тощо.

В процесі було пророблено такі етапи:

– було розглянуто більш масштабні проблеми такі як логістика в обміну в інформації, зручність в навігації по сайту, зручність в додаванні нового матеріалу та зручність в його застосуванні;

– було визначено основні вимоги, які повинні бути враховані при розробці ОП. Ці вимоги будуть складатися з потрібних характеристик ПК, потрібних програм, встановлених на комп’ютер фреймворків для їх роботи та інше;

– розроблено навчальну ОП для організації навчального процесу та описати її алгоритми роботи та структуру баз даних;

– було виконано економічну частину цієї роботи для розуміння витрат на розробку програмного продукту.

Результатом виконання цієї кваліфікаційної роботи є онлайн платформа для організації навчального процесу, яка має усі необхідні для цього функції.

Також в ході цієї кваліфікаційної роботи було прораховано трудомісткість розробленої системи, було проведено розрахунок вартості роботи (545064 грн) та було розраховано необхідний час на виконання цього проекту (17.87 міс).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Java tutorials/Spring boot/Hibernate: https://proselyte.net/tutorials/hibernate-tutorial/mapping-types/
2. What is Java : https://www.java.com/en/download/help/whatis_java.html
3. Guides for Spring boot : https://www.baeldung.com/
4. Thymeleaf guide: https://www.thymeleaf.org/doc/tutorials/2.1/usingthymeleaf.html
5. Maven repository: https://mvnrepository.com/
6. Maven: https://maven.apache.org/
7. Hibernate: https://hibernate.org/
8. Learn hibernate: https://www.geeksforgeeks.org/hibernate-tutorial/
9. What is rest api: https://www.ibm.com/topics/rest-apis#:~:text=A%20REST%20API%20(also%20called,transfer%20(REST)%20architectural%20style.
10. Детальніше про Rest api: https://foxminded.ua/shcho-take-rest-api/
11. HTML tutorials: https://www.w3schools.com/html/
12. CSS tutorials: https://www.w3schools.com/css/
13. Spring MVC: https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/mvc.html
14. Spring Security: https://spring.io/projects/spring-security
15. Spring data jpa: https://spring.io/projects/spring-data-jpa
16. Intellij IDEA: https://www.jetbrains.com/idea/
17. PostgreSQL: https://www.postgresql.org/

ДОДАТОК А

КОД ПРОГРАМИ

SPRING BOOT

```
package com.example.web.config;

import com.example.web.service.impl.MyUserDetailsService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.boot.web.servlet.FilterRegistrationBean;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBu
ilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityCustomizer;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.SecurityFilterChain;
import
org.springframework.security.web.server.authentication.logout.DelegatingServerLogoutHandler;
import
org.springframework.security.web.server.authentication.logout.SecurityContextServerLogoutHan
dler;
import
org.springframework.security.web.server.authentication.logout.WebSessionServerLogoutHandler;
import
org.springframework.security.web.servletapi.SecurityContextHolderAwareRequestFilter;
import org.springframework.security.web.util.matcher.AntPathRequestMatcher;
```

```
import java.util.logging.Filter;

import static org.springframework.security.config.Customizer.withDefaults;

@Configuration
@EnableWebSecurity
public class SecurityConfig {
   private MyUserDetailsService myUserDetailsService;

   @Autowired
   public SecurityConfig(MyUserDetailsService myUserDetailsService) {
      this.myUserDetailsService = myUserDetailsService;
   }

   //    @Bean
//    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
//       http.authorizeRequests()
//             .requestMatchers ("/register", "/css/**", "/js/**")
//             .permitAll()
//             .anyRequest()
//             .authenticated()
//             .and()
//             .formLogin(form -> form
//                   .loginPage("/login")
//                   .defaultSuccessUrl("/courses")
//                   .loginProcessingUrl("/login")
//                   .failureForwardUrl("/login?error=true")
//                   .permitAll())
//             .logout(logout -> logout
//                   .logoutRequestMatcher(new AntPathRequestMatcher("/logout"))
//                   .logoutSuccessUrl("/login?logout").permitAll());
//       return http.build();
//    }
   @Bean
```

```
    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        http.csrf().disable()
                .authorizeRequests()
                .requestMatchers("/register", "register/save", "/courses" ,"/login", "/css/**",
"/js/**")
                .permitAll()
                .anyRequest().authenticated()
                .and()
                .formLogin(form -> form
                        .loginPage("/register")
                        .defaultSuccessUrl("/courses")
                        .loginPage("/login")
                        .defaultSuccessUrl("/courses")
                        .loginProcessingUrl("/login")
                        .failureUrl("/login?error=true")
                        .permitAll()
                ).logout(
                        logout -> logout
                                .logoutRequestMatcher(new AntPathRequestMatcher("/logout"))
                                .logoutSuccessUrl("/login?logout")
                                .permitAll());

        return http.build();
    }

    @Bean
    public static PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }

    public void configure(AuthenticationManagerBuilder builder) throws Exception {

builder.userDetailsService(myUserDetailsService).passwordEncoder(passwordEncoder());
    }
```

```
}

package com.example.web.config;

import org.springframework.security.authentication.AnonymousAuthenticationToken;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;

public class SecurityUtil {
   public static String getSessionUser(){
      Authentication authentication =
SecurityContextHolder.getContext().getAuthentication();
      if(!(authentication instanceof AnonymousAuthenticationToken)){
         String currentUsername = authentication.getName();
         return currentUsername;
      }
      return null;
   }
}
package com.example.web.controller;

import com.example.web.dto.RegistrationDto;
import com.example.web.models.UserEntity;
import com.example.web.service.UserService;
import com.example.web.service.impl.MyUserDetailsService;
import jakarta.validation.Valid;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;
```

```
@Controller
public class AuthController {
    private UserService userService;
    private MyUserDetailsService myUserDetailsService;

    @Autowired
    public AuthController(UserService userService, MyUserDetailsService
myUserDetailsService) {
        this.userService = userService;
        this.myUserDetailsService = myUserDetailsService;
    }

    @GetMapping("/login")
    public String loginPage() {
        return "login";
    }

    @PostMapping("/login")
    public String loginSubmit(@RequestParam String username, @RequestParam String
password, Model model) {
        UserDetails userDetails = myUserDetailsService.loadUserByUsername(username);

        if (userDetails == null) {
            return "redirect:/error";
        }

        if ("validUsername".equals(username) && "validPassword".equals(password)) {
            model.addAttribute("user", userDetails);
            return "redirect:/courses";
        }
        model.addAttribute("user", userDetails);
        return "redirect:/courses";
    }
```

```
    @GetMapping("/logout")
    public String logoutPage() throws Exception {
        return "redirect:/";
    }

    @GetMapping("/register")
    public String getRegisterForm(Model model) {
        RegistrationDto user = new RegistrationDto();
        model.addAttribute("user", user);
        return "register";
    }

    @PostMapping("/register/save")
    public String register(@Valid @ModelAttribute("user") RegistrationDto user,
                           BindingResult result, Model model) {
        UserEntity existingUserEmail = userService.findByEmail(user.getEmail());
        if (existingUserEmail != null && existingUserEmail.getEmail() != null &&
!existingUserEmail.getEmail().isEmpty()) {
            return "redirect:/register?fail";
        }
        UserEntity existingUserUsername =
userService.findByUsername(user.getUsername());
        if (existingUserUsername != null && existingUserUsername.getUsername() != null
&& !existingUserUsername.getUsername().isEmpty()) {
            return "redirect:/register?fail";
        }
        if (result.hasErrors()) {
            model.addAttribute("user", user);
            return "register";
        }
        userService.saveUser(user);
        return "redirect:/login?success";
    }
}
```

```
package com.example.web.controller;

import com.example.web.config.SecurityUtil;
import com.example.web.dto.CourseDto;
import com.example.web.models.Course;
import com.example.web.models.UserEntity;
import com.example.web.service.CourseService;
import com.example.web.service.UserService;
import jakarta.validation.Valid;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.Banner;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.annotation.AuthenticationPrincipal;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.User;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.zip.DataFormatException;

@Controller
public class CourseController {
    private CourseService courseService;
    private UserService userService;


    @Autowired
    public CourseController(CourseService courseService, UserService userService) {
        this.courseService = courseService;
        this.userService = userService;
    }
```

```
    @GetMapping("/courses/{courseId}/allRegisteredUsers")
    public String allRegisteredUsers(@PathVariable(value = "courseId")Long courseId,
                                     Model model){
        Authentication authentication =
SecurityContextHolder.getContext().getAuthentication();
        String username = authentication.getName();
        UserEntity userEntity = userService.findByUsername(username);
        boolean isRegistered = userService.isAlreadyUserOnCourse(userEntity.getId(),
courseId);
        CourseDto courseDto = courseService.findCourseById(courseId);
        List<UserEntity> users = userService.findUsersByCourse(courseId);
        model.addAttribute("isRegistered", isRegistered);
        model.addAttribute("course", courseDto);
        model.addAttribute("users", users);
        return "all-users-course";
    }

    @GetMapping("/courses/{courseId}/{userId}/registrationOnCourse")
    public String userRegistration(@PathVariable("courseId") Long courseId,
                                   @PathVariable("userId")Long userId,
                                   Model model){
        String username = userService.findUserById(userId).getUsername();
        UserEntity userEntity = userService.findByUsername(username);
        CourseDto courseDto = courseService.findCourseById(courseId);
        model.addAttribute("user", userEntity);
        model.addAttribute("course", courseDto);
        return "register-on-course";
    }
    @PostMapping("/courses/{courseId}/{userId}/save")
    public String registerUserOnCourse(@PathVariable("courseId")Long courseId,
                                       @PathVariable("userId")Long userId
                                       ) {
        if(userService.isAlreadyUserOnCourse(userId, courseId)){
            return "redirect:/error";
        }
```

```
        userService.updateUser(userId ,courseId);
        return "success";
    }


    @GetMapping("/courses")
    public String listCourses(Model model){
        UserEntity userEntity = new UserEntity();
        List<CourseDto> courses = courseService.findAllCourses();
        String username = SecurityUtil.getSessionUser();
        if(username != null){
            userEntity = userService.findByUsername(username);
            model.addAttribute("user", userEntity);
        }
        model.addAttribute("user", userEntity);
        model.addAttribute("courses", courses);
        return "courses-list";
    }

    @GetMapping("/courses/{courseId}")
    public String courseDetail(@PathVariable("courseId") Long courseId,
                               Model model){
        Authentication authentication =
SecurityContextHolder.getContext().getAuthentication();
        String username = authentication.getName();
        UserEntity userEntity = userService.findByUsername(username);
        CourseDto courseDto = courseService.findCourseById(courseId);
        boolean isRegistered = courseService.isRegisteredOnCourse(userEntity.getId(),
courseId);
        model.addAttribute("isRegistered", isRegistered);
        model.addAttribute("user", userEntity);
        model.addAttribute("course", courseDto);
        return "courses-detail";
    }
```

```
@GetMapping("/courses/new")
public String createCourseForm(Model model){
   Course course = new Course();
   model.addAttribute("course", course);
   return "course-create";
}
@PostMapping("/courses/new")
public String saveCourse(@Valid @ModelAttribute("course")CourseDto courseDto,
                  BindingResult result, Model model){
   if(result.hasErrors()){
      model.addAttribute("course", courseDto);
      return "course-create";
   }
   courseService.saveCourse(courseDto);
   return "redirect:/courses";
}

@PostMapping("/courses/{courseId}/delete")
public String deleteCourse(@PathVariable("courseId") Long courseId){
   courseService.delete(courseId);
   return "redirect:/courses";
}

@GetMapping("/courses/search")
public String searchCourse(@RequestParam(value = "query")String query,
                     Model model){
   List<CourseDto> courses = courseService.searchCourses(query);
   model.addAttribute("courses", courses);
   return "courses-list";
}


@GetMapping("/courses/{courseId}/edit")
public String editCourseForm(@PathVariable("courseId") Long courseId, Model
model){
```

```
        CourseDto courseDto = courseService.findCourseById(courseId);
        model.addAttribute("course", courseDto);
        return "courses-edit";
    }
    @PostMapping("/courses/{courseId}/edit")
    public String updateCourse(@PathVariable("courseId") Long courseId,
                               @Valid @ModelAttribute("course") CourseDto courseDto,
                               BindingResult result, Model model){
        if(result.hasErrors()){
            model.addAttribute("course", courseDto);
            return "courses-edit";
        }
        courseDto.setId(courseId);
        courseService.updateCourse(courseDto);
        return "redirect:/courses";
    }
}
package com.example.web.controller;

import jakarta.servlet.RequestDispatcher;
import jakarta.servlet.http.HttpServletRequest;
import org.springframework.boot.web.servlet.error.ErrorController;
import org.springframework.http.HttpStatus;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class MyErrorController implements ErrorController {

    @GetMapping("/error")
    public String handleError(HttpServletRequest request) {
        Object status = request.getAttribute(RequestDispatcher.ERROR_STATUS_CODE);

        if (status != null) {
```

```
            Integer statusCode = Integer.valueOf(status.toString());

            if(statusCode == HttpStatus.NOT_FOUND.value()) {
                return "error-404";
            }
            else if(statusCode == HttpStatus.INTERNAL_SERVER_ERROR.value()) {
                return "error-500";
            }
        }
        return "error";
    }
}
```

```
package com.example.web.controller;

import com.example.web.dto.CourseDto;
import com.example.web.dto.TaskDto;
import com.example.web.models.Course;
import com.example.web.models.Task;
import com.example.web.service.CourseService;
import com.example.web.service.TaskService;
import com.example.web.service.impl.Mappers;
import jakarta.validation.Valid;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.Banner;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@Controller
public class TaskController {
    private TaskService taskService;
```

```
private CourseService courseService;

@Autowired
public TaskController(TaskService taskService, CourseService courseService) {
   this.taskService = taskService;
   this.courseService = courseService;
}



@GetMapping("/tasks")
public String taskList(Model model){
   List<CourseDto> courses = courseService.findAllCourses();
   List<TaskDto> tasks = taskService.findAllTasks();
   model.addAttribute("tasks", tasks);
   model.addAttribute("courses", courses);
   return "tasks-list";
}

@GetMapping("/tasks/{taskId}/{courseId}")
public String viewTask(@PathVariable("taskId") Long taskId,
                       @PathVariable("courseId") Long courseId,
                       Model model){
   TaskDto taskDto = taskService.findTaskById(taskId);
   model.addAttribute("courseId", courseId);
   model.addAttribute("task", taskDto);
   return "tasks-detail";
}
@GetMapping("/tasks/{taskId}/delete")
public String deleteTask(@PathVariable("taskId") Long taskId){
   taskService.delete(taskId);
   return "redirect:/tasks";
}
@GetMapping("/tasks/search")
public String searchTasks(@RequestParam(value = "query")String query,
                          Model model){
```

```
    List<CourseDto> courses = courseService.findAllCourses();
    List<TaskDto> tasks = taskService.searchTasks(query);
    model.addAttribute("courses", courses);
    model.addAttribute("tasks", tasks);
    return "tasks-list";
}

@GetMapping("/tasks/{courseId}/new")
public String createTaskForm(@PathVariable("courseId") Long courseId,
                             Model model){
    Task task = new Task();
    model.addAttribute("courseId", courseId);
    model.addAttribute("task", task);
    return "tasks-create";
}
@PostMapping("/tasks/{courseId}")
public String createTask(@PathVariable("courseId") Long courseId,
                         @ModelAttribute("task")TaskDto taskDto,
                         BindingResult result,Model model){
    if(result.hasErrors()){
        model.addAttribute("task", taskDto);
        return "tasks-create";
    }
    taskService.createTask(courseId,taskDto);
    return "redirect:/courses";
}
@GetMapping("/tasks/{taskId}/{courseId}/edit")
public String editTaskForm(@PathVariable("taskId") Long taskId,
                           @PathVariable("courseId") Long courseId,
                           Model model){
    TaskDto taskDto = taskService.findTaskById(taskId);
    model.addAttribute("courseId",courseId);
    model.addAttribute("task", taskDto);
    return "tasks-edit";
}
```

```
    @PostMapping("/tasks/{taskId}/{courseId}/edit")
    public String updateTask(@PathVariable("taskId") Long taskId,
                             @PathVariable("courseId") Long courseId,
                             @Valid @ModelAttribute("task") TaskDto taskDto,
                             BindingResult result, Model model){
        if(result.hasErrors()){
            model.addAttribute("task", taskDto);
            return "tasks-edit";
        }
        if (courseId == null) {
            model.addAttribute("errorMessage", "Course ID is required.");
            model.addAttribute("task", taskDto);
            return "tasks-edit";
        }

        taskDto.setId(taskId);

        Course course = Mappers.mapToCourse(courseService.findCourseById(courseId));
        taskDto.setCourse(course);

        taskService.updateTask(taskDto);

        return "redirect:/tasks";
    }


}

package com.example.web.controller;

import com.example.web.dto.AssessmentDto;
import com.example.web.dto.CourseDto;
import com.example.web.dto.TaskDto;
import com.example.web.models.Assessment;
import com.example.web.models.UserEntity;
```

```
import com.example.web.service.AssessmentService;
import com.example.web.service.CourseService;
import com.example.web.service.TaskService;
import com.example.web.service.UserService;
import com.example.web.service.impl.Mappers;
import jakarta.validation.Valid;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.parameters.P;
import org.springframework.security.core.userdetails.User;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;

import java.util.List;

@Controller
public class UserDetailController {

    private UserService userService;
    private CourseService courseService;
    private TaskService taskService;
    private AssessmentService assessmentService;

    @Autowired
    public UserDetailController(UserService userService, CourseService courseService,
TaskService taskService,AssessmentService assessmentService) {
        this.userService = userService;
        this.courseService = courseService;
        this.taskService = taskService;
```

```
        this.assessmentService = assessmentService;
    }

    @GetMapping("/addAssessment/{userId}/{taskId}/delete/{assessmentId}")
    public String deleteUserAssessment(@PathVariable(value = "userId") Long userId,
                                       @PathVariable(value = "taskId")Long taskId,
                                       @PathVariable(value = "assessmentId")Long assessmentId,
                                       Model model){
        assessmentService.deleteAssessment(assessmentId);
        return "redirect:/userDetails";
    }
    @GetMapping("/addAssessment/{userId}/{taskId}/edit/{assessmentId}")
    public String editUserAssessment(@PathVariable(value = "userId") Long userId,
                                     @PathVariable(value = "taskId")Long taskId,
                                     @PathVariable(value = "assessmentId")Long assessmentId,
                                     Model model){
        AssessmentDto assessmentDto =
assessmentService.findAssessmentById(assessmentId);
        model.addAttribute("assessment", assessmentDto);
        return "assessment-edit";
    }
    @PostMapping("/addAssessment/{userId}/{taskId}/edit/{assessmentId}/save")
    public String saveEditedUserAssessment(@PathVariable(value = "userId") Long userId,
                                           @PathVariable(value = "taskId")Long taskId,
                                           @PathVariable(value = "assessmentId")Long assessmentId,
                                           @Valid @ModelAttribute("assessment") AssessmentDto
assessmentDto,
                                           BindingResult result, Model model){
        if(result.hasErrors()){
            model.addAttribute("assessment", assessmentDto);
            return "assessment-edit";
        }
        AssessmentDto assessmentDto1 =
assessmentService.findAssessmentById(assessmentId);
        assessmentDto.setId(assessmentId);
```

```
        assessmentDto.setUser(userService.findUserById(userId));
        assessmentDto.setTask(Mappers.mapToTask(taskService.findTaskById(taskId)));
        assessmentService.updateAssessment(assessmentDto);
        return "redirect:/editLogOfCourse/" + assessmentDto.getTask().getCourse().getId();
    }

    @GetMapping("/addAssessment/{userId}/{taskId}/new")
    public String addUserAssessment(@PathVariable(value = "userId") Long userId,
                                    @PathVariable(value = "taskId")Long taskId,
                                    Model model){
        Assessment assessment = new Assessment();
        model.addAttribute("assessment", assessment);
        model.addAttribute("userId", userId);
        model.addAttribute("taskId", taskId);
        return "assessment-create";
    }
    @PostMapping("/addAssessment/{userId}/{taskId}")
    public String saveUserAssessment(@PathVariable(value = "userId") Long userId,
                                    @PathVariable(value = "taskId")Long taskId,
                                    @ModelAttribute("assessment") AssessmentDto assessmentDto,
                                    BindingResult result,
                                    Model model){
        if(result.hasErrors()){
            model.addAttribute("assessment", assessmentDto);
            return "assessment-create";
        }
        assessmentService.createAssessment(userId, taskId, assessmentDto);
        return "redirect:/userDetails";
    }

    @GetMapping("/userDetails")
    public String userDetails(Model model){
        Authentication authentication =
SecurityContextHolder.getContext().getAuthentication();
        String userName = authentication.getName();
```

```
        UserEntity userEntity = userService.findByUsername(userName);
        model.addAttribute("user", userEntity);
        return "user-detail";
    }
    @GetMapping("/assessments/{courseId}")
    public String assessmentsOfCourse(@PathVariable(value = "courseId") Long courseId,
                                      Model model){
        Authentication authentication =
SecurityContextHolder.getContext().getAuthentication();
        String username = authentication.getName();
        UserEntity userEntity = userService.findByUsername(username);
        CourseDto courseDto = courseService.findCourseById(courseId);
        List<TaskDto> tasks = taskService.findTaskByCourse(courseDto);
        List<AssessmentDto> assessments = assessmentService.findAllAssessments();
        model.addAttribute("user", userEntity);
        model.addAttribute("assessments", assessments);
        model.addAttribute("course", courseDto);
        model.addAttribute("tasks", tasks);
        return "assessments";
    }
    @GetMapping("/editLogOfCourse/{courseId}")
    public String editLogOfCoursePage(@PathVariable(value = "courseId")Long courseId,
                                      Model model){
        CourseDto courseDto = courseService.findCourseById(courseId);
        List<TaskDto> tasks = taskService.findTaskByCourse(courseDto);
        List<UserEntity> users = userService.findUsersByCourse(courseId);
        List<AssessmentDto> assessments = assessmentService.findAllAssessments();
        model.addAttribute("assessments", assessments);
        model.addAttribute("users", users);
        model.addAttribute("course", courseDto);
        model.addAttribute("tasks", tasks);
        return "log-of-course";
    }
}
```

```
package com.example.web.dto;

import com.example.web.models.Task;
import com.example.web.models.UserEntity;
import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;
import org.hibernate.annotations.CreationTimestamp;
import org.hibernate.annotations.UpdateTimestamp;

import java.time.LocalDateTime;

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class AssessmentDto {
    private Long id;
    private int assessment;
    private String comment;
    @CreationTimestamp
    private LocalDateTime createdOn;
    @UpdateTimestamp
    private LocalDateTime updatedOn;
    private Task task;
    private UserEntity user;
}
```

```
package com.example.web.dto;

import com.example.web.models.UserEntity;
import jakarta.validation.constraints.NotEmpty;
import lombok.AllArgsConstructor;
```

```
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.time.LocalDateTime;
import java.util.List;

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class CourseDto {

    private Long id;
    @NotEmpty(message = "Course title should not be empty")
    private String title;
    @NotEmpty(message = "Course link should not be empty")
    private String photoUrl;
    @NotEmpty(message = "Course content should not be empty")
    private String content;
    private LocalDateTime createOn;
    private LocalDateTime updatedOn;
    private UserEntity createdBy;
    private List<TaskDto> tasks;
    private List<UserEntity> users;

}

package com.example.web.dto;

import jakarta.validation.constraints.NotEmpty;
import lombok.Data;

import java.util.List;
```

```
@Data
public class RegistrationDto {
    private Long id;
    @NotEmpty
    private String username;
    @NotEmpty
    private String email;
    @NotEmpty
    private String password;
}

@SpringBootApplication
public class WebApplication {

public static void main(String[] args) {
        SpringApplication.run(WebApplication.class, args);
}

}
```

ДОДАТОК Б

ВІДГУК

Керівника економічного розділу

на кваліфікаційну роботу бакалавра на тему:

« навчальна онлайн-платформа для організації навчального процесу з використанням мови програмування Java»

Студента групи 122-20-2 Бурлаки Олександра Сергійовича

Керівник економічного розділу **доц. каф. ПЕП та ПУ, к.е.н**	**Л.В. Касьяненко**

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файла	Опис
Пояснювальні документи	
Диплом_Бурлака.doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Диплом_Бурлака.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Program.rar	Архів. Містить коди програми і програму
Презентація	
Презентація_ Бурлака.pptx	Презентація кваліфікаційної роботи