МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРНІВЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЮРІЯ ФЕДЬКОВИЧА

Навчально-науковий інститут фізико-технічних та комп'ютерних наук
кафедра комп'ютерних наук

Інформаційна система "Портал електронного навчання на java"

Курсова робота

Рівень вищої освіти – другий (магістерський)

Виконав:
студент 5 курсу, 544 групи
Лучик Павло Олегович
Керівник:
доктор фізико-математичних наук,
професор Ушенко Ю.О.

Чернівці – 2024

ЗАВДАННЯ
НА КУРСОВУ РОБОТУ СТУДЕНТА

Лучика Павла Олеговича

(*прізвище, ім'я, по батькові*)

1. Тема роботи

Інформаційна система "Портал електронного навчання на java"

затверджена засіданням кафедри від «___» _____ 2024 року, протокол №1

2. Термін подання студентом закінченої роботи

2024

3. Вхідні дані до роботи

Мета, предмет, аналіз

4. Зміст розрахунково-пояснювальної записки (перелік питань, які треба розробити)

Здійснити огляд аналогічних систем та програмного забезпечення, здійснити аналіз програмних засобів, огляд алгоритмів та методів реалізації окремих частин отриманого завдання, здійснити проектування бекенду, здійснити проектування фронтенду, виконати розробку додатку

5. Перелік графічного, наочного матеріалу

Рисунки

6. Консультант(и) курсової роботи

Ушенко Ю.О.

№ з/п	Назва етапів курсової роботи	Строк виконання етапів проекту	Примітка
1	Отримання завдання на курсову роботу		
2	Розробка технічного завдання		
3	Виконання аналізу методів реалізації завдання		
4	Розробка дизайну інтерфейсу додатку		
5	Розробка бекенду		
6	Розробка фронтенку		
7	Написання пояснювальної записки		
8	Подача роботи керівнику для написання відгуку		
9	Коригування роботи за результатами розгляду керівника. Остаточне оформлення пояснювальної записки.		
10	Написання доповіді створення слайдів		
11	Захист курсової роботи		

Студент Лучик П. О. / П.І.Б. / науковий керівник Ушенко Ю.О. / П.І.Б. /

(підпис) (підпис)

Завдання видане «__» **2024р.**

АНОТАЦІЯ

У рамках виконання курсової роботи були здійснені дослідницькі, аналітичні, проектувальні та розробницькі етапи створення інформаційної системи "Портал електронного навчання на java". Проект базується на принципах MVC (Model-View-Controller), що сприяє легкості управління структурою, підтримки та ефективної інтеграції нових можливостей. Було проведено аналіз схожих систем та методологій проектування та розробки, що включав порівняльний огляд та обґрунтування вибору використаних технологій.

Розроблений програмний продукт надає можливості для створення курсів, завантаження навчальних матеріалів, перегляду курсів та участі у дискусіях. Цей додаток дозволяє користувачам ефективно взаємодіяти з контентом і спільнотою, забезпечуючи багатий набір функцій, що задовольняє потреби сучасних користувачів. Всі ці етапи були детально реалізовані з метою створення зручного та функціонального інтерфейсу.

Сторінок – 58, рисунків – 24, джерел літератури – 17, додатків – 1.

Ключові слова: електронне навчання, java, eclipse, sql, курс.

Курсова робота містить результати власних досліджень. Використання ідей, результатів і текстів наукових досліджень інших авторів мають посилання на відповідне джерело.

_____ П.О.Лучик
(підпис)

АНОТАЦІЯ

As part of the course work, the research, analytical, design and development stages of the creation of the information system "Portal of e-learning in java" were carried out. The project is based on the principles of MVC (Model-View-Controller), which facilitates the ease of structure management, support and effective integration of new features. An analysis of similar systems and design and development methodologies was conducted, which included a comparative review and justification of the choice of technologies used.

The developed software product provides opportunities for creating courses, downloading educational materials, viewing courses and participating in discussions. This application allows users to effectively interact with content and community, providing a rich set of features that meet the needs of modern users. All these stages were implemented in detail in order to create a convenient and functional interface.

Pages – 58, figures – 22, literature sources – 17, appendices – 1.

The course work contains the results of own research. The use of ideas, results and texts of scientific research of other authors have a link to the corresponding source.

_____ P.O. Luchyk
(signature)

ЗМІСТ

ВСТУП

З появою інтернету звичні для нас речі почали своє «переміщення» у мережу. Тенденція не оминула й освітню галузь. Проект електронного ресурсу для дистанційного навчання створений з метою надати освітнім закладам, викладачам та студентам ефективний інструмент для організації навчального процесу в онлайн-форматі.

Розроблена програма допомагає автоматизувати подачу матеріалу. Також, передбачена можливість підтримки з боку викладачів, контроль свого профілю тощо.

Об'єкт дослідження цього проекту — це процес дистанційного навчання у вищих навчальних закладах. З урахуванням росту попиту на віртуальні курси та навчання на відстані, дана ініціатива акцентує увагу на аспектах ефективності та загальнодоступності освітніх послуг через мережу Інтернет.

Предмет дослідження охоплює розгляд користувацького інтерфейсу, структури системи та швидкодії транзакцій між frontend- та backend- частиною. Особливий акцент робиться на створенні модулів для проведення курсів та організації завдань.

Мета розробки полягає у створенні стабільної, масштабованої та інтуїтивно зрозумілої системи, яка, завдяки дружньому GUI, спростила процес навчання. Завданням є об'єднання практик у сфері електронного навчання для створення середовища, де кожен студент може досягти вищих академічних досягнень.

Завдання розробки даної програми включають:

- **Забезпечення гнучкості навчання:** Створення умов, за яких студенти можуть вивчати матеріал у зручний для них час і в зручному темпі, що сприяє самостійному навчанню.
- **Підтримка взаємодії:** Розробка інструментів для ефективної комунікації між студентами та викладачами, включаючи форуми і чати.
- **Доступність:** На курс може зареєструватись будь-хто з бажаючих, відсутні будь які обмеження.

- **Навявність особистого кабінету:** кожен користувач матиме змогу керувати своїм профілем.
- **Комбінована подача матеріалу:** матеріал подається у відео та текстовому форматі.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області. Обґрунтування необхідності використання ПЗ в предметній області

Основа аналізу предметної області електронного навчання полягає в розумінні базових потреб користувачів, які використовують платформу для доступу до курсів. На цьому етапі основна увага приділяється створенню інтуїтивно зрозумілого інтерфейсу, що дозволяє користувачам легко знаходити і обирати курси, а також переглядати їх вміст. Ефективне управління профілями користувачів також є важливим, оскільки воно дозволяє користувачам особисто налаштовувати свої дані.

Аналіз управління курсами має велике значення для платформ електронного навчання. Адміністратори повинні мати можливість легко створювати та оновлювати курси, включаючи різні модулі навчання і ресурси. Забезпечення легкості цього процесу впливає на швидкість розгортання нових навчальних матеріалів та їх актуалізацію, що є ключовим для підтримки актуальності освітніх програм в динамічному світі знань.

Аналіз споживчого досвіду при перегляді курсів є критичним компонентом. Важливо, щоб студенти могли не тільки легко доступати до курсів, але й зручно переглядати навчальні матеріали. Це включає забезпечення зручності навігації, чіткості представлення контенту та можливості відстежувати свій прогрес по курсу. Крім того, ефективність завантаження контенту та його сумісність із різними пристроями та операційними системами також є важливими аспектами для забезпечення доступності та задоволення потреб користувачів.

1.2 Аналіз програм-аналогів

Coursera пропонує широкий спектр курсів від світових університетів, що забезпечує її популярність серед студентів, які шукають якісну освіту. Гнучкість в організації навчання дозволяє користувачам самостійно вибирати темп і час навчання, що є великою перевагою для зайнятих людей. Інтеграція верифікованих

сертифікатів також мотивує студентів до завершення курсів.

Udemy відрізняється своєю відкритістю для викладачів, які бажають розміщувати та монетизувати свої курси. Це приваблює професіоналів з різних сфер, що робить контент Udemy надзвичайно різноманітним і актуальним. Платформа підтримує швидке оновлення матеріалів, що гарантує їх відповідність сучасним вимогам ринку.

Khan Academy сконцентрована на наданні безкоштовних освітніх ресурсів, що робить її доступною для широкого кола учнів по всьому світу. Інтерактивні уроки та наочні відео значно покращують залучення студентів та їх здатність засвоювати матеріал, а постійний зворотній зв'язок сприяє персоналізації навчального процесу.

LinkedIn Learning фокусується на розвитку професійних навичок, необхідних для кар'єрного зростання. Платформа інтегрована з профілем користувача на LinkedIn, дозволяючи демонструвати набуті навички потенційним роботодавцям, що є значною перевагою для професіоналів, зацікавлених у просуванні своєї кар'єри.

1.3 Технічне завдання на розробку ПЗ

Як ми знаємо, сьогодні кожна людина, будь то студент, чи вчитель, чи працює за професією тощо, використовує інтернет не лише для розваг, а й до самоосвіти, з будь-якої точки планети. Це не що інше, як електронне навчання. Отже, щоб розробити таку систему, де всі курси, які ви хочете вивчити, будуть доступні у вас під рукою, цей проект розроблено на основі існуючих програм для електронного навчання.

Посередником процесів є адміністратор, який додаватиме курси та теми відповідно до курсів, доданих до нього; він додає PDF-документ для кожної теми, а також відеоінструкцію. Студент – це користувач, який переглядатиме курси та може записатися на них. Ролі студентів також мають розділ для обговорення, де, якщо у студента є запитання, він/вона може його поставити. Останньою роллю є

викладач, чия відповідальність полягає в тому, щоб переглядати запити, створені студентами, і вирішувати їх.

Таким чином, розділення по ролям виглядає наступним чином:

1) Адміністратор (фасилітатор)

- Адміністратор має можливість переглядати існуючі курси.
- Можливість додавання нових курсів.
- Адміністратор відповідає за введення нових тем до курсів.
- Перегляд власного профілю.

2) Студент

- Студентам надано доступ до перегляду курсів.
- Вони можуть використовувати пошук для виявлення потрібних курсів.
- Запис на курси доступний для студентів.
- Студенти мають доступ до перегляду тем курсів.
- Навчання може відбуватись через PDF-документи або відеоматеріали.
- Можливість завантаження навчальних матеріалів у форматах PDF і відео також передбачена.
- Студенти можуть ставити запитання на форумі.
- Перегляд власного профілю.

3) Викладач

- Викладачі мають доступ до перегляду запитань від студентів.
- Вони можуть надавати відповіді на поставлені запитання.
- Викладачі також можуть переглядати свій особистий профіль.

1.4. Впровадження продукту

Підготовчий етап до запуску

Перед введенням в експлуатацію нашої платформи для електронного навчання слід здійснити низку підготовчих кроків. Зокрема, життєво важливим є ретельне випробування усіх функцій системи, щоб забезпечити їх надійність і продуктивність. Організація діалогу з майбутніми користувачами, включно з анкетуванням потенційних студентів і викладачів, дозволить краще зрозуміти їх

вимоги та переваги. Ці дані допоможуть оптимізувати платформу для забезпечення максимальної користі і зручності.

Стратегія запуску та реклама

Лаунч продукту передбачає розробку комплексного маркетингового плану і стратегії розповсюдження. Ключ до успіху — ідентифікація ефективних каналів зв'язку, які допоможуть досягти цільової аудиторії, включаючи соціальні мережі, освітні блоги та інші платформи онлайн-комунікації. Виробництво відео, що демонструють переваги та особливості платформи, також сприятиме популяризації продукту.

Нагляд та удосконалення після запуску

На постзапусковому етапі важливо встановити процеси для систематичного моніторингу результативності платформи. Виявлення слабких місць та можливостей для удосконалень є неминучим, тому збір відгуків від користувачів є незамінним для коригування і поліпшення системи. Регулярне оновлення програмного забезпечення, вдосконалення користувацького інтерфейсу та додавання нових функцій слід виконувати з огляду на отримані зворотні зв'язки та динаміку розвитку освітніх технологій.

Висновки до розділу 1:

Цей розділ деталізує основні компоненти нашого проекту електронного навчального ресурсу, окреслюючи його важливість у покращенні якості обслуговування, збільшенні ефективності адміністрування та зручності для користувачів. Проект включає розробку інтуїтивно зрозумілого графічного інтерфейсу для студентів, систему для інтеграції нових курсів, можливості для доступу до електронних курсів, а також створення персональних кабінетів для користувачів.

Зазначені ініціативи спрямовані на те, щоб перетворити взаємодію користувачів з платформою на більш ефективний та приємний процес. Введення дружнього графічного інтерфейсу користувача полегшує навігацію та підвищує загальну продуктивність навчального процесу. Механізм завантаження нових

курсів і доступ до електронних ресурсів забезпечує стале оновлення навчальних матеріалів і сприяє неперервному розвитку освітнього потенціалу користувачів. Особисті кабінети збільшують контроль користувачів над їхнім освітнім процесом і дозволяють ефективно керувати їхнім навчальним процесом у розрізі roadmap.

2. РОЗРОБКА ПЗ ДЛЯ КУРСОВОГО ПРОЄКТУ

2.1 Вибір інструментарію для реалізації ПЗ для курсового проєкту

Вибір інструментів для створення ПЗ курсового проекту є вирішальним аспектом, що впливає на продуктивність розробки, гнучкість адаптації до нововведень та зручність управління кінцевим продуктом. У рамках розробки даного проекту було обрано використання Eclipse як IDE, Java як головної мови програмування, а SQL було вибрано для керування базами даних.Розглянемо ці продукти детальніше.

Eclipse посідає провідні позиції серед середовищ розробки, що підтримують Java, пропонуючи розширені можливості для написання коду, його налагодження та тестування. Як платформа, Java забезпечує чудову переносимість програм, дозволяючи їм ефективно працювати на будь-якій системі, що підтримує Java-віртуальну машину. Ця мова є ідеальним вибором для розробки програмного забезпечення, що функціонує на різноманітних операційних системах, завдяки своїй багатій колекції стандартних інструментів та фреймворків.

Java стала основою для безлічі програмних проектів завдяки своїй гнучкості, об'єктно-орієнтованому дизайну та широкому застосуванню у розробці додатків різної складності — від мобільних додатків до обширних корпоративних систем. У сферах веб-розробки та створення корпоративних додатків Java надає надійне середовище з великим вибором бібліотек і фреймворків, які сприяють розробці масштабованих, продуктивних та захищених веб-аплікацій. Важливими інструментами у портфоліо Java для вирішення цих завдань є JDBC (Java Database Connectivity) та Spring Framework. JDBC — це API Java, яке дозволяє підключення до баз даних і виконання SQL-запитів, незалежно від специфіки систем управління базами даних. Така модульність є незамінною у сфері веб-розробки, де ставляться високі вимоги до обробки, збереження даних і їхньої безпеки.

Фреймворки Java є незамінними інструментами у світі розробки програмного забезпечення, оскільки вони надають вже готові архітектурні шаблони та набори компонентів, які спрощують створення додатків. Це дозволяє розробникам сконцентруватися на унікальних особливостях своїх проектів,

мінімізуючи потребу в розробці загальноприйнятих функцій з нуля. Фреймворки, такі як Spring, Hibernate і Struts, надають інструменти для ефективного управління даними, реалізації безпеки, створення веб-інтерфейсів і багато іншого. Вони забезпечують додаткам структуру, підтримують їх інтеграцію з різноманітними технологіями та базами даних, сприяють швидкому запуску та масштабуванню. Завдяки високому рівню абстракції і великій ефективності, Java фреймворки стали основним інструментарієм для розробки сучасних програмних рішень. Розглянемо основні з них нижче.

Spring Framework займає чільне місце серед найвідоміших і переважно вибраних фреймворків розробниками завдяки його універсальності, високій продуктивності та інтуїтивно зрозумілому інтерфейсу. Модульна структура цього фреймворку надає розробникам Java значні переваги, що відкривають нові можливості на кожному етапі процесу розробки програм. Водночас, фреймворк спрощує взаємодію з розширеними можливостями та інструментами, що пришвидшує процес створення проектів.

Ця потужна платформа дозволяє Java-розробникам ефективно реалізувати та інтегрувати широкий спектр функцій, включаючи аутентифікацію та верифікацію користувачів, забезпечуючи при цьому високий рівень безпеки для додатків та їх користувачів. Корпорації такі як Netflix, Dick’s Sporting Goods, eBay та багато інших вже скористалися цими можливостями. Однак, Spring так само ефективний і для розробки менших проектів завдяки своїй стабільності та потужності.

Spring Framework надає широкий набір функціональностей, включаючи MVC для розробки веб-додатків, аспектно-орієнтоване програмування (AOP) та інверсію контейнерів керування (IoC), які є корисними для створення мікросервісів. Станом на 2023 рік, Spring продовжує бути достойним вибором для веб-розробників, оскільки він добре підтримується, широко документований і користується великою популярністю серед розробників по всьому світу.

Apache є однією з найвідоміших та найбільших відкритих платформ, що протягом багатьох років служила основою для створення численних

розробницьких інструментів. Фреймворк переважно використовується для створення веб-додатків на Java. Struts, як інструмент базований на архітектурі Model-View-Controller (MVC), дозволяє розробляти потужні та масштабовані веб-додатки.

Еластичність та можливість розширення Struts роблять його ідеальним доповненням до портфеля інструментів веб-розробника, завдяки чому він стає незамінним ресурсом для різноманітних проектів. Фреймворк вирізняється на тлі інших завдяки великій колекції плагінів та акценту на користувацькій зручності. Розробники можуть застосувати його для розробки різноманітних застосунків, від комплексних веб-сайтів до високобезпечних додатків, що гарантують захист даних та верифікацію користувачів.

Struts пропонує багатий набір функціоналу, зокрема вбудовані можливості для перевірки даних, розширену архітектуру, безперебійну інтеграцію з іншими Java-технологіями, а також підтримку AJAX і JSON. Фреймворк оптимально підходить для розробки додатків, що вимагають модульної та чистої архітектури.

JSF є відкритим фреймворком для веб-розробки, який спеціалізується на створенні інтуїтивно зрозумілих і функціонально насичених користувацьких інтерфейсів за допомогою Java. Цей фреймворк постійно еволюціонує і переважно застосовується для розробки складних користувацьких інтерфейсів і динамічних веб-додатків. За допомогою JSF розробники можуть використовувати різноманітні компоненти, які полегшують створення користувацьких інтерфейсів, тим самим розширюючи функціонал веб-сторінок та веб-додатків.

Фреймворк JSF інтегрує JavaServer Pages (JSP) і JavaServer Servlet API для створення структури JavaServer Face, а також підтримує взаємодію з джерелами даних через протокол HTTP, використовуючи підключення TCP або UDP. Також він підтримує стандарти HTML 4.01 (HTML) і XHTML 1.0 для розробки веб-додатків. Функціонал JSF включає, але не обмежується, анотаціями bean, впровадженням технології Facelets, підтримкою HTML5, інтеграцією мови Expression для стандартної обробки винятків і швидкої веб-розробки. Фреймворк

ідеально підходить для конструювання розгорнутих користувацьких інтерфейсів та динамічних веб-додатків, відповідаючи потребам сучасних веб-розробників.

Bootstrap є потужним інструментом для створення фронтенду, який широко використовується у поєднанні з Java Server Pages (JSP) для розробки респонсивних та адаптивних веб-інтерфейсів. Цей фреймворк надає великий вибір стильових компонентів та утиліт, що дозволяють розробникам ефективно і швидко створювати візуально привабливі веб-сторінки. Використання Bootstrap у JSP-додатках сприяє оптимізації процесу розробки, гарантуючи консистентний вигляд веб-сторінок на різноманітних пристроях, та вивільняє час для зосередження на більш складних задачах.

Інтегрування Bootstrap в JSP проходить через додавання необхідних CSS та JavaScript файлів до проекту. Розробники можуть застосовувати різноманітні класи Bootstrap у своїх JSP-файлах для стилізації HTML-елементів, включаючи форми, кнопки та навігаційні панелі. Цей підхід значно пришвидшує розробку, дозволяючи розробникам фокусуватися на складніших аспектах серверної логіки, тоді як основні візуальні вимоги ефективно вирішуються завдяки функціональності Bootstrap.

Vaadin є Java-платформою, спеціалізованою на створенні передових веб-додатків. Розробники можуть писати код інтерфейсу користувача безпосередньо на Java, що забезпечує автоматизацію зв'язку між клієнтом і сервером та управління DOM у фоновому режимі. Такий підхід значно спрощує створення складних інтернет-додатків, зосереджуючись на серверній логіці. Завдяки цим перевагам, Vaadin є популярним вибором серед Java-розробників, які прагнуть створити гнучкі веб-інтерфейси. Платформа функціонує на сервері, автоматично обробляючи зв’язок клієнт-сервер та забезпечуючи швидку розробку інтерфейсу завдяки наявності готових компонентів. Vaadin також пропонує прямий Java API для HTML, що мінімізує потребу в глибоких знаннях HTML або JavaScript. Поставляється з настроюваними темами та параметрами макета, Vaadin полегшує створення привабливих інтерфейсів. Крім того, вбудований маршрутизатор

дозволяє управляти навігацією та URL, підтримуючи функції односторінкових програм безперешкодно.

Hibernate — це Java-платформа, що спрощує створення Java-додатків для роботи з базами даних. Як інструмент ORM (Object Relational Mapping) з відкритим кодом, Hibernate дозволяє легко взаємодіяти з реляційними базами даних. Він реалізує стандарти Java Persistence API (JPA) та забезпечує перетворення Java-класів у таблиці бази даних та Java-типів даних у SQL-типи. Розробка Hibernate мала на меті усунення розбіжностей між об'єктно-орієнтованими мовами програмування та системами управління реляційними базами даних (RDBMS).

Hibernate відомий своєю можливістю автоматичного зіставлення ORM, яке автоматизує відображення між Java-класами і таблицями бази даних, спрощуючи тим самим базові операції з даними. Цей інструмент також є незалежним від конкретної бази даних, що дозволяє розробникам легко змінювати бази даних без необхідності переписувати Java-код. Вбудований механізм кешування підвищує продуктивність застосувань, мінімізуючи частоту звернень до бази даних. Функція відкладеного завантаження оптимізує виконання, забезпечуючи завантаження даних лише за необхідності. Нарешті, Hibernate пропонує гнучкі можливості формулювання запитів через Hibernate Query Language (HQL) та критеріальні запити, що збільшує ефективність управління даними.

Обрання SQL як інструменту для роботи з базами даних визначається потребою у систематичному зберіганні, доступі та аналізі даних. SQL сприяє ефективній взаємодії з реляційними базами даних, здійснюючи комплексні запити для оптимального управління інформацією. Використання реляційних систем управління базами даних забезпечує організоване зберігання даних та їх можливість до масштабування, що є надзвичайно важливим для курсових проектів з перспективою збільшення обсягів інформації та посилення взаємодії між бізнес-логікою додатків та збереженою інформацією.

Керування залежностями проєкту покладено на Maven. Maven є потужним інструментом для управління проектами, що базується на POM (Project Object

Model). Він використовується для компіляції проектів, управління залежностями та генерації документації, полегшуючи процес зборки подібно до ANT, але надає більш просунуті можливості. Можна стверджувати, що Maven є ідеальним інструментом для створення та адміністрування будь-яких Java-базованих проектів, спрощуючи роботу Java-розробників та допомагаючи краще розуміти проекти, розроблені на Java.

Maven переважно використовується для компіляції Java-проектів та створення веб-пакетів. Він пропонує різноманітні можливості, які полегшують збірку веб-додатків, дозволяючи ефективно керувати складними проектами.

Файли POM, які є XML-документами, містять всю необхідну інформацію про проект, включно з залежностями, конфігурацією вихідних каталогів, плагінами та іншими параметрами, що використовує Maven для збірки проекту. Коли виконується команда Maven, вона використовує файл POM для виконання встановлених налаштувань та процесів.

Залежності в Maven визначають зовнішні Java-бібліотеки, необхідні для проекту, тоді як сховища — це місця зберігання пакетів JAR. Якщо залежності не знайдені в локальному репозиторії, Maven автоматично завантажує їх з центрального репозиторію та зберігає локально.

Життєвий цикл зборки в Maven містить послідовність фаз, кожна з яких включає певні цілі. Запуск команди Maven для певного життєвого циклу призводить до автоматичного виконання усіх відповідних фаз.

Профілі зборки в Maven дозволяють адаптувати конфігурацію проекту для різних умов зборки, наприклад, для розробки або тестування на локальному комп'ютері. Це можна здійснити шляхом включення різних профілів у файл POM.

Плагіни зборки в Maven використовуються для досягнення специфічних цілей у процесі зборки. Ви можете додати стандартний плагін до файлу POM або реалізувати свій власний на Java для задоволення конкретних потреб проекту.
Maven є визначним інструментом для створення артефактів Java-проектів з використанням файлу POM.XML. Життєвий цикл зборки в Apache Maven

складається з добре визначених етапів і цілей, які виконуються у строгій послідовності для ефективного створення та управління проектом.

Основний життєвий цикл має кілька підциклів, описаних нижче:

- Перевірка: здійснює перевірку правильності конфігурації проекту.
- Компіляція: перетворює вихідний код у байт-код.
- Тестування: виконує тестування проекту.
- Пакування: упаковує скомпільований код та ресурси у артефакт, такий як JAR або WAR.
- Інсталяція: встановлює артефакт у локальному репозиторії.
- Розгортання: передає артефакт у віддалене сховище.

2.2 Проектування та реалізація БД

Структура таблиць показана на рисунку нижче. Дані користувача зберігаються у таблиці User. Інформація про створені курси зберігається у таблиці course. Кожен курс може містити окремі розділи. Інформація про ниї а також, про додані матеріали зберігається у таблиці topic. Додатково, кожен користувач мое задати певне запитання. Відповідями займається викладач. Інформація про їх діалог зберігається у таблиці raisedQuery. Таблиця enroll служить для того, щоб зв’язати користувача і курс, який він проходить.

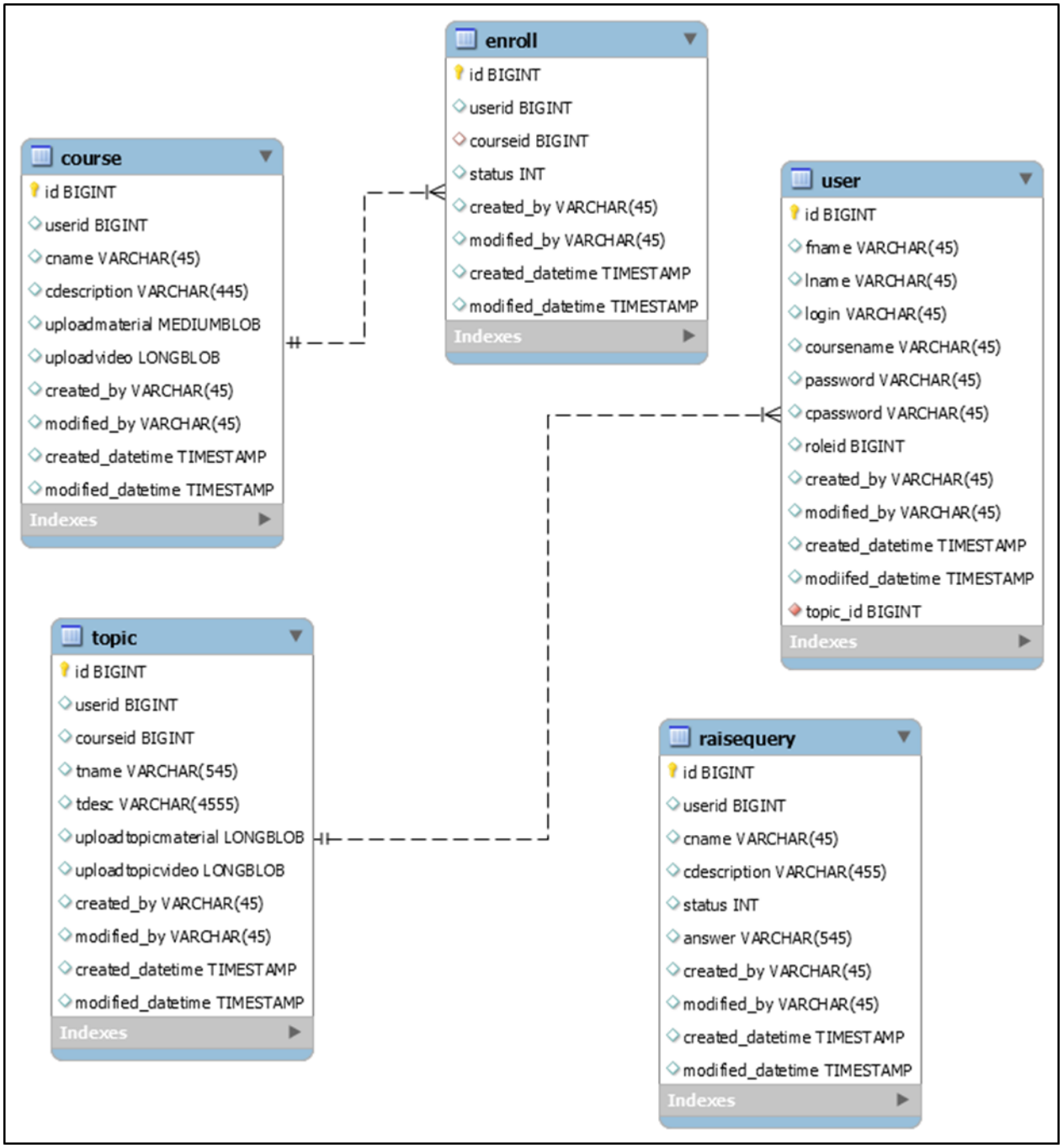

Рис. 2.1. Макет бази даних

2.3. Опис класів та програмних модулів

Програмний додаток складається з трьох основних компонентів. Перша частина - база даних, створена за допомогою SQL-сервера. Вона інтегрована з програмою, розробленою на платформі Eclipse, за допомогою конфігураційних файлів. Ці файли, написані в форматі XML, містять вказані налаштування, які автоматично імпортуються при завантаженні додатків.

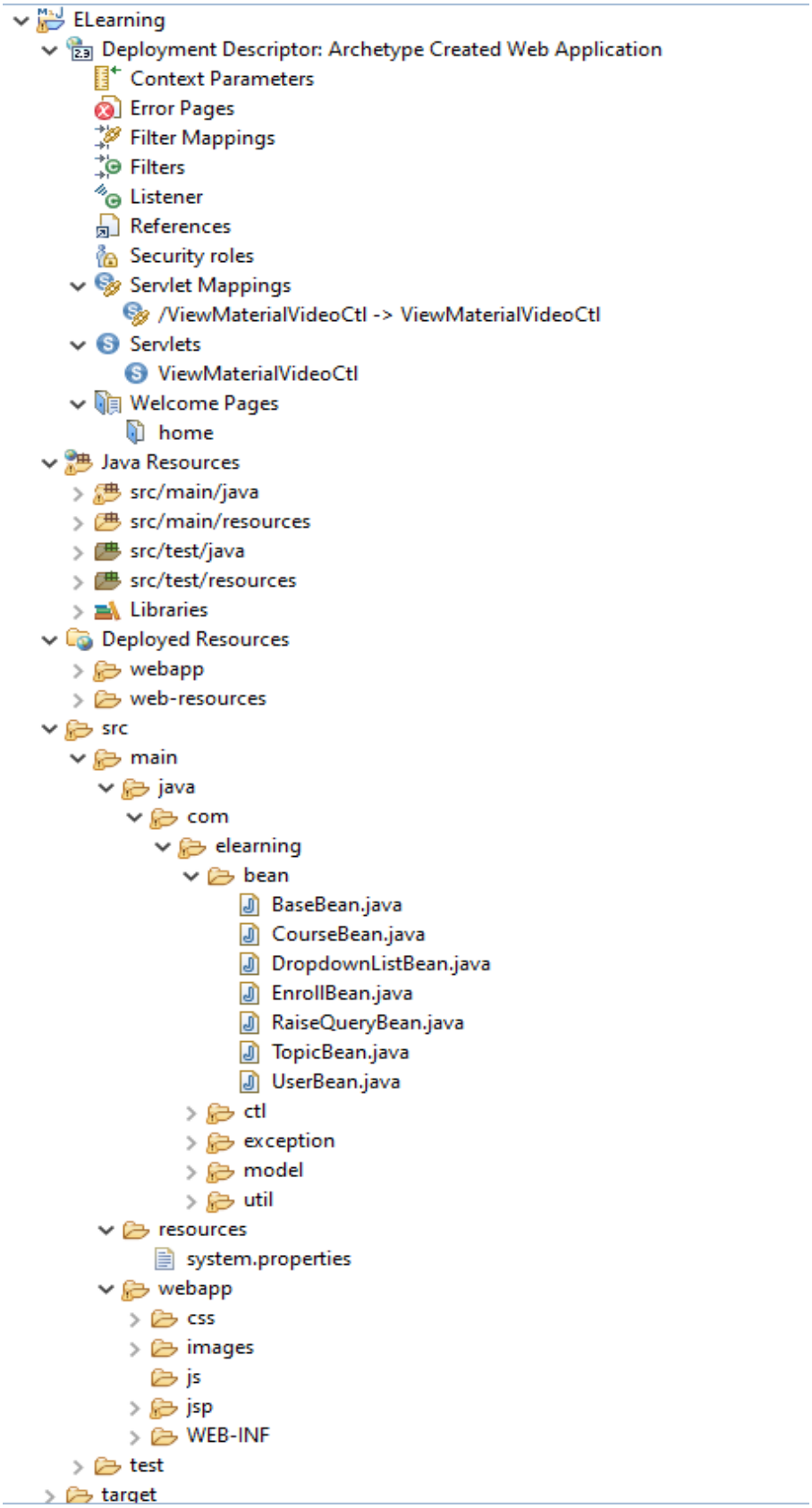

Рис. 2.2. Будоів проекту

Backend-частина:

- bean – Тут зберігаються основні біни продукту. Бін – це клас, яким керує Spring. При старті програми вони записуються в Spring Context. Це дозволяє імпортувати їх та використовувати в будь-якому місці коду.
- ctl – містить класи, що являють собою форми у frontend- частині.
- exception – містить кастомні класи для відображення помилок. Вони допомагаються при роботі, оскільки дозволяють легше діагностувати проблему та швише її вирішити. Також, вони слугують елементом покращення GUI, оскільки відображають помилки в зрозумілому для користувача вигляді.
- model – тут зібрані класи, що використовуються для взаємодії з БД. Модель являє собою сутність таблиці у базі, що спрощує роботу в java-коді з нею.
- util - містить допоміжні класи з статичними методами, які використовуються в коді.
- resources - тут зібрані змінні, які використовуються для запуску програми та роботи Spring.

Frontend-частина:

- css – слугує для збереження файлів, що відповідають за елементи стилю.
- images – тут зберігаються картинки, що використовуються для фону додатку.
- js та jsp – тут містяться веб-сторінки, які включають js-код та елементи динамічної взаємодії з java.
- Web-inf – відповідає за поєднання відповідних класів та js-view.

2.4. Огляд рольової моделі продукту

В додатку реалізовано три ролі. Вони умовно розбиті на незалежні модулі, такі як Facilitator, Student, Faculty. Нижче, схматично, зображено їх можливості, у відповідності до ТЗ.

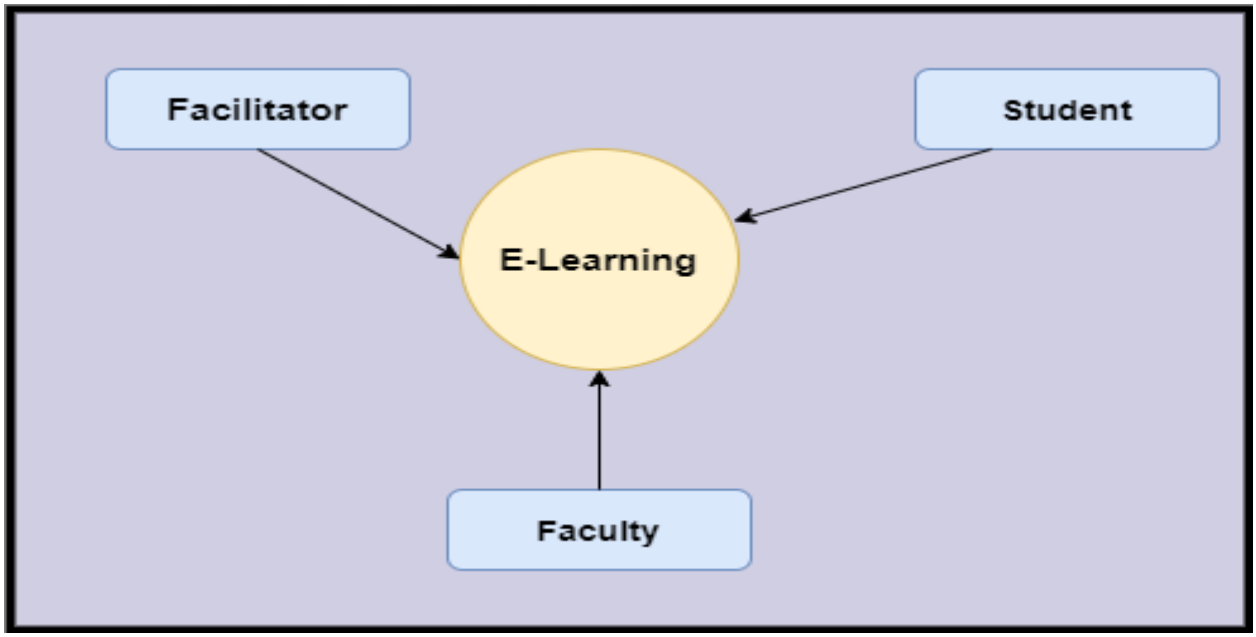

Рис. 2.3. Схема проекту

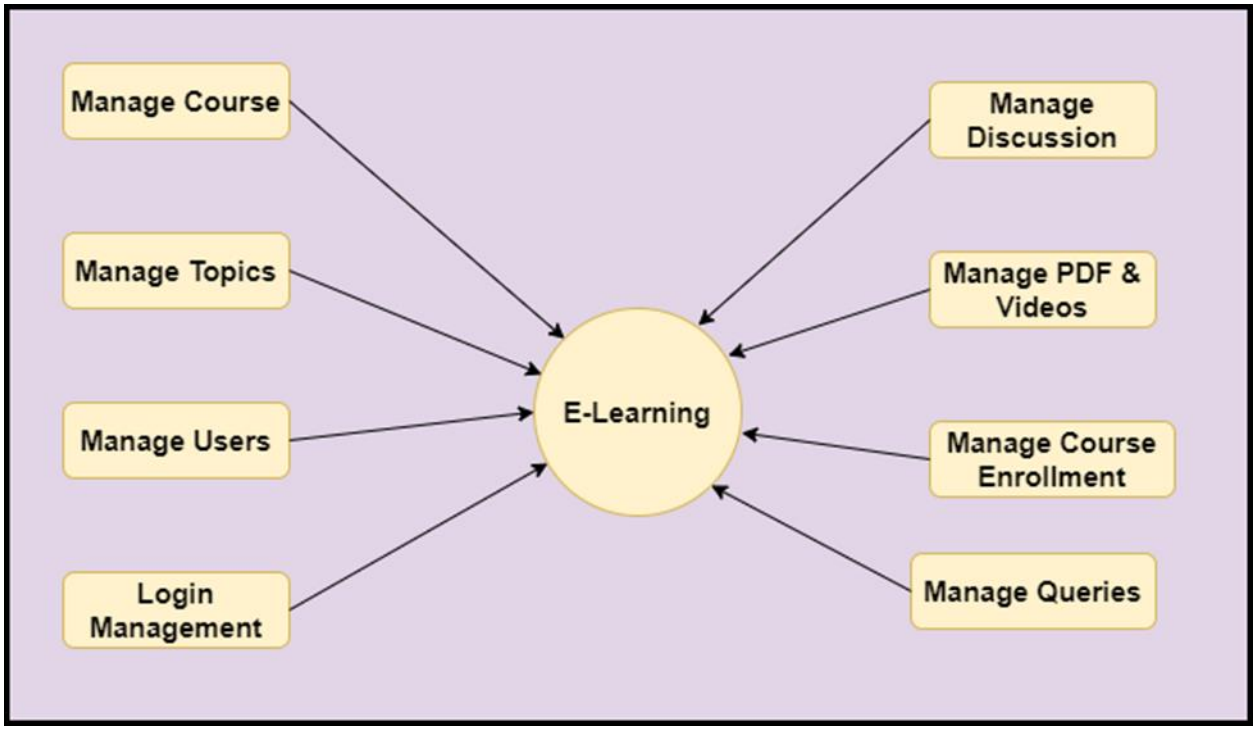

Рис. 2.4. Модулі проекту

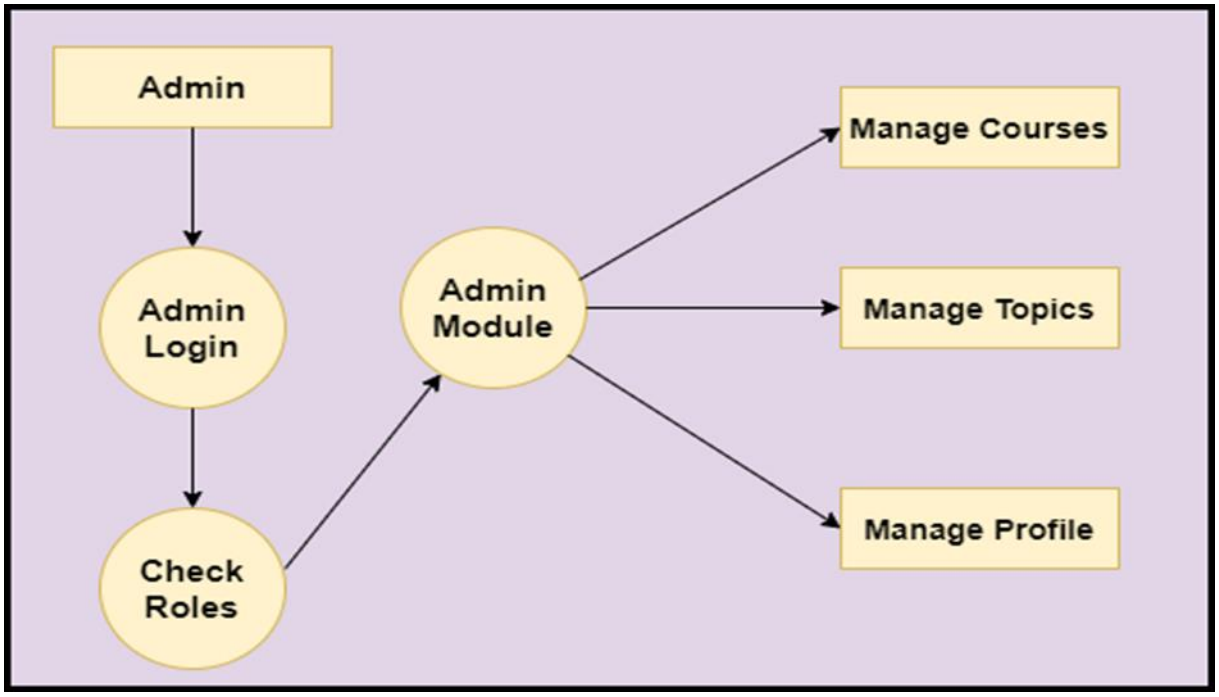

Рис. 2.5. Можливості адміністратора

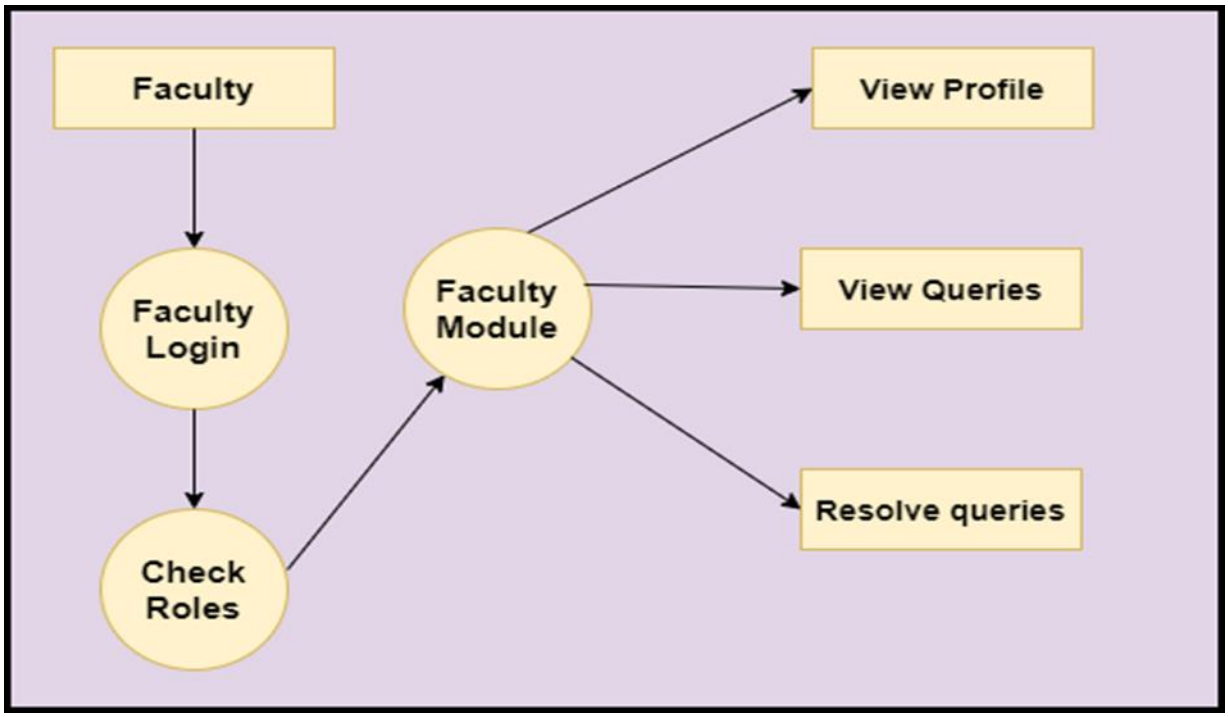

Рис. 2.6. Можливості викладача

2.5 Створення та тестування програмованого продукту

Для тестування продукту застосовано локальний сервер Tomcat. При запуску додатку ми переходимо на стартову сторінку.

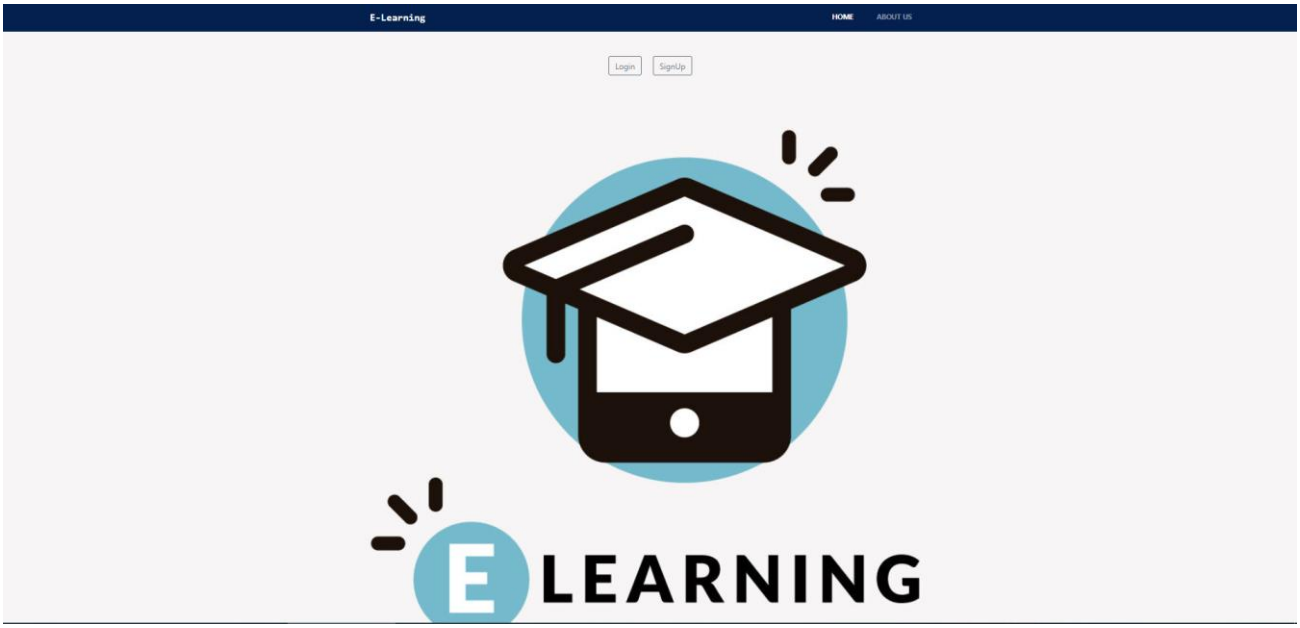

Рис. 2.7 – Стартова сторінка

Для входу доступні 3 ролі, необхідно вибрати свою і ввести дані у форму.

Рис. 2.8 – Сторінка реєстрації

Для додавання курсів виконаємо вхід під адміністратором.

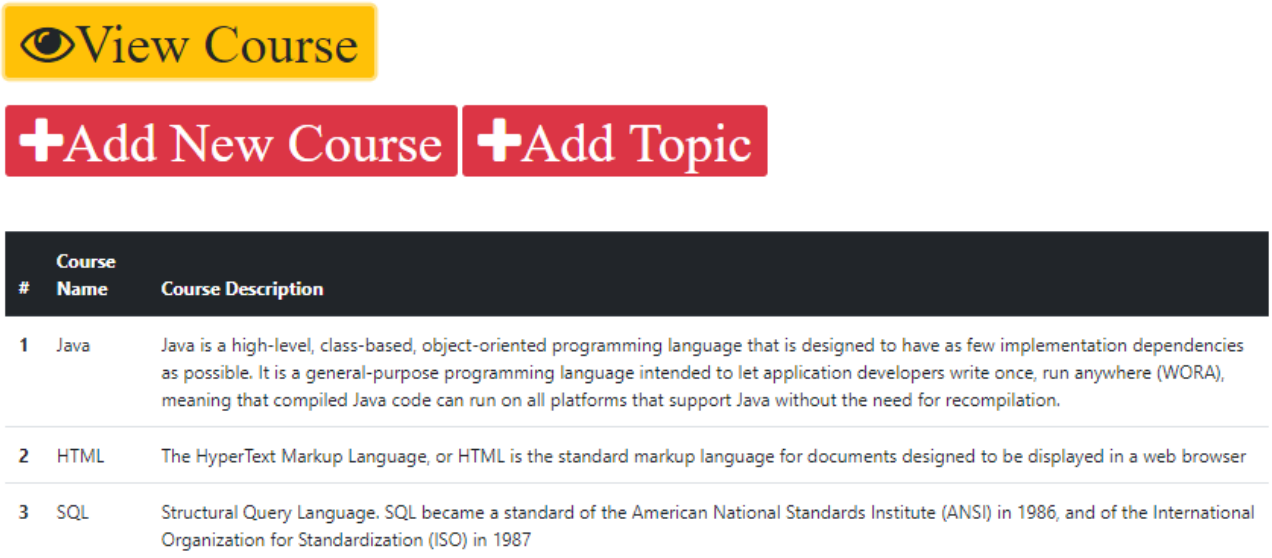

Рис. 2.9 – Перегляд курсів

При вході адміністратор має змогу переглянути існуючі курси з їх детальним описом. Додатково, він може створити новий курс або додати туди деталі(topic).

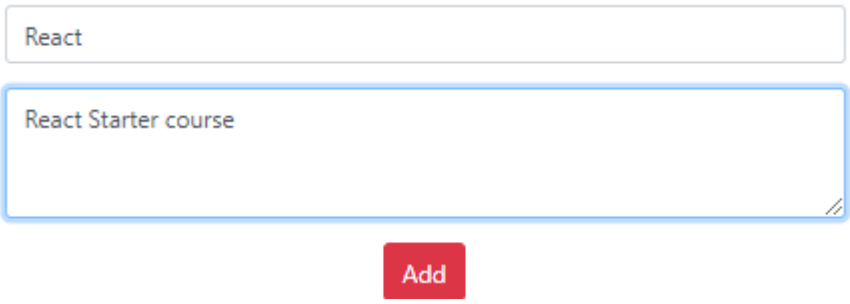

Рис. 2.10 – Створення нового курсу

Наприклад, створимо курс по React. Він включає назву та опис, після чого можна додавати туди відповідні матеріали. Після створення курс доступний у списку перегляду.

Рис. 2.11. Оновлення списку курсів

Тепер задача адміністратора наповнити курс матеріалами. Для цього він натискає на «Add Topic», обирає потрібний курс і приступає до завантаження. Необхідно пронайменувати розділи а також додати відеофайл і текстовий документ pdf. Це дозволяє врахувати інтереси ширної групи студентів в плані зручності навчання.

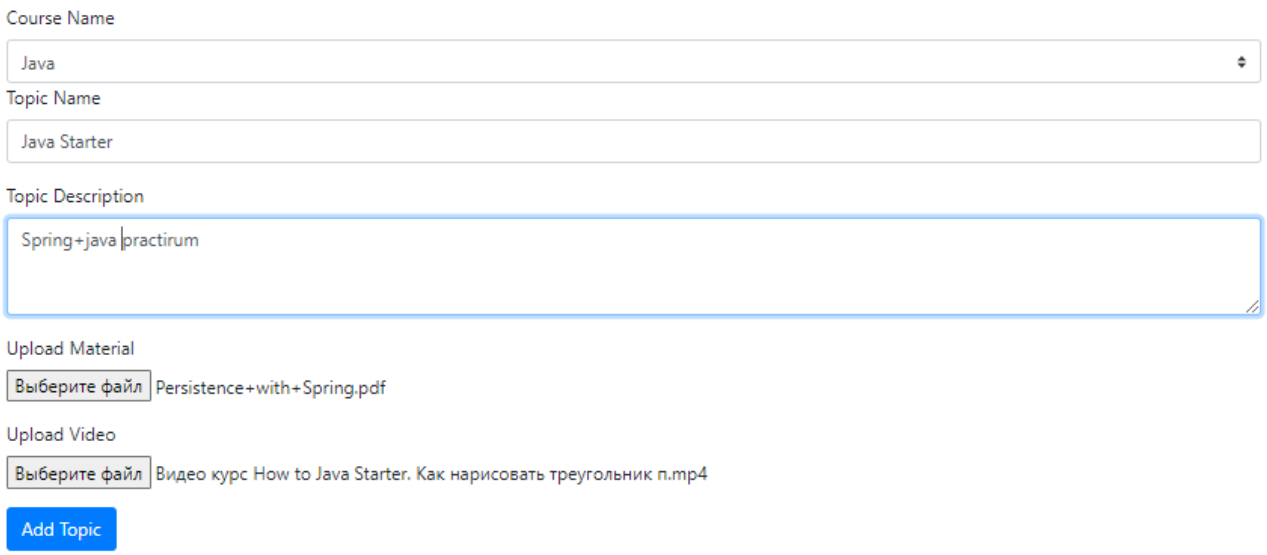

Рис. 2.12. Новий topic

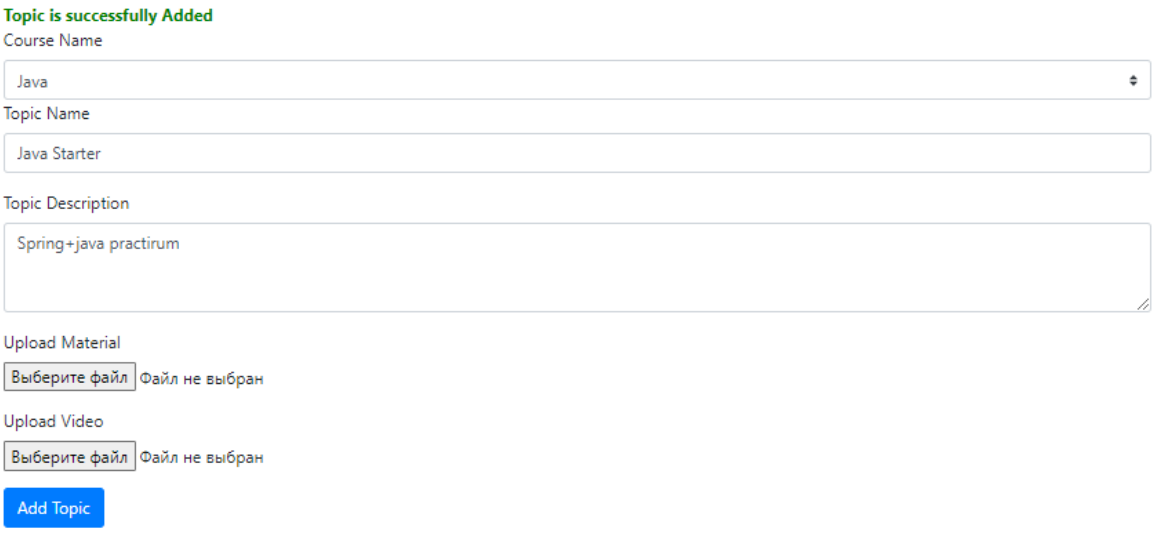

Рис. 2.13. Підтвердження

Якщо далі зайти під користувачем, який має доступ до даного курсу, то ми зможемо проглянути оновлення.

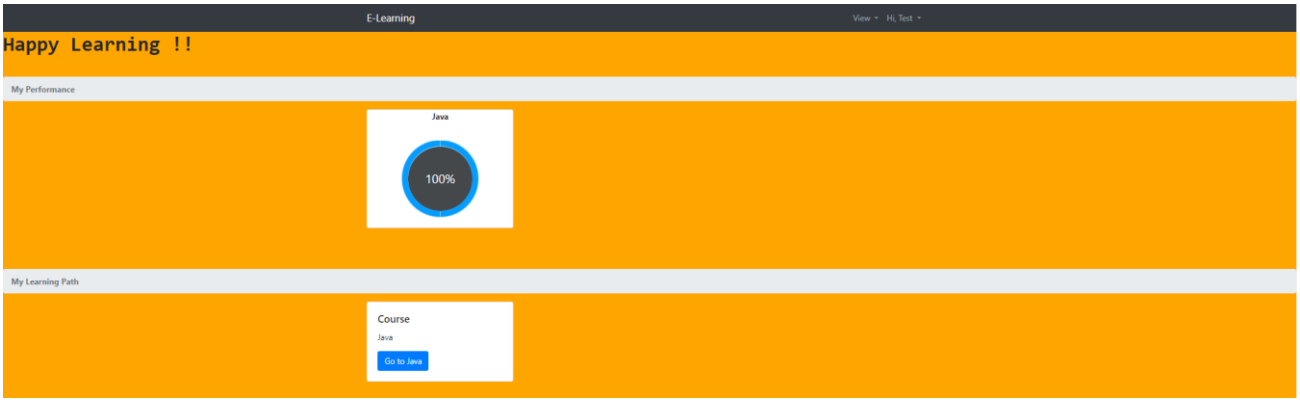

Рис. 2.13. Доступні курси

Рис. 2.14 – Перегляд тем

Можемо обрати будь-яку тему і проглянути матеріали.

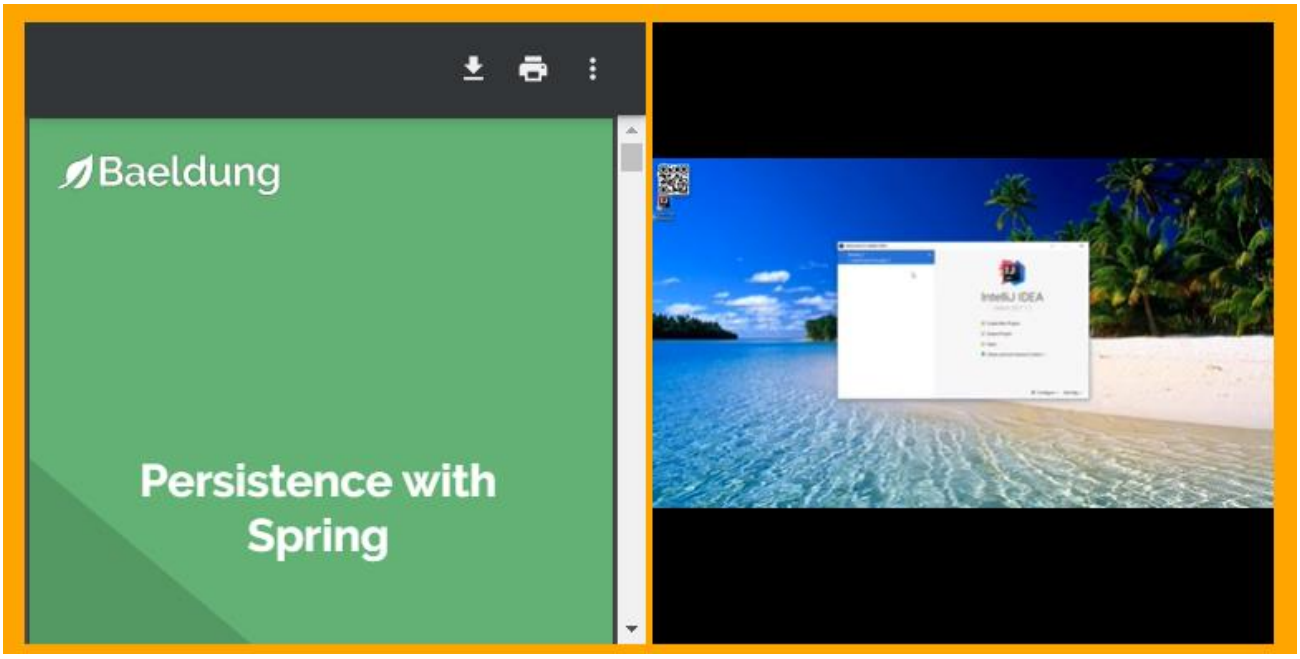

Рис. 2.15 – Перегляд матеріалів

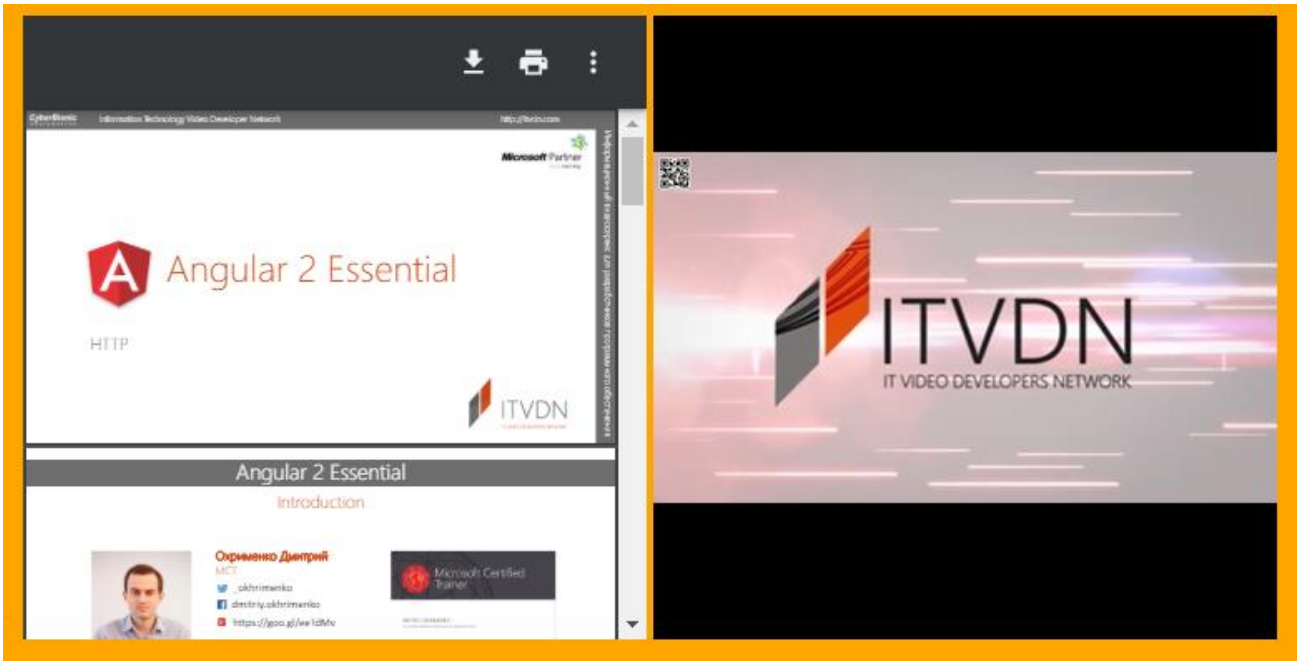

Рис. 2.16. Перегляд наступної теми

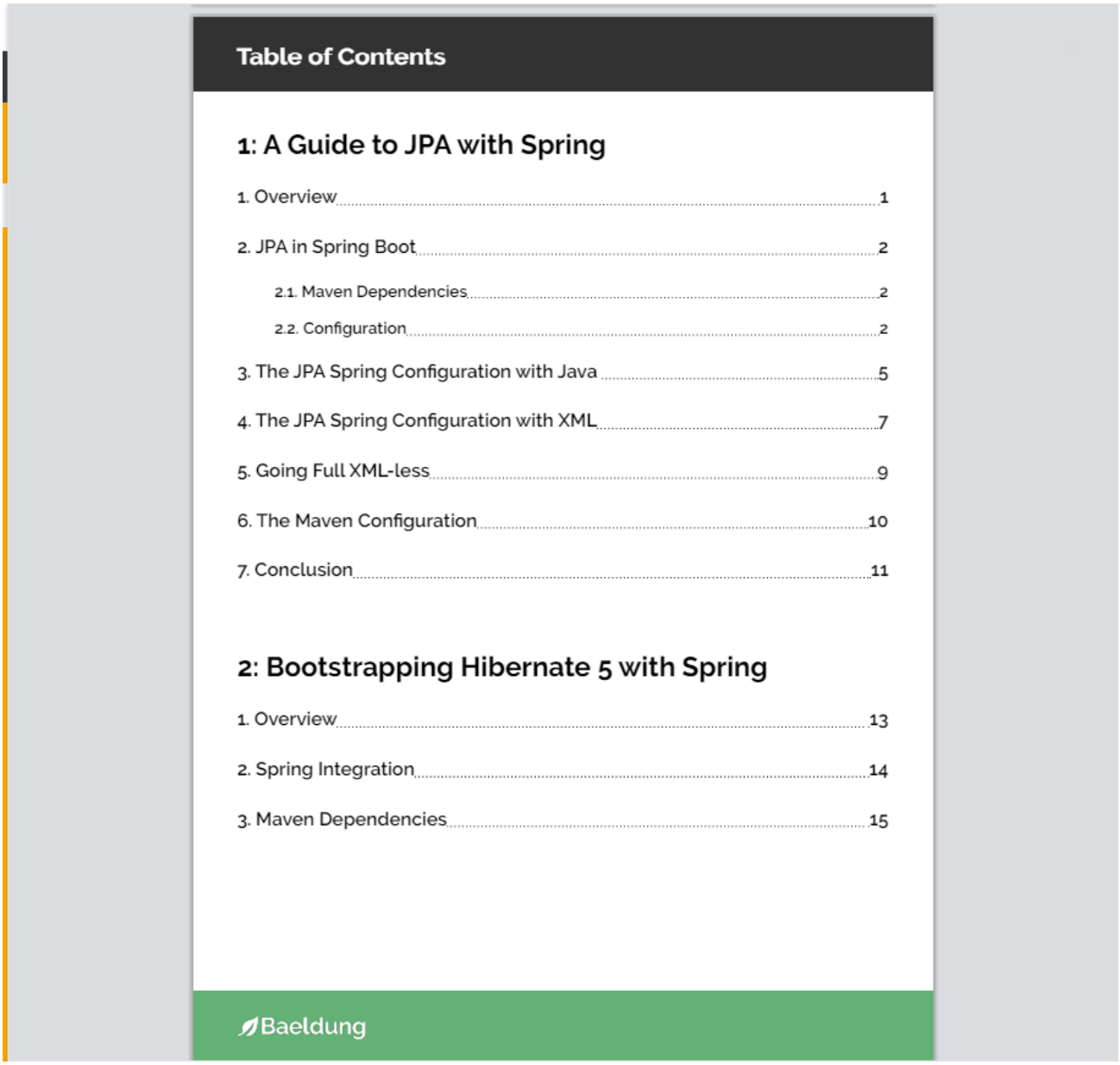

Рис. 2.17. Детальний перегляд тексту

Також, студент має змогу самостійно записатись на курс.

Рис. 2.18. Перегляд усіх курсів

Зрозуміло, що у процесі навчання можуть виникати запитання, тому студент має змогу задати їх, і отримати відповідь від викладачів.

Рис. 2.19 – Перегляд історії запитань

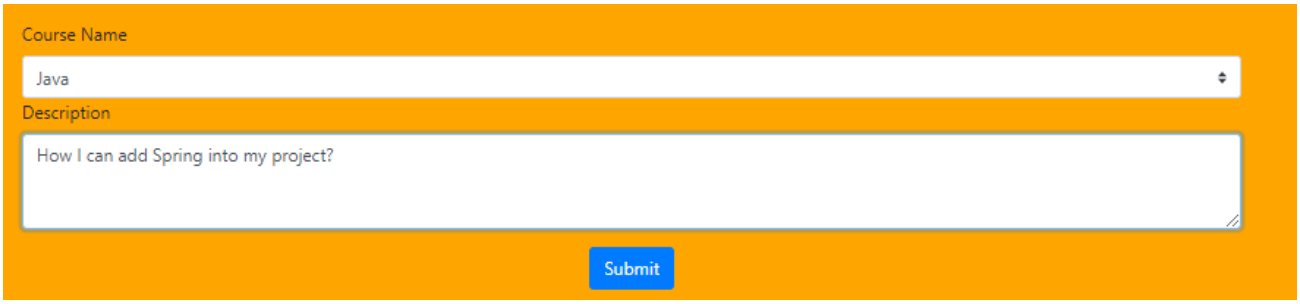

Рис. 2.20 Нове запитання

Рис. 2.21. Відповідь викладача

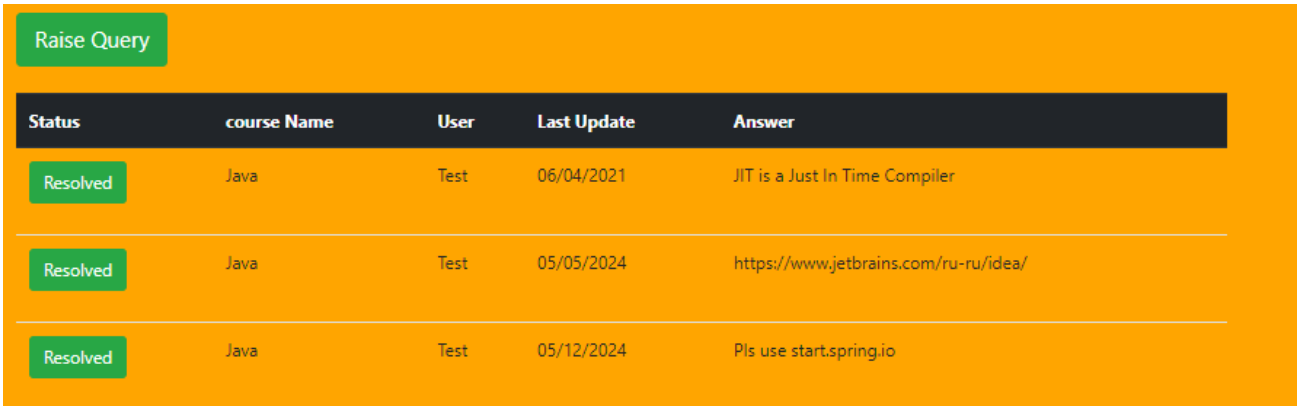

Рис. 2.22. Перегляд відповіді

Отже, тестування програмного продукту пройшло успішно.

Висновки до розділу 2

Проведено аналіз ПЗ, структуру БД та всього додатку. Розглянуто основні можливості веб-додатку: створення курсу, наповнення їх матеріалами, перегляд, функціонал підтримки.

ВИСНОВКИ

У ході реалізації курсової роботи було розроблено систему електронного навчання на базі Java, яка відповідає сучасним вимогам до адаптивних та ефективних інформаційних систем. Результати дослідження та практичної реалізації показали наступні ключові досягнення:

1. **Аналіз наявних рішень:** Було проведено ґрунтовний огляд існуючих систем управління бібліотеками з метою виявлення актуальних трендів у цій сфері. Дослідження дозволило визначити ефективні стратегії для побудови та введення в експлуатацію подібних інформаційних систем, що стало основою для розробки власного рішення.

2. **Вибір програмного забезпечення:** Для кожного компонента архітектури MVC (Model-View-Controller) було обрано відповідне програмне забезпечення. Це включало вибір бази даних, серверної частини та клієнтського інтерфейсу. Використання передових технологій дозволило створити адаптивну систему, яка легко піддається модифікаціям відповідно до мінливих умов використання та технологічних оновлень.

3. **Реалізація та тестування:** Завершено створення та інтеграцію програмного рішення. Мануальне тестування системи підтвердило її стабільність, високу продуктивність та зручність у використанні. Система електронного навчання виявилася надійною платформою, яка здатна ефективно задовольняти потреби користувачів.

Таким чином, розроблена система електронного навчання на Java є сучасним та ефективним інструментом, що відповідає вимогам ринку та забезпечує високий рівень функціональності і зручності у використанні. Вона готова до подальших покращень та адаптацій, що дозволяє підтримувати актуальність та конкурентоспроможність у динамічних умовах сучасних інформаційних технологій.

ПЕРЕЛІК ДЖЕРЕЛ

1. "Інформаційні технології в медицині" / О. І. Михайлова, В. В. Стадник. - «Професіонал», 2016. 15 с.
2. "Digital Medicine: Health Care in the Internet Era" - Darrell M. West, 2015. 32 с.
3. Spring in Action, Fifth edition / Craig Walls - «Manning Publications»б 2018. 12-62 с.
4. Герберт Шилдт. Java: The Complete Reference, 10-те видання - М.: "Вільямс", 2018. - С. 1022-1132.
5. Роберт С Мартін. Clean Code: A Handbook of Agile Software Craftsmanship- Л.: "Технокнига", 2020. - С. 530.
6. М.Ф. Копитко,. Основи програмування мовою Java - Х.: " Видавничий центр ЛНУ імені Івана Франка ", 2002. - С. 60.
7. Роберт Л. Мартін. Чистий код: Створення, аналіз та рефакторинг - М.: "Програміст", 2016. - С. 264.
8. Ерік Фрімен. Head First Design Patterns- О.: " Вільямс ", 2018. - С. 288.
9. Весті, Р. (2020). Освоєння Java JDBC: З'єднання з базами даних. 3-тє видання - TechPress, 334 с.
10. Гудвілл, Дж., Ліланд, М. (2018). Професійний Spring Boot 2: Авторитетний посібник зі створення мікросервісів, веб- та підприємницьких додатків, та кращих практик. Видавництво Apress, 648 с.
11. Маркхам, Н. (2016). Вивчення Bootstrap 4, створюючи проекти. Видавництво Packt Publishing, 292 с.
12. Дауні, П. (2017). SQL для початківців: Все в одному. 2-ге видання - For Dummies, 768 с.
13. Джонсон, Р., Голлер, Дж., Арендсен, А., Рісберг, Т., Сампалеану, К. (2018). Професійне Java програмування з фреймворком Spring. Видавництво Wiley, 672 с.
14. Кларк, С. (2019). Початкове Java програмування: Підхід, орієнтований на об'єкти. Видавництво Wiley, 960 с

15. Peter T.Knight(2003). Employability and Good Learning in Higher Education, ResearchGate. 112 с
16. Бауер, К., Кінг, Г., Грегорі, Г. (2015). Java Persistence з Hibernate. 2-ге видання - Manning Publications, 608 с.
17. Сільбершатц, А., Корт, Г.Ф., Сударшан, С. (2019). Концепції систем баз даних. 7-ме видання - McGraw-Hill Education, 1376 с.

ДОДАТКИ

Додаток А

Лістинг коду

```
@WebServlet(name = "RaiseQueryCtl", urlPatterns = {"/home/login/student/dashboard/raisequery"})
public class RaiseQueryCtl extends BaseCtl {
    private static final long serialVersionUID = 1L;
    public static final String OP_SIGN_UP = "SignUp";

  /**
   * @see HttpServlet#HttpServlet()
   */
  public RaiseQueryCtl() {
    super();
    // TODO Auto-generated constructor stub
  }

  @Override
  protected boolean validate(HttpServletRequest request) {
    boolean pass =true;


            if (DataValidator.isNull(request.getParameter("tdesc"))) {
                request.setAttribute("tdesc",
                                PropertyReader.getValue("error.require", "Topic Description"));
                System.out.println("Pass 4 "+pass);
                pass = false;
            }
            System.out.println("pass "+pass);
            return pass;
  }

  @Override
  protected BaseBean populateBean(HttpServletRequest request) {
    RaiseQueryBean bean = new RaiseQueryBean();
    bean.setId(DataUtility.getLong(request.getParameter("id")));
    bean.setCourseName(DataUtility.getString(request.getParameter("cname")));
```

```
    bean.setCourseDescription(DataUtility.getString(request.getParameter("tdesc")));
    bean.setStatus(2);

    populateDTO(bean, request);
    return bean;
}


    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
                ServletUtility.forward(getView(), request, response);
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
                System.out.println("******in do post*****");
                        String op = DataUtility.getString(request.getParameter("operation"));
                        System.out.println("op is "+op);
                    // get model
                    QueryModel model = new QueryModel();
                    long id = DataUtility.getLong(request.getParameter("id"));
                    System.out.println("*****here-here****");
                    if (OP_SIGN_UP.equalsIgnoreCase(op)) {
                        System.out.println("*****here****");
                        RaiseQueryBean bean = (RaiseQueryBean) populateBean(request);
                       HttpSession session=request.getSession();
                       UserBean uBean=(UserBean)session.getAttribute("user");
                       bean.setUserId(uBean.getId());
                       try {
                          System.out.println("//////hello*****");
                             long pk = model.add(bean);
                             ServletUtility.setBean(bean, request);
```

```
                    ServletUtility.setSuccessMessage("Query is successfully Raised",request);
                    ServletUtility.forward(getView(), request, response);
                                        return;
                }
            catch(DuplicateRecordException e){
                ServletUtility.setBean(bean, request);
                                        ServletUtility.setErrorMessage("Login id already exists",
                                                request);
                                        ServletUtility.forward(getView(), request, response);
            } catch (ApplicationException e) {
                                        // TODO Auto-generated catch block
                                        e.printStackTrace();
                                    }


            }

    }

    @Override
    protected String getView() {
            // TODO Auto-generated method stub
            return ELearnView.RAISE_QUERY_VIEW;
    }

}
@WebServlet(name = "ImageServlet", urlPatterns = {"/ImageServlet"})
@MultipartConfig(maxFileSize = 16177215)

public class ImageServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

  /**
   * @see HttpServlet#HttpServlet()
   */
  public ImageServlet() {
```

```
    super();
    // TODO Auto-generated constructor stub
}

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
            // TODO Auto-generated method stub
            System.out.println("hello");
            int id = Integer.parseInt(request.getParameter("id"));

            BufferedInputStream bin=null;
        BufferedOutputStream bout=null;
        InputStream in =null;
        response.setContentType("application/pdf");
        ServletOutputStream out;
        out = response.getOutputStream();
            try {

                    Connection con=JDBCDataSource.getConnection();
                    String sql="SELECT * FROM TOPIC WHERE ID="'+id+'"";
                    PreparedStatement ps=con.prepareStatement(sql);
                    ResultSet rs=ps.executeQuery();
                    if (rs.next()) {
               /* byte[] imgData = rs.getBytes("photo");//Here..... rs.getBytes() extract byte data from resultSet
                System.out.println(imgData);
                response.setHeader("expires", "0");
                response.setContentType("image/jpg");

                OutputStream os = response.getOutputStream(); // output with the help of outputStream
                os.write(imgData);
                os.flush();
                os.close();*/
                                    Blob blob = rs.getBlob("uploadtopicmaterial");
                            byte byteArray[] = blob.getBytes(1, (int)blob.length());
```

```
                                response.setContentType("application/pdf");
                                OutputStream os = response.getOutputStream();
                                os.write(byteArray);
                                os.flush();
                                os.close();

                        }

                        } catch (Exception e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();

                                 response.getOutputStream().flush();
                                 response.getOutputStream().close();
                        }


        }

        /**
         * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
         */
        protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
                // TODO Auto-generated method stub
                doGet(request, response);
        }

}
@WebServlet(name = "TopicRegCtl", urlPatterns = {"/home/login/facilitator/viewcourse/topic"})
@MultipartConfig(maxFileSize = 16177215)
public class TopicRegCtl extends BaseCtl {
        private static final long serialVersionUID = 1L;
        public static final String OP_SIGN_UP = "SignUp";

    /**
```

```
 * @see HttpServlet#HttpServlet()
 */
public TopicRegCtl() {
   super();
   // TODO Auto-generated constructor stub
}



@Override
protected void preload(HttpServletRequest request) {
   CourseModel model = new CourseModel();
   try {
                        List l = model.list();
                        System.out.println("is list returning?" + l.get(0));
                        request.setAttribute("courseList", l);
                        Iterator<CourseBean> itr=l.iterator();
                        while(itr.hasNext()){
                                CourseBean b=new CourseBean();
                                b=itr.next();
                                System.out.println("b"+b.getCourseName());
                        }

            } catch (ApplicationException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
            }
}
@Override
protected boolean validate(HttpServletRequest request) {
   boolean pass =true;
   if (DataValidator.isNull(request.getParameter("cname"))) {
                        request.setAttribute("cname",
                                            PropertyReader.getValue("error.require", "Cousre Name"));
                        pass = false;
                        System.out.println("Pass 1 "+pass);
            }
```

```
    if (DataValidator.isNull(request.getParameter("tname"))) {
                request.setAttribute("tname",
                        PropertyReader.getValue("error.require", "Topic Name"));
                System.out.println("Pass 3 "+pass);
                pass = false;
            }
            if (DataValidator.isNull(request.getParameter("tdesc"))) {
                request.setAttribute("tdesc",
                        PropertyReader.getValue("error.require", "Topic Description"));
                System.out.println("Pass 4 "+pass);
                pass = false;
            }
            /*if (DataValidator.isNull(request.getParameter("uploadmaterial"))) {
                request.setAttribute("uploadmaterial",
                        PropertyReader.getValue("error.require", "Upload Material"));
                pass = false;
            }
            if (DataValidator.isNull(request.getParameter("uploadvideo"))) {
                request.setAttribute("uploadvideo",
                        PropertyReader.getValue("error.require", "Upload Video"));
                pass = false;
            } */
            System.out.println("pass is "+pass);
    return pass;
}
public Blob MaterialUpload(Part part) throws IOException {
            InputStream inputStream = null;
            Blob blob = null;
            inputStream = part.getInputStream();
            byte[] b = new byte[inputStream.available()];
            inputStream.read(b);

            try {
                blob = new SerialBlob(b);

            } catch (SerialException e) {
```

```
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            return blob;
    }

@Override
protected BaseBean populateBean(HttpServletRequest request) {


    TopicBean bean = new TopicBean();
    bean.setId(DataUtility.getLong(request.getParameter("id")));
    bean.setCourseId(DataUtility.getLong(request.getParameter("cname")));
    bean.setTopicName(DataUtility.getString(request.getParameter("tname")));
    bean.setTopicDescription(DataUtility.getString(request.getParameter("tdesc")));
    Blob blob=null;
            Blob blob2=null;   // For Video Upload


                Part filepart;
                try {
                    filepart = request.getPart("uploadmaterial");
                    Part filepart2 = request.getPart("uploadvideo");
                    blob = MaterialUpload(filepart);
                    blob2 = MaterialUpload(filepart2);
                } catch (IOException | ServletException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
                 //Upload material method called


            bean.setUploadTopicMaterial(blob);
            bean.setUploadTopicVideo(blob2);
    populateDTO(bean, request);
```

```
    return bean;
}

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

                ServletUtility.forward(getView(), request, response);
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
                 System.out.println("******in do post*****");
                String op = DataUtility.getString(request.getParameter("operation"));
            // get model
            TopicModel model = new TopicModel();
            long id = DataUtility.getLong(request.getParameter("id"));
            if (OP_SIGN_UP.equalsIgnoreCase(op)) {
                System.out.println("*****here*****");
                TopicBean bean = (TopicBean) populateBean(request);
               HttpSession session=request.getSession();
               UserBean uBean=(UserBean)session.getAttribute("user");
               bean.setUserId(uBean.getId());
               try {
                  System.out.println("//////hello******");
                     long pk = model.add(bean);
                     ServletUtility.setBean(bean, request);
                     ServletUtility.setSuccessMessage("Topic is successfully Added",request);
                     ServletUtility.forward(getView(), request, response);
                                          return;
                  }
               catch(DuplicateRecordException e){
                  ServletUtility.setBean(bean, request);
```

```
                                        ServletUtility.setErrorMessage("Login id already exists",
                                                        request);
                                        ServletUtility.forward(getView(), request, response);
                } catch (ApplicationException e) {
                                        // TODO Auto-generated catch block
                                        e.printStackTrace();
                                }


            }
        }

        @Override
        protected String getView() {
                // TODO Auto-generated method stub
                return ELearnView.TOPIC_REG_VIEW;
        }

}
@WebServlet(name = "VideoServlet", urlPatterns = {"/VideoServlet"})
public class VideoServlet extends HttpServlet {
        private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public VideoServlet() {
        super();
        // TODO Auto-generated constructor stub
    }

        /**
         * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
         */
        protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
                System.out.println("hello");
```

```
        int id = Integer.parseInt(request.getParameter("id"));


        BufferedInputStream bin=null;
BufferedOutputStream bout=null;
InputStream in =null;
response.setContentType("video/mp4");
ServletOutputStream out;
out = response.getOutputStream();
        try {


                Connection con=JDBCDataSource.getConnection();
                String sql="SELECT * FROM TOPIC WHERE ID="'+id+'"";
                PreparedStatement ps=con.prepareStatement(sql);
                ResultSet rs=ps.executeQuery();
                if (rs.next()) {
    /* byte[] imgData = rs.getBytes("photo");//Here..... rs.getBytes() extract byte data from resultSet
     System.out.println(imgData);
     response.setHeader("expires", "0");
     response.setContentType("image/jpg");


     OutputStream os = response.getOutputStream(); // output with the help of outputStream
     os.write(imgData);
     os.flush();
     os.close();*/
                                Blob blob = rs.getBlob("uploadtopicvideo");
                        byte byteArray[] = blob.getBytes(1, (int)blob.length());


                        response.setContentType("video/mp4");
                        OutputStream os = response.getOutputStream();
                        os.write(byteArray);
                        os.flush();
                        os.close();


  }


        } catch (Exception e) {
```

```
                        // TODO Auto-generated catch block
                        e.printStackTrace();

                        response.getOutputStream().flush();
                        response.getOutputStream().close();
                    }

            }

            /**
             * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
             */
            protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
                    // TODO Auto-generated method stub
                    doGet(request, response);
            }

    }
    @WebServlet(name = "ViewMaterialVideoCtl", urlPatterns =
{"/home/login/student/dashboard/coursesearch/courseview/videomaterial"})
    @MultipartConfig(maxFileSize = 16177215)
    public class ViewMaterialVideoCtl extends BaseCtl {
        private static final long serialVersionUID = 1L;

       /**
        * @see HttpServlet#HttpServlet()
        */
       public ViewMaterialVideoCtl() {
          super();
          // TODO Auto-generated constructor stub
       }

            /**
             * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
             */
            protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
```

```
            String op = DataUtility.getString(request.getParameter("operation"));
// get model

            CourseModel model = new CourseModel();

            long id = DataUtility.getLong(request.getParameter("id"));

            if (id > 0 || op != null) {



                        CourseBean bean;
   try {
      bean = model.findByPK(id);

      ServletUtility.setBean(bean, request);

   } catch (ApplicationException e) {



      ServletUtility.handleException(e, request, response);
      return;
   }
}
            ServletUtility.forward(getView(), request, response);
}

/**
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
 */
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
            // TODO Auto-generated method stub
            doGet(request, response);
}

@Override
protected String getView() {
```

```
                // TODO Auto-generated method stub
                return ELearnView.MATERIAL_VIDEO_VIEW;
        }

}
%@page import="com.elearning.ctl.FacilitatorRegCtl"%>
<%@page import="com.elearning.util.ServletUtility"%>
<%@page import="com.elearning.util.DataUtility"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@include file="header.jsp"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Facilitator Page</title>
<link rel="stylesheet" href="../css/style.css">
<link rel="stylesheet" href="path/to/font-awesome/css/font-awesome.min.css">
</head>
<body>
    <div class="container">
        <div class="card mt-5 mx-auto text-center" style="width: 18rem;">

            <div class="card-body">
                <h5 class="card-title">Facilitator</h5>
                <form action="<%=ELearnView.FACILITATOR_REG_CTL%>" method="post">
                <jsp:useBean id="bean" class="com.elearning.bean.UserBean"
                        scope="request"></jsp:useBean>

                        <input type="hidden" name="id" value="<%=bean.getId()%>">
                        <input type="hidden" name="roleid" value="<%=1%>">
            <input type="hidden" name="createdBy" value="<%=bean.getCreatedBy()%>">
            <input type="hidden" name="modifiedBy" value="<%=bean.getModifiedBy()%>">
            <input type="hidden" name="createdDatetime" value="<%=DataUtility.getTimestamp(bean.getCreatedDatetime())%>">
            <input type="hidden" name="modifiedDatetime"
value="<%=DataUtility.getTimestamp(bean.getModifiedDatetime())%>">
```

```
                    <b><font color="red"> <%=ServletUtility.getErrorMessage(request)%>

                </font></b>


                <b><font color="Green"> <%=ServletUtility.getSuccessMessage(request)%></font></b>

                    <div  class="form-group"> <input

                                                type="text" class="form-control" id="exampleInputEmail1"

                                                aria-describedby="emailHelp" name="fname"
value="<%=DataUtility.getStringData(bean.getFirstName())%>" placeholder="First Name">

                                        <font  color="red"><%=ServletUtility.getErrorMessage("fname", request)%></font>

                                        </div>

                                        <div  class="form-group"> <input

                                                type="text" class="form-control" id="exampleInputEmail1"

                                                aria-describedby="emailHelp" name="lname"
value="<%=DataUtility.getStringData(bean.getLastName())%>" placeholder="Last Name">

                                        <font  color="red"><%=ServletUtility.getErrorMessage("lname", request)%></font>

                                        </div>

                                        <div  class="form-group"> <input

                                                type="email" class="form-control" id="exampleInputEmail1"

                                                aria-describedby="emailHelp" name="login"
value="<%=DataUtility.getStringData(bean.getLogin())%>" placeholder="Email/Username">

                                        <font  color="red"><%=ServletUtility.getErrorMessage("login", request)%></font>

                                        </div>

                                        <div class="form-group">

                                                <input

                                                type="password" name="password" class="form-control"
value="<%=DataUtility.getStringData(bean.getPassword())%>" id="exampleInputPassword1"

                                                placeholder="Password">

                                                <font
color="red"><%=ServletUtility.getErrorMessage("password", request)%></font>

                                        </div>

                                        <div class="form-group">

                                                <input

                                                type="password" name="cpassword"
value="<%=DataUtility.getStringData(bean.getConfirmPassword())%>" class="form-control" id="exampleInputPassword1"

                                                placeholder="Retype Password">

                                                <font
color="red"><%=ServletUtility.getErrorMessage("cpassword", request)%></font>

                                        </div>
```

```
                                                    <button type="submit" name="operation" class="btn btn-primary"
value="<%=FacilitatorRegCtl.OP_SIGN_UP%>">Submit</button>

                                                </form>


                                        </div>

                                </div>

                        </div>



                        <div style="margin-top: 500px">

                                <%@ include file="footer.jsp"%>

                        </div>

                </body>

                </html>

                <%@page import="java.sql.ResultSet"%>

                <%@page import="java.sql.PreparedStatement"%>

                <%@page import="com.elearning.util.JDBCDataSource"%>

                <%@page import="java.sql.Connection"%>

                <%@page import="com.elearning.ctl.CourseRegCtl"%>

                <%@page import="com.elearning.util.ServletUtility"%>

                <%@page import="com.elearning.util.DataUtility"%>

                <%@ page language="java" contentType="text/html; charset=ISO-8859-1"

                        pageEncoding="ISO-8859-1"%>

                <%@include file="header.jsp"%>

                <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

                <html>

                <head>

                <style type="text/css">

                #form1 {

                        display: none;

                }

                #form2{

                display: none;

                }

                #form3{

                display: none;
```

```
}
</style>
<link rel="stylesheet"
    href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Dashboard</title>
</head>
<body>
    <div class="container mt-5">


            <button type="button" id="formButton2" class="btn btn-warning">
                <i class="fa fa-eye fa-3x" aria-hidden="true">View Course</i>
            </button><br><br>
            <button type="button" id="formButton" class="btn btn-danger">
                <i class="fa fa-plus fa-3x" aria-hidden="true">Add New Course</i>
            </button>
            <a class="btn btn-danger" href="<%=ELearnView.TOPIC_REG_CTL%>"><i class="fa fa-plus fa-3x" aria-hidden="true">Add Topic</i></a>
    </div>
    <div class="container">
            <div class=" mt-5 mx-auto text-center" style="width: 35rem;">


                <div class="">
                    <form id="form1" method="post" action="<%=ELearnView.Cousre_Reg_CTL%>">


                        <jsp:useBean id="bean" class="com.elearning.bean.CourseBean"
                            scope="request"></jsp:useBean>


                        <input type="hidden" name="id" value="<%=bean.getId()%>"> <input
                            type="hidden" name="roleid" value="<%=2%>"> <input
                            type="hidden" name="createdBy" value="<%=bean.getCreatedBy()%>">
                        <input type="hidden" name="modifiedBy"
                            value="<%=bean.getModifiedBy()%>"> <input type="hidden"
                            name="createdDatetime"
                            value="<%=DataUtility.getTimestamp(bean.getCreatedDatetime())%>">
```

```
                                        <input type="hidden" name="modifiedDatetime"
                                            value="<%=DataUtility.getTimestamp(bean.getModifiedDatetime())%>">
                                        <b><font color="red"> <%=ServletUtility.getErrorMessage(request)%>
                                        </font></b><b><font color="Green">
<%=ServletUtility.getSuccessMessage(request)%></font></b>

                                        <div class="form-group">
                                            <input type="text" class="form-control" id="exampleInputEmail1"
                                                aria-describedby="emailHelp" name="cname"
        value="<%=DataUtility.getStringData(bean.getCourseName())%>"
                                                placeholder="Course Name"> <font
color="red"><%=ServletUtility.getErrorMessage("login", request)%></font>
                                        </div>
                                        <div class="form-group">

                                            <textarea name="cdesc" class="form-control"
                                                id="exampleFormControlTextarea1" rows="3"
placeholder="Course Description"></textarea>
                                        </div>

                                        <button type="submit" name="operation" class="btn btn-danger"
                                            value="<%=CourseRegCtl.OP_SAVE%>">Add</button>
                                    </form>
                                </div>
                            </div>
                        </div>
    <!-- TOPIC ADD START -->

    <!-- TOPIC END  -->
    <div class="container">
    <table class="table" id="form2">
      <thead class="thead-dark">
        <tr>
          <th scope="col">#</th>
          <th scope="col">Course Name</th>
```

```
      <th scope="col">Course Description</th>
      <!-- <th scope="col">Upload</th> -->
    </tr>
  </thead>
  <tbody>
  <%
  int index=1;
  Connection conn = JDBCDataSource.getConnection();
        String sql = "SELECT * FROM COURSE";
        PreparedStatement ps = conn.prepareStatement(sql);
        ResultSet rs = ps.executeQuery();
        while(rs.next()){

  %>
    <tr>
      <th scope="row"><%=index++ %></th>
      <td><%=rs.getString(3) %></td>
      <td><%=rs.getString(4) %></td>
     <%--  <td><a href="viewcourse/upload?id=<%=rs.getInt(1)%>">Upload</a></td> --%>
    </tr>

    <%} %>
  </tbody>
</table>




</div>
        <script type="text/javascript">
$("#formButton").click(function(){
    $("#form1").toggle();
});
/* JQUERY FOR TABLE */
  $("#formButton2").click(function(){
            $("#form2").toggle();
  });
```

```
 /* JQUERY FOR TOPIC ADDITION */
 $("#formButton3").click(function(){
        $("#form3").toggle();
 });
</script>
    <div style="margin-top: 500px">
            <%@ include file="footer.jsp"%>
    </div>
</body>
</html>
```