math_2022_039

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРНІВЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЮРІЯ ФЕДЬКОВИЧА

Факультет математики та інформатики
кафедра математичного моделювання

РОЗРОБКА ОНЛАЙН-ПЛАТФОРМИ ДЛЯ НАВЧАННЯ ТА ПРЕЗЕНТАЦІЇ КУРСІВ ДЛЯ СТУДЕНТІВ ЗАСОБАМИ ФРЕЙМВОРКА RUBY ON RAILS

Кваліфікаційна робота

Рівень вищої освіти – другий (магістерський)

Виконав:
студент 6 курсу, 607 групи
Ілащук Микола Миколайович
Керівник*:*
кандидат фізико-математичних наук,
доцент Олександр Матвій

До захисту допущено
на засіданні кафедри
протокол № 8 від 6 грудня 2022 р.
Зав. кафедрою _____ проф. Черевко І.М.

Чернівці – 2022

АНОТАЦІЯ

Дана робота присвячена створенню інтернет платформи для організації онлайн навчання.

В даній роботі було створено онлайн «майданчик» за допомогою високорівневої мови програмування загального призначення Ruby, інструменту для створення користувацьких інтерфейсів Ruby on Rails та мови структурованих запитів PostgreSQL. Для дизайну користувацького інтерфейсу було використано Bootstrap.

This master’s thesis is devoted to the creation of an Internet platform for the organization of online education.

In this project, an online "playground" was created using the high-level general-purpose programming language Ruby, the tool for creating user interfaces Ruby on Rails, and the structured query language PostgreSQL. Bootstrap was used to design the user interface.

Ключові слова: ***Ruby, Ruby on Rails, онлайн навчання, платформа, публікація матеріалу.***

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

_____ М.М. Ілащук
(підпис)

ЗМІСТ

ВСТУП

Постійне навчання та здобуття навичок сьогодні, як досвід показує, є необхідністю, щоб залишатися конкурентноздатним на ринку. В залежності від того чи потрібно отримати інформацію, знання чи відпрацювати навик навчання можна умовно поділити на лекції, майстер-класи, тренінги, тощо. В даний час більшості людям доцільно використовувати інтернет ресурси, особливо, коли мова йде про навчання. Також варто зазначити, що дана тема набрала більшої популярності саме після пандемії.

В даній магістерській роботі розглянута задача створення онлайн-платформи для навчання використовуючи MVC патерн. Для цього було розроблено архітектуру додатку, структуру бази даних з використанням технологій DSL міграцій та інтерфейс, що складається із загальнодоступної та адміністративної частини.

Мета магістерської роботи – створити платформу, як для навчання "студентами", так і для завантаження курсів викладачами.

Були поставлені наступні завдання:

1. дослідження та аналіз веб-сайтів подібної тематики, порівняння їхніх функціоналів та виділення їх переваг та недоліків;
2. вибір технологій та мов програмування;
3. бути завжди в тренді, проходячи курси з найпопулярніших технологій;
4. надати "майданчик" для викладачів щодо зручного відображення курсів для студентів.

При створенні проекту були використані наступні технології:

- мова програмування Ruby, JavaScript;
- фреймворк Ruby on Rails (RoR);
- технології для створення веб-сторінок: HTML та CSS;
- бібліотека Bootstrap;
- мови структурованих запитів PostgreSQL;
- середовище RubyMine (в навчальних цілях);

- Heroku – для хостингу проекту;
- GitHub – для контролю версій та зберігання проекту.

Дана дипломна робота складається із вступу, двох частин та висновків:

- Перша частина містить опис використаних технологій, такі як: Ruby, Ruby on Rails, Bootstrap, фронтенд технології, Ruby бібліотеки (gems), Heroku;
- Друга частина містить опис предметної області, аналіз схожих веб-сайтів (переваги та недоліки). Також викладено опис розробленої частини програмного продукту, інструкція для користувачів, описано структуру БД, об`єкти, контролери, дизайн частина.

Дана дипломна робота була написана за допомогою Ruby та Ruby on Rails технологій, які дають можливість написати бізнес-логіку проекту, JavaScript, Bootstrap, HTML та CSS дають можливість створити користувацький інтерфейс та PostgreSQL - створити базу даних.

§1. Опис використаних технологій

1.1. Мова програмування Ruby

Веб-технології ніколи не були такими вишуканими і гнучкими як сьогодні. Якщо користувач хоче створити якісний веб-сайт або веб-додаток, він без зусиль знайде масу інструментів для реалізації будь-яких своїх задумів. І один з них - Ruby. Всупереч поширеній думці Ruby корисний не тільки для обробки даних, а й скриптів чи написання логіки для обробки невеликої кількості даних. На даний час Ruby стало мовою широкого спектру використання через те, що середовище розробників Ruby (англ. – Ruby community) почало розвивати дану мову з боку швидкості та багатопоточності.

Ruby [1] - це мова “ретельного балансу”. Його творець Юкіхіро “Мац” Мацумото змішав частини своїх улюблених мов (Perl, Smalltalk, Eiffel, Ada та Lisp), щоб сформувати нову мову, яка збалансувала функціональне програмування з імперативним програмуванням. Найчастіше Ruby використовується з веб-додатками. Його головне звернення полягає в тому, що програмістам не потрібно витрачати багато часу на налаштування своїх файлів.

Ruby набирає популярності і бібліотека Ruby on Rails допомогла збільшити її використання для веб-програмування. Дана мова повністю об'єктно-орієнтована, оскільки все є об'єктом окрім деяких випадків: умовних та логічних операторів. Наприклад, навіть самі основні типи даних, такі як цілі числа, мають методи та змінні екземпляра. Це забезпечує більшу можливість використовувати ланцюжок методів, де багато рядків коду можна об’єднати в один і в результаті код написаний на Ruby читається як звичайний текст.

Основні властивості мови:

- Мова надвисокого рівня;
- Об'єктна мова з динамічною типізацією;
- Універсальна (службові скрипти, веб-додатки, графічні програми з Qt-інтерфейсом).

Якщо говорити про загальні речі про багатопотоковість, то дана можливість є найкориснішою властивістю Ruby, яка дозволяє одночасне програмування двох або більше частин програми для максимального використання ЦП. Кожна частина програми називається потоком. Іншими словами, потоки — це легкі процеси в процесі. Звичайна програма містить один потік, і всі оператори або інструкції виконуються послідовно. Але багатопотокова програма містить більше одного потоку, і в кожному потоці код виконуються послідовно, але сам потік виконується одночасно на багатоядерному процесорі. Дана можливість зменшує використання пам’яті порівняно з одним потоком завдяки виконанню кількох завдань. До Ruby версії 1.9 потоки перемикалися в інтерпретаторі, які називаються Green Threads. Але починаючи з Ruby 1.9 і далі, потоки виконуються операційною системою [2].

1.2. Фреймворк Ruby on Rails

Ruby on Rails — об'єктно-орієнтований фреймворк для створення веб-застосунків, написаний на мові програмування Ruby. Маючи простий в побудові початкового проекту патерн, такий як MVC (Model View Controller), початкова архітектура є проста та зрозуміла навіть для людини, яка тільки почала знайомитися з Ruby загалом. Дана технологія має широкий спектр розширення «тіла» проекту вміщуючи команди для створення різних сутностей, такі як: scaffold, контролери, моделі, фронтенд частини (англ. – partial), міграцій та інших.

Фреймворк - це набір компонентів, які допомагають розробляти веб-сайти швидко і просто. Кожен раз при розробці веб-сайтів потрібні схожі компоненти: спосіб аутентифікація користувачів, панель управління сайтом, форми, інструменти для завантаження файлів і т.д. [3].

Ruby on Rails [4] - це програмне забезпечення з відкритим вихідним кодом, тому не тільки воно є безкоштовним у користуванні, але і допоможе зробити його кращим. Ruby on Rails надає каркас модель-вид-контролер або коротко кажучи MVC (Model View Controller) для веб-застосунків, а також забезпечує їхню інтеграцію з веб-сервером і сервером бази даних.

Ruby on Rails - мабуть, найвідоміший і найпопулярніший фреймворк для веб-розробки з використанням Ruby. Він поставляється з десятками прекрасно зібраних вбудованих модулів та які бездоганно взаємодіють один з одним.

Один з головних переваг Ruby on Rails – швидке створення проектів, виконуючи спеціальні команди для генерації моделей, контролерів, користувацького інтерфейсу чи взагалі цілого скелету веб-сторінки. Варто зазначити, що Ruby on Rails є фреймворк з відкритим вихідним кодом. Цікаво те, що понад 5000 людей вже внесли (законтребютили) код у Rails і це число зростає. Ruby Community зростає через те, що дані розробники працюють над кампанією просування даної технології, створюючи сайти з щотижневими новинами у світі Ruby та Ruby on Rails [4].

Якщо говорити про історію даного фреймворка, то коріння Ruby on Rails лежить в додатку Basecamp (http://www.basecamphq.com) – рішенні для управління проектами, створеного данським веб-розробником для студії дизайну 37signals (http://www.37signals.com/). Багато в чому завдяки успіху Basecamp 37signals перейшла від створення веб-дизайну до розробки додатків, а Heinemeier Hansson (псевдонім – DHH) став співвласником фірми [5].

Той факт, що Rails витягували з Basecamp, розцінюється Rails-товариством як сильний бік фреймворка. З моменту виходу, Rails вже вирішував реальні проблеми. Rails створювався не в ізоляції, так що його успіх був результатом сприйняття фреймворка розробниками, створення додатків за його допомогою і пошук та подолання його недоліків. Rails довів, що є корисним, зрозумілим і повним фреймворком [5].

Heinemeier Hansson почав розробку Rails і до цього часу керує нею, але фреймворк багато виграв від відкритості коду. З часом, Rails-розробники внесли тисячі розширень і виправлень до репозиторія коду Rails. Репозиторій знаходиться під контролем команди Rails, в яку входять близько 12 висококваліфікованих професійних розробників, вибраних з числа учасників, під керівництвом Heinemeier Hansson [5].

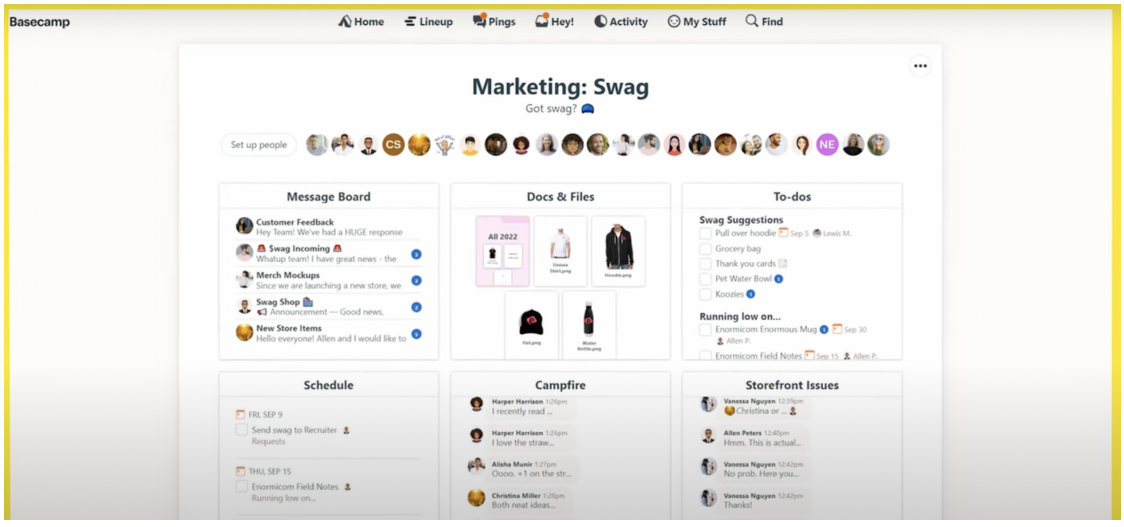

Рис.1 Приклад сайту для керування команди Basecamp

1.3. Бібліотека Bootstrap

Bootstrap є хорошим інструментом для фронтенд частини проекту, який дає розробнику створити простий, приємний та логічний користувацький інтерфейс використовуючи тільки офіційний сайт Bootstrap, як інструкцію з прикладами, та знання CSS і HTML5, що дає хорошу економію часу завдяки уникненню використання сил дизайнера (UI/UX розробника). Варто розуміти, що якщо певний проект виходить на рівень продакшену та сайтом користуються пару сот тисяч користувачів, то потрібно враховувати тренди дизайну на сьогодні та також використовувати власні стилі окрім бібліотеки Bootstrap.

Bootstrap — це безкоштовна бібліотека CSS із відкритим вихідним кодом, спрямована на адаптивну інтерфейсну веб-розробку, яка ще орієнтована на мобільні пристрої. Вона містить шаблони дизайну на основі HTML, CSS і (опціонально) JavaScript шаблони для облицювання, форм, кнопок, навігації та інших компонентів інтерфейсу.

Станом на грудень 2022 року Bootstrap є 14-м проектом із найбільшою кількістю зірочок (4-ю бібліотекою за кількістю зірочок) на GitHub з понад 161 000 зірочок.

Бібліотека Bootstrap [6] — це бібліотека HTML, CSS і JS, яка спрямована на спрощення розробки інформативних веб-сторінок (на відміну від веб-додатків). Основна мета для веб-застосунку є застосування вибору кольору, розміру, шрифту та макету Bootstrap до цього проекту. Таким чином, головним фактором є те, чи відповідальні розробники вважають цей вибір своїм смаком, бо вони (розробники) мають розуміти, що дана бібліотека вміщує нестандартні підходи щодо дизайну, тобто для розуміння бібліотеки потрібно час. Після додавання до проекту Bootstrap надає базові визначення стилю для всіх елементів HTML. Результатом є уніфікований вигляд таблиць і елементів форм у веб-додатках. Крім того, розробники можуть скористатися перевагами класів CSS, визначених у Bootstrap, для подальшого налаштування зовнішнього вигляду свого вмісту. Наприклад, Bootstrap підготував таблиці

світлого та темного кольорів, заголовки сторінок, більш помітні цитати та текст із виділенням.

Дана бібліотека також постачається з кількома компонентами JavaScript, які не потребують інших бібліотек, таких як jQuery. Вони надають додаткові елементи для інтерфейсу користувача, такі як діалогові вікна, спливаючі підказки, індикатори виконання, навігаційні відкриваючі меню та так звані «каруселі». Кожен компонент Bootstrap складається зі структури HTML, CSS і в деяких випадках супровідного коду JavaScript. Вони також розширюють функціональність деяких існуючих елементів інтерфейсу, включаючи, наприклад, функцію автозаповнення для полів введення.

Найпомітнішими компонентами даного помічника є його компоненти макета, оскільки вони впливають на всю веб-сторінку. Основний компонент макета називається «Контейнер», оскільки всі інші елементи сторінки розміщуються в ньому. Розробники можуть вибирати між контейнером фіксованої ширини так і не фіксованої. Хоча останній завжди заповнює всю ширину веб-сторінки, то перший використовує одну з п’яти попередньо визначених фіксованих ширин залежно від розміру екрана, на якому відображається сторінка [6]:

- Менше 576 пікселів;
- 576–768 пікселів;
- 768–992 пікселів;
- 992–1200 пікселів;
- Більше 1200 пікселів.

Також потрібно розуміти, що дана бібліотека вміщує кілька підтримуючих версій, однак в даній магістерській було впроваджено більш новішу версію: Bootstrap 4 (випущено 2018). Також є Bootstrap 5, який був випущений у 2021 році.

Bootstrap 5 — найновіша версія Bootstrap; з новими компонентами, швидшими таблицями стилів, більшою швидкістю реагування тощо. Він

підтримує найновіші стабільні випуски всіх основних браузерів і платформ. Однак Internet Explorer 11 і старіші версії не підтримуються.

Основна відмінність між Bootstrap 5 і Bootstrap 3 & 4 полягає в тому, що Bootstrap 5 перейшов на JavaScript замість jQuery.

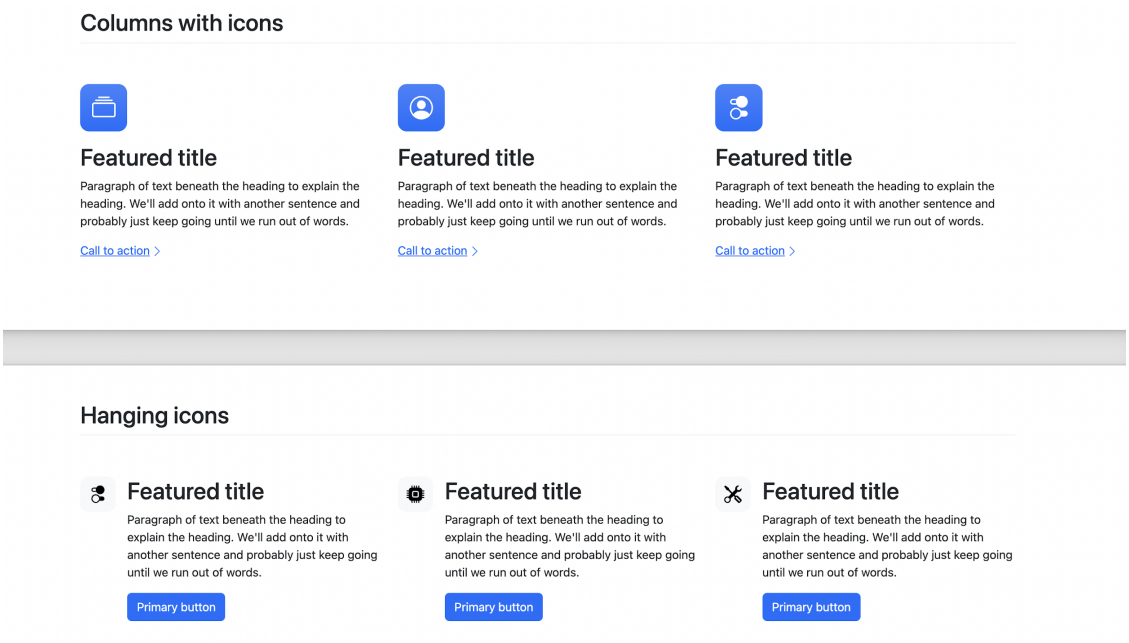

Рис. 2 Приклад дизайну веб-сторінки використовуючи Bootstrap бібліотеку

1.4. Фронтенд технології

HTML [7] – основна мова тегів Всесвітньої павутини. Більшість веб-сторінок, розміщених в Інтернеті, написані будь-якою з варіацій HTML. Починаючи від елементів, які встановлюють зв'язок з користувацьким документом (гіпертекстом), до елементів які роблять ці документи інтерактивними (наприклад форми) - все це є складовими частинами HTML. Варто розуміти, що дана «мова» є першочерговою для розуміння в написанні фронтенду як основи. Варто зазначити, що HTML не був єдиним стандартом веб-розробки.

З плином часу в HTML було багато оновлень і в даний час його новою версією є HTML5. HTML5 звичайно перш за все є мовою розмітки, але вона набула багато функцій на відміну від HTML і усунула деякі з суворих обмежень, які були присутні в XHTML. Хоча HTML5 оновлюється практично щодня - однак нових випущених пронумерованих випусків немає.

CSS - це формальна мова та основа для опису оформлення зовнішнього вигляду певного тега чи групи тегів, створеного з використанням мови розмітки (HTML). Назва походить від англійського Cascading Style Sheets, що означає «каскадні таблиці стилів» [8].

Призначенням CSS є відокремлення налаштування зовнішнього вигляду сторінки від її змісту. Якщо документ створено тільки з використанням HTML, то в ньому визначається не тільки кожен елемент, але і спосіб його відображення (колір, шрифт, положення блоку і т. Д.). Якщо ж підключені каскадні таблиці стилів, то HTML описує тільки черговість об'єктів. А за їх властивості відповідає CSS. В HTML досить прописувати клас, не перераховуючи всі стилі кожен раз.

Рис. 3. Приклад побудови CSS-селектора

jQuery - бібліотека JavaScript, що фокусується на взаємодії JavaScript та HTML. Дана бібліотека допомагає легко отримувати доступ до будь-якого елементу DOM, звертатися до атрибутів і вмісту елементів DOM, маніпулювати ними. Також бібліотека jQuery надає зручний API для роботи з AJAX. Зараз розробка jQuery ведеться командою jQuery на чолі з Джоном Резігом [9]. Бібліотека була представлена громадськості на комп'ютерній конференції «BarCamp» в Нью-Йорку в 2006 році [9].

Можливості бібліотеки Jquery [9]:

- рушій кросбраузерності CSS-селекторів Sizzle, що виділився в окремий проект;
- зручний рух по дереву DOM, включаючи підтримку XPath як плагіна;
- події;
- візуальні ефекти;
- AJAX-доповнення;
- JavaScript-плагіни.

Точно так само, як CSS відокремлює візуалізацію від структури HTML, JQuery відділяє поведінку від структури HTML. Такий поділ поведінки і структури також називається принципом ненав'язливого JavaScript [9].

Бібліотека jQuery містить функціональність, яка корисна для максимально широкого кола завдань. Тим не менш, розробниками бібліотеки не ставилося завдання суміщення в jQuery функцій, які підійшли б усюди, оскільки це призвело б до великого коду, більша частина якого б не використовувалась. Тому була реалізована архітектура компактного універсального ядра бібліотеки та плагінів [9].

Вся робота з jQuery ведеться за допомогою функції $. Якщо на сайті застосовуються інші JavaScript бібліотеки, де $ може використовуватися для своїх потреб, то можна використовувати її синонім – jQuery. Другий спосіб вважається більш правильним, а щоб код не виходив занадто громіздким, можна писати його наступним чином [10]:

```
jQuery (function ($) {
  // Тут код скрипта, де в $ знаходитиметься об'єкт, що дає доступ до функцій jQuery
})
```

Роботу з jQuery можна розділити на 2 типи [10]:

- Отримання jQuery-об'єкта за допомогою функції $ (). Наприклад, передавши в неї CSS-селектор, можна отримати jQuery-об'єкт всіх елементів HTML, що потрапляють під критерій і далі працювати з ними за допомогою різних методів jQuery-об'єкта. У випадку, якщо метод не повинен повертати жодного значення, він повертає посилання на jQuery об'єкт, що дозволяє вести ланцюжок викликів методів згідно концепції текучого інтерфейсу;
- Виклик глобальних методів в об'єкт $, наприклад, зручних ітераторів для руху по масиву.

Типовий приклад маніпуляції відразу декількома вузлами DOM полягає у виклику $ функції з рядком селектора CSS, що повертає об'єкт jQuery та містить деяку кількість елементів HTML-сторінки. Ці елементи потім обробляються методами jQuery [10]. Наприклад:

```
$ ("div.test"). add ("p.quote"). addClass ("blue"). slideDown ("slow");
```

знаходить всі елементи div з класом test, а також всі елементи p з класом quote, потім додає їм усім клас blue і візуально плавно спускає вниз. Тут методи add, addClass і slideDown повертають посилання на вихідний об'єкт $ ("div.test"), тому можливо вести такий ланцюжок.

Методи, що починаються з $., зручно застосовувати для обробки глобальних об'єктів [10]. Наприклад:

```
$ .each ([1,2,3], function () {
  document.write (this + 1);
});
```

додасть на сторінку 234.

$ .ajax і відповідні функції дозволяють використовувати методи AJAX. наприклад:

```
$ .ajax ({
   type: "POST",
   url: "some.php",
   data: {name: 'John', location: 'Boston'},
   success: function (msg) {
     alert ("Data Saved:" + msg);
   }
});
```

Приклад додавання до елементу обробника події click за допомогою jQuery[10]:

```
$ ("a"). click (function () { alert ("Hello world!"); });
```

Yarn — це система упакування програмного забезпечення, розроблена у 2016 році компанією Facebook (нині Meta) для середовища виконання Node.js JavaScript. Дана система є альтернативою менеджеру npm, в свою чергу yarn була створена в результаті співпраці Facebook, Exponent (тепер Expo.dev), Google і Tilde (компанія, яка розробляє Ember.js) для вирішення проблем узгодженості, безпеки та продуктивності з великими кодовими базами [11].

Якщо коротко описати дану систему, то yarn — це менеджер пакетів, який виконує функції менеджера проекту. Незалежно від того, чи команда працює над одноразовими проектами чи великими репозиторіями, дана технологія підходить будь-кому.

Основні характеристики, якими можна описати Yarn від інших конкурентів:

- Робочі області – можна розділити свій проект на підкомпоненти, які зберігаються в одному сховищі;
- Документація - особлива увага приділяється документації Yarn, яка постійно вдосконалюється на основі розробницьких відгуків;
- Плагіни – Yarn не може вирішити всі проблеми, але він може стати основою для інших, тобто надає змогу встановлювати інші бібліотеки/залежності;
- Відкритість – Yarn являється незалежним проектом із відкритим кодом, який не пов’язаний із жодною компанією.

Також хорошим плюсом являється те, що даний менеджер зберігає в кеш-пам’яті кожен пакет, який він завантажив, тому йому більше ніколи не доведеться завантажувати той самий пакет. Він також робить майже все одночасно, щоб максимізувати використання ресурсів. Це означає ще швидше встановлення. Використовуючи детальний, але стислий формат файлу блокування та детермінований алгоритм для операцій встановлення, Yarn може гарантувати, що будь-яка інсталяція, яка працює в одній системі, точно так само працюватиме в іншій системі.

Багато проектів у Facebook, як-от React, залежать від коду в реєстрі npm. Однак під час внутрішнього масштабування деякі проекти зіткнулися з проблемами узгодженості під час встановлення залежностей між різними машинами та користувачами, кількістю часу, який знадобився для зменшення залежностей, і мали певні проблеми з безпекою щодо того, як npm менеджер версій виконує код із деяких із цих залежностей. Тому дані проекти

намагалися знайти рішення навколо цих проблем, але вони часто самі порушували нові проблеми.

Під час усього цього процесу Yarn надає суворі гарантії щодо встановлення пакетів. Розробник контролює, які сценарії життєвого циклу виконуються та для яких пакетів. Контрольні суми пакетів також зберігаються у файлі блокування (lockfile), щоб гарантувати, що ви отримуєте той самий пакет кожного разу [11].

Webpacker — це оболонка Rails навколо системи збирання webpack, яка забезпечує стандартну конфігурацію webpack і прийнятні значення за замовчуванням. Мета Webpack або будь-якої зовнішньої системи збірки полягає в тому, щоб дозволити написати зовнішній код у спосіб, який зручний для розробників, а потім упакувати цей код у спосіб, який зручний для браузерів. За допомогою webpack можна керувати JavaScript, CSS і статичними ресурсами, такими як зображення або шрифти. Webpack дозволить написати свій код, посилатися на інший код у програмі, трансформувати свій код і об’єднати код у пакети, які легко завантажити.

Варто зазначити, що версія Node повинна бути вища 10.17.0, оскільки команда webpacker:install автоматично викликається командою rails new.

Після виконання команди встановлення створюються такі файли [12]:

- config/webpacker.yml — основний файл конфігурації, який містить конфігурацію за замовчуванням і конфігурації для певних середовищ;
- config/webpacker/ — каталог, де створюються файли конфігурації JavaScript для певних середовищ;
- bin/webpack — виконуваний файл, який викликає webpack для створення пакетів;
- bin/webpack-dev-server — виконуваний файл, який запускає сервер розробки, який перезавантажує webpack кожного разу, коли ви вносите зміни у файли JavaScript, включені в комплект.

Основний файл конфігурації для Webpacker розташований у каталозі config і називається webpacker.yml. На відміну від Webpacker, конфігурація для Webpack зберігається окремо для кожного середовища в каталозі config/webpack.

Якщо потрібно отримати доступ до конфігурації з консолі rails або на рівні коду, можна викликати Webpacker.manifest.config, який поверне екземпляр класу конфігурації [12].

1.5. Бібліотеки Ruby on Rails фреймворка (Gems)

В даний час Ruby on Rails (далі RoR) спільнота (RoR Community) налічує не одну тисячу розробників, які працюють над покращенням коду та ідей та, розуміючи, що всі розробки та ідеї не додати у фремворк - створили нову платформу RubyGems, яка містить бібліотеки, відомі як gems. Кожна з даних бібліотек має чітку ідею та реалізацію, яку вона вирішує та дає можливість розробнику створити певну реалізацію, не повторюючи те саме, що вже створене попередником gem'у. Вищезгаданий сайт вміщує всі бібліотеки (gems), які створені для Ruby розробників.

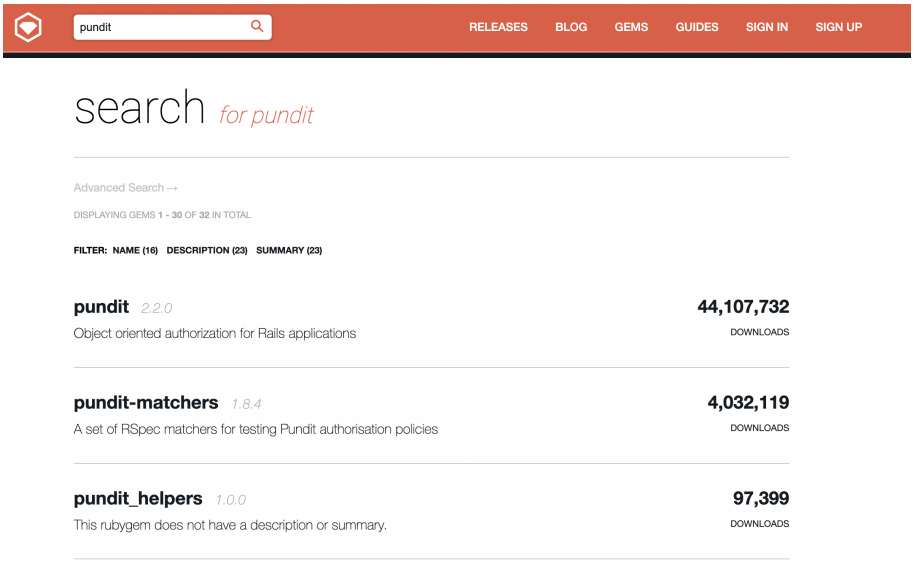

Рис. 4: Приклад пошуку gem'у на сайті RubyGems

Щодо Ruby бібліотек gems, які було використано в даній магістерській.

Pg - це бібліотека (інтерфейс) Ruby для СУБД PostgreSQL [13], яка працює з PostgreSQL 9.3 і пізніших версій. Вона за бажанням може вводити значення результатів та параметри запиту в коді Ruby, що може пришвидшити передачу даних до та з бази даних, оскільки виділення рядків зменшено.

Варто зазначити, що для зручної роботи з вже використаним Pg gem'ом, використовується pgAdmin — інструмент керування для PostgreSQL і похідних реляційних баз даних, таких як EnterpriseDB EDB Advanced Server. Його можна запускати як веб-програму, так і як настільну програму.

Byebug — це бібліотека (налагоджувач, debugger) Ruby. Вона реалізована за допомогою TracePoint C API для керування виконанням і Debug Inspector C API для навігації стеком викликів [14]. Основний компонент забезпечує підтримку, на якій можуть спиратися інтерфейси. Серед іншого забезпечується обробка точок зупинок та прив'язки для фреймів стека, а також простота у використанні інтерфейсу командного рядка.

Debugger дає можливість зрозуміти, що відбувається всередині програми Ruby під час її виконання та пропонує багато традиційних функцій налагодження, таких як:

- Поетапність: виконання вашої програми по одному рядку;
- Breaking: призупинення програми на певній події чи певній інструкції для перевірки поточного стану;
- Оцінка: базова функціональність REPL, хоча pry справляється з цим краще;
- Відстеження: відстеження різних значень ваших змінних або різних рядків, які виконує ваша програма.

Haml-rails – це бібліотека (перетворювач файлів) на розширення .html.haml, що надає генератори Haml для Rails 5, 6 і 7. Вона також дозволяє використовувати Haml як механізм створення шаблонів - тому не потрібно щось придумувати у власному application.rb, коли у Gemfile вже чітко вказано, який механізм створення шаблонів встановлено. Це гарантує, що [15]:

- Кожного разу, коли створюється ресурс, представлення чи поштова програма (mailer), буде отримане шаблони Haml (замість ERB);
- Коли програма Rails завантажується, Haml буде завантажено та ініціалізовано автоматично;
- Шаблони Haml будуть враховуватися дайджестом кешу шаблонів перегляду.

Simple Form – це бібліотека, яка прагне бути максимально гнучкою, допомагаючи створювати форми за допомогою потужних компонентів. Основна мета Simple Form — не торкатися способу визначення макета,

дозволяючи знайти кращий дизайн чи формат форми. Більшу частину DSL було успадковано від Formtastic, тому даний gem не є виключенням.

В Rails 5 дана бібліотека використовується за допомогою методу form_tag, HTML-форми або form_for, але ресурсів для Simple Form не так багато, навіть незважаючи на те, що вона стала золотим стандартом форм Rails. Крім того, form_for та form_tag вважаються застарілими після Rails 5.2 і замінені на form_with [16].

Faker [17] - бібліотека для створення підроблених даних, таких як імена, адреси та номери телефонів. Якщо більш теоретично, то даний gem — це порт бібліотеки Perl Data::Faker. Це бібліотека для створення фальшивих даних та допомагає генерувати реалістичні тестові дані та заповнювати базу даних декількома записами під час розробки.

Devise — це бібліотека, яка надає гнучке рішення автентифікації для Rails на основі Warden - Rack фреймворка для автентифікації (General Rack Authentication Framework) [18].

Основні плюси даної бібліотеки:

- Rack основа;
- Повне рішення MVC, засноване на двигунах Rails;
- Дозволяє одночасно ввійти в обліковий запис кількох моделям;
- Базується на концепції модульності, яка звучить “використовуйте лише те, що вам дійсно потрібно”.

FriendlyId — це бібліотека («швейцарський армійський ніж» плагінів) для постійних посилань Active Record, що дає змогу створювати гарні URL-адреси та працювати із зручними для людини рядками, наче це цифрові ідентифікатори [19]. За допомогою FriendlyId легко змусити програму використовувати такі URL-адреси:

«https://example.com/states/washington» замість:

«https://example.com/states/4323454».

Rolify - бібліотека без будь-яких примусових авторизацій, що підтримує область для об’єкта ресурсу. Бібліотека керування Rolify потрібна для

визначення ролей певного проекту. Після підключення даного gem'у до магістерської, виконалася команда «rails g rolify Role User», яка створила автоматично таблицю з Roles та створила міграцію для зв'язку з таблицею Users [20].

Pundit [21] – бібліотека, яка надає набір помічників, які допомагають використовувати звичайні класи Ruby та об'єктно-орієнтовані шаблони проектування для створення надійної та масштабованої системи авторизації користувачів в Ruby on Rails проекту.

Вищезгадана бібліотека зосереджується навколо поняття policy класів, що дає розмістити ці класи в «app/policies» директорії. Хорошим інтуїтивним прикладом можна вважати, що є потреба оновити публікацію, якщо користувач є адміністратором, або якщо публікація неопублікована і потрібно для певного користувача її відобразити.

Pagy робить розбиття на сторінки настільки простим, наскільки це можливо: немає «декларативного DSL», якого потрібно вивчати, немає необхідності в будь-якому спеціальному модулі чи адаптері [22]. Немає вкладених модулів, класів, незліченних методів і багатьох сотень рядків коду.

Aws-sdk gem для Ruby є хорошим рішенням, яке дає можливість створити з'єднання з Amazon AWS для використання можливостей даної платформи, такі як: відображення проекту в глобальному доступі, збереження фотографій ззовні, створення БД ззовні. Дана бібліотека доступна у RubyGems.

Wicked PDF [23] – gem, який використовує утиліту оболонки «wkhtmltopdf» для надання PDF-файлу користувачеві в HTML. Іншими словами, замість того, щоб мати справу з якоюсь DSL генерацією PDF, просто пишеться вигляд HTML, як зазвичай, а потім дозволяється Wicked PDF генерувати pdf файл.

1.6. Платформа Heroku

Platform as a Service (PaaS, «платформа як послуга») — модель надання хмарних обчислень, при якій споживач отримує доступ до використання інформаційно-технологічних платформ: операційних систем, систем управління базами даних, необхідному програмному забезпеченню, засобам розробки та тестування, які розміщені у хмарного провайдера. У цій моделі вся інформаційно-технологічна інфраструктура, включаючи обчислювальні мережі, сервери, системи зберігання, цілком управляється провайдером, провайдером же визначається набір доступних для споживачів видів платформ та набір керованих параметрів платформ, а споживачеві надається можливість використовувати платформи, створювати їх віртуальні примірники, встановлювати, розробляти, тестувати, експлуатувати на них прикладне програмне забезпечення, при цьому динамічно змінюючи кількість споживаних обчислювальних ресурсів [24].

Heroku — це одна з перших хмарних (PaaS) платформ, яка з'явилася в червні 2007 року і спочатку підтримувала тільки мову програмування Ruby, але на даний момент список підтримуваних мов включає в себе Java, Node.js, Scala, Clojure, Python і PHP (неофіційно). На серверах Heroku використовуються операційні системи Debian або Ubuntu [24].

Heroku також надає підтримку таких систем управління базами даних, як Cloudant, Membase, MongoDB і Redis, крім основної – PostgreSQL [24].

Додатки, що працюють на Heroku, використовують також DNS-сервер Heroku (зазвичай додатки мають доменне ім'я виду «імя_додатку.herokuapp.com»). Для кожної програми виділяється декілька незалежних віртуальних процесів, які називаються «dynos». Вони розподілені за спеціальною віртуальною сіткою («dynos grid»), яка складається з декількох серверів [24]. Heroku також має систему контролю версій Git .

Платформа Heroku для розробника програмного забезпечення з метою взаємодії локальної та віддаленої системи надає певний інтерфейс після встановлення на локальну систему додаткової бібліотеки [24].

Назва команди	Призначення
account	Керування опціями запису heroku
certs	Керування кінцевими точками ssl для додатку
drains	Виведення системного log каталогу для додатків
db	Керування базою даних додатку
keys	Керування ключами аутентифікації для додатку
pg	Керування heroku-postgersql базою даних
plugins	Керування плагінами до gem heroku
ssl	Керування ssl сертифікатами додатку
status	Перевірка статусу платформи
pgbackup	Керування резервним копіюванням бази даних

§2. Програмна частина

2.1. Опис предметної області

Предметною областю називається частина системи, що становить зацікавленість даного дослідження. При проектуванні програми предметна область відображається моделями даних декількох рівнів. Кількість рівнів залежить від рівня складності розв'язуваних завдань, але концептуальний і логічний рівні є важливими.

В магістерській роботі предметною областю є створення онлайн-платформи для курсів різних спеціальностей, які можна вивчати студенту, так і представляти викладачу. За управління високорівневими даними відповідатиме адміністратор. За даними курсів, які опублікував викладач відповідає він же. Бізнес-процеси проекту:

1. реєстрація користувачів;
2. перегляд курсів;
3. перегляд даних користувачів;
4. керування курсами та секціями курсу;
5. керування ролями;
6. пошук курсів по певним особливостям;
7. опублікування даних;
8. перегляд активності певного користувача;
9. перегляд новин по трендам на веб-сайті.

На рисунку 5 відображено діаграма функцій процесу реєстрації користувача:

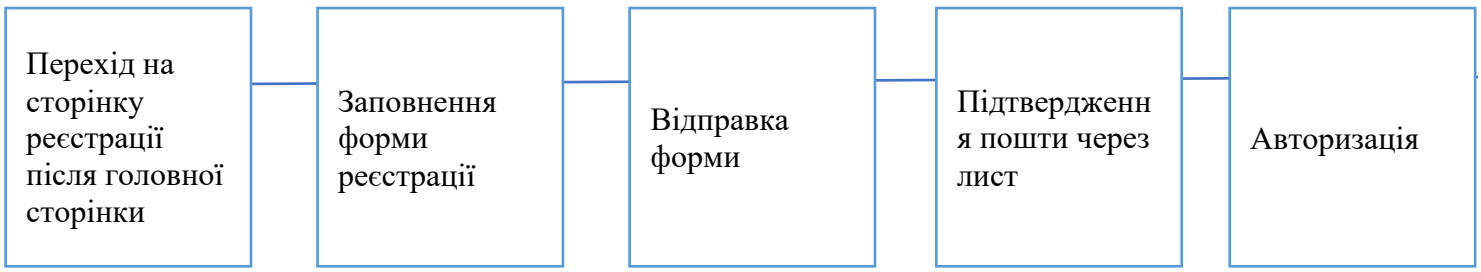

Рис. 5

Щоб користувач (студент) міг переглядати курси та інформацію, яку містять вони, він повинен ідентифікуватися у системі. Це відбувається після того, як користувач зареєструвався. Процес реєстрації зображено на рис.5. Після авторизації користувач має право шукати потрібні курси, проходити їх.

Викладач має можливість створювати курси, наповнювати їх інформацією, редагувати та взагалі видаляти з системи, якщо він вважатиме це за потрібне. Також можливість переглядати всі курси, які є на онлайн-платформі.

Адміністратор, має можливість переглядати інформацію, яка є на сайті. Він керує ролями для користувачів. Також бачить всю інформацію про користувачів, чи вони онлайн на сайті, чи вони пройшли реєстрацію, коли заходили останній раз, з якої IP-адреси заходили та взагалі, корегувати відображення всіх курсів (видаляти, редагувати).

2.2. Структура проекту

Проект складається із 2 частин: так званої архітектури MVC та другорядних директорій. Кожна частина містить певний набір файлів та потрібних піддиректорій.

MVC частина (“app” директорія) складається з таких папок: model, view, controller, javascript, assets. Про кожну з них більш детально.

Model директорія містить 10 моделей: Courses, Lessons, Comment, Roles, UserRoles, Enrollments, UserLessons, CourseTag, Tag та Users. Остання модель використовується для авторизації (див. дод. 4).

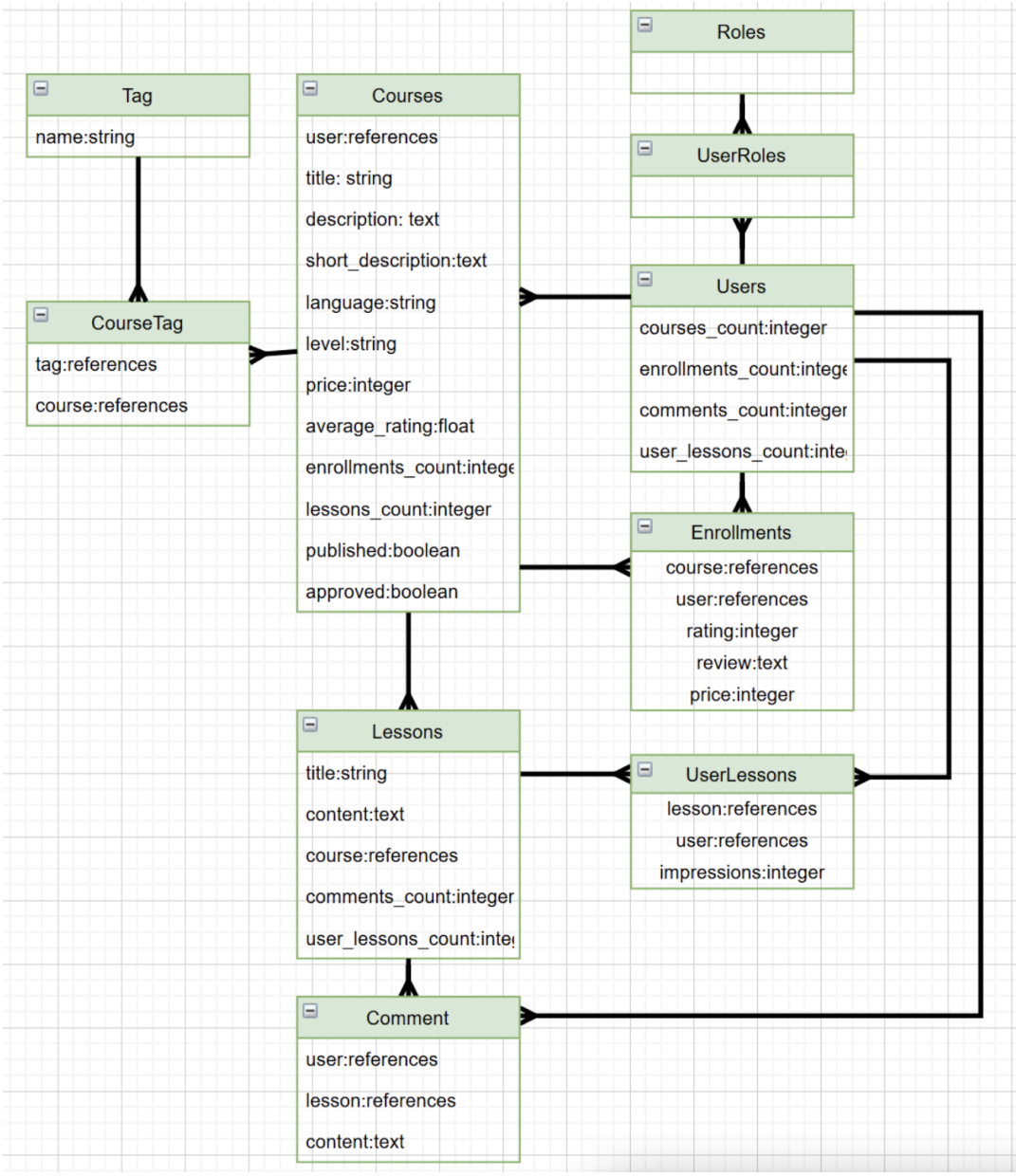

Рис. 6: Схематичне зображення БД проекту

View вміщує такі директорії: courses, landing_page, layouts, lessons так users, а вони в свою чергу в кожній папці вміщують такі файли: _user.html.haml, edit.html.haml, index.html.haml, show.html.haml, _form.html.haml, _course.html.haml, _new.html.haml (див. дод. 9).

Controller директорія складається з 5-ти контролерів: ApplicationController, CoursesController, LandingPageController, LessonsController та UsersController, що містяться у файлах: application_controller.rb, courses_controller.rb, landing_page_controller.rb, lessons_controller.rb та users_controller.rb відповідно (див. дод. 7).

Друга частина містить директорії, які потрібні для управління, підключення, підвантаження та обробки файлів проекту. Такі папки як: config, db, test та policies (див. дод. 8). Про кожну з них більш детально.

Config директорія вміщує такі з основних файлів як: database.yml (див. дод. 10), environment.rb, routes.rb, application.rb та піддерикторії environments та initializers.

Db має основні файли для потрібної роботи з БД, тобто: seeds.rb (див. дод. 11), schema.rb (див. дод. 12) та теку migrate, де зберігаються всі міграції проекту.

Test – частина юніт тестування (Unit testing) де відображені такі файли як: courses_controller_test.rb, lessons_controller_test.rb, users_controller_test.rb, landing_page_controller_test.rb та інші (див. дод. 13).

Policies директорія вміщує такі файли: application_policy.rb, course_policy.rb, lesson_policy.rb, user_policy.rb, що вміщують такі класи як: ApplicationPolicy, CoursePolicy, LessonPolicy, UserPolicy відповідно (див. дод.8).

2.3. Аналіз схожих веб-сайтів

Для аналізу інтерфейсу та функціональності було обрано наступні веб-сайти:

- «Udemy», режим доступу: (https://www.udemy.com/);
- «Moodle», режим доступу: (https://moodle.chnu.edu.ua/);
- «Prometheus», режим доступу: (https://prometheus.org.ua/).

«Udemy» надає висококваліфіковані курси для всіх типів галузей та приємним користувацьким інтерфейсом (див. рис.7).

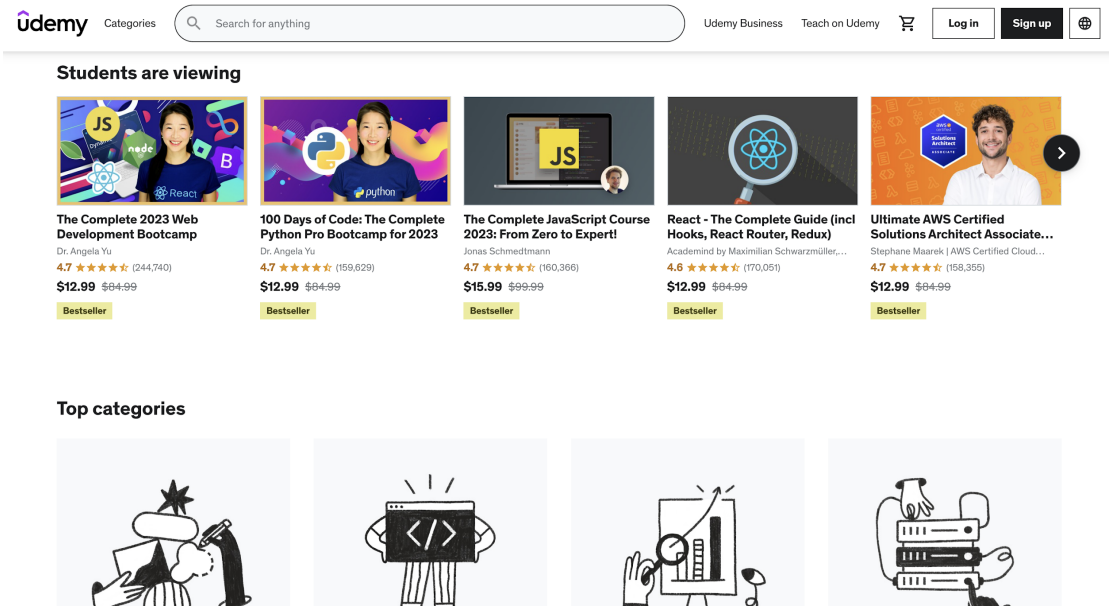

Рис.7

Мінусами є те, що платформа вміщує багато курсів без українського перекладу та весь якісний контент є платний, навіть, якщо переглядає студент.

«Moodle» є міжнародна онлайн система з інформацією про курси різних спеціальностей, яка відома всьому студентському середовищу. (див. рис.8).

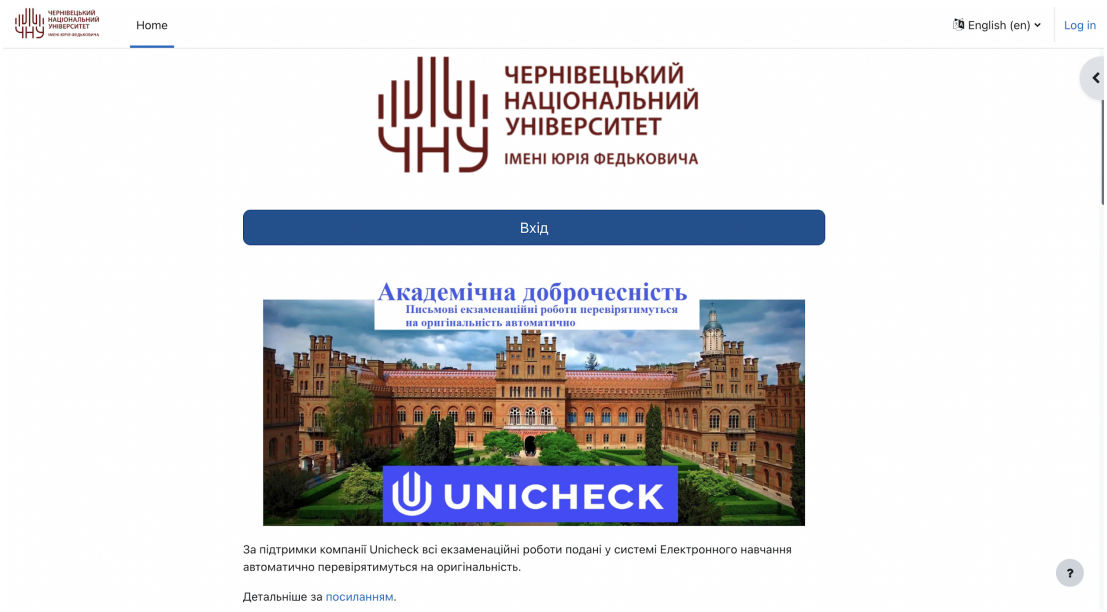

Рис.8

Мінусом даного сайту є те, що сайт є важкий в користуванні на перших порах для студентів.

«Prometheus» – українська платформа для всіх галузей та по різним тематиках, яка вміщує велику кількість безкоштовних українських курсів.

Мінусом даного веб-сайту: не вміщує окремого середовища для кожного університету, викладачам важко зрозуміти успішність студентів з їхнього факультету.

Їхній веб-сайт виглядає наступним чином (див. рис.9).

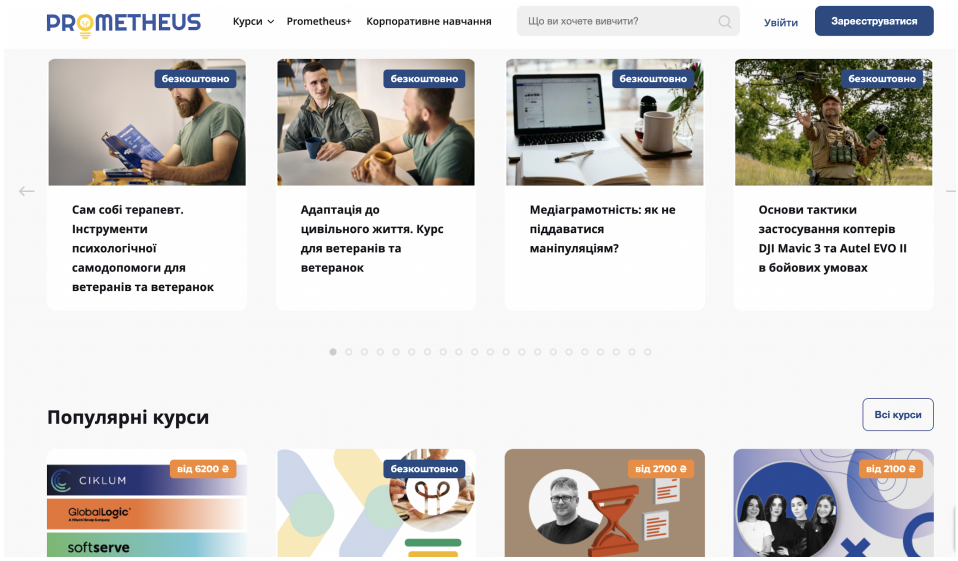

Рис.9

2.4. Інструкція для користувачів

В даному розділі детально розписаний функціонал та можливості студента та викладача.

При переході за посиланням користувачу відображається головна сторінка. Інтерфейс головної сторінки можна глянути на рисунках 10, 11, 12.

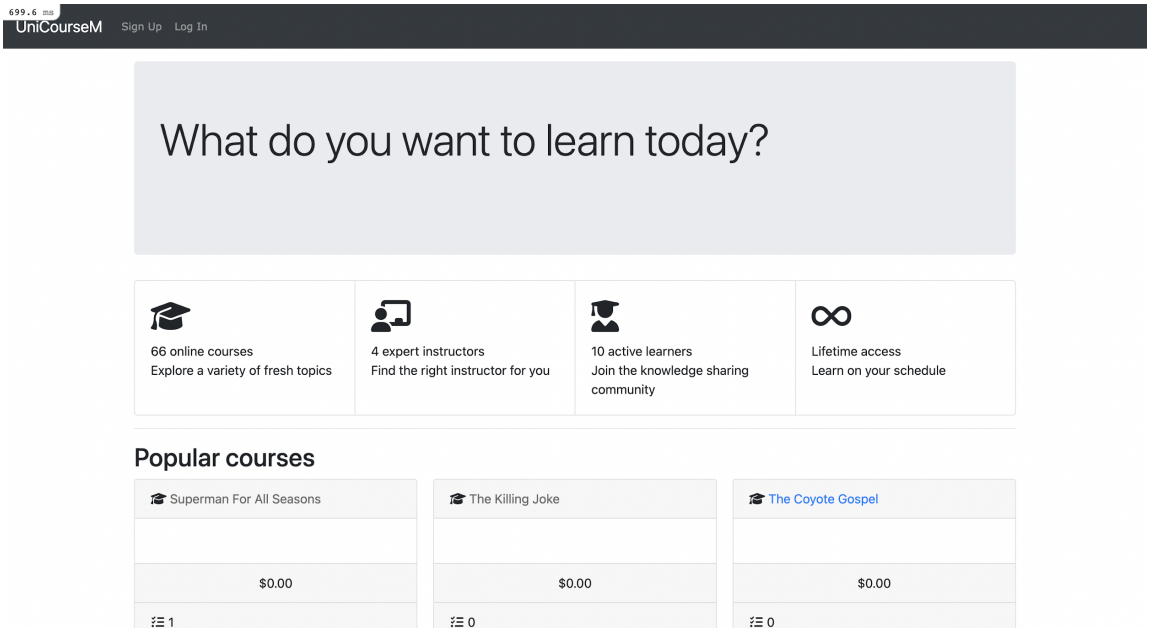

Рис.10

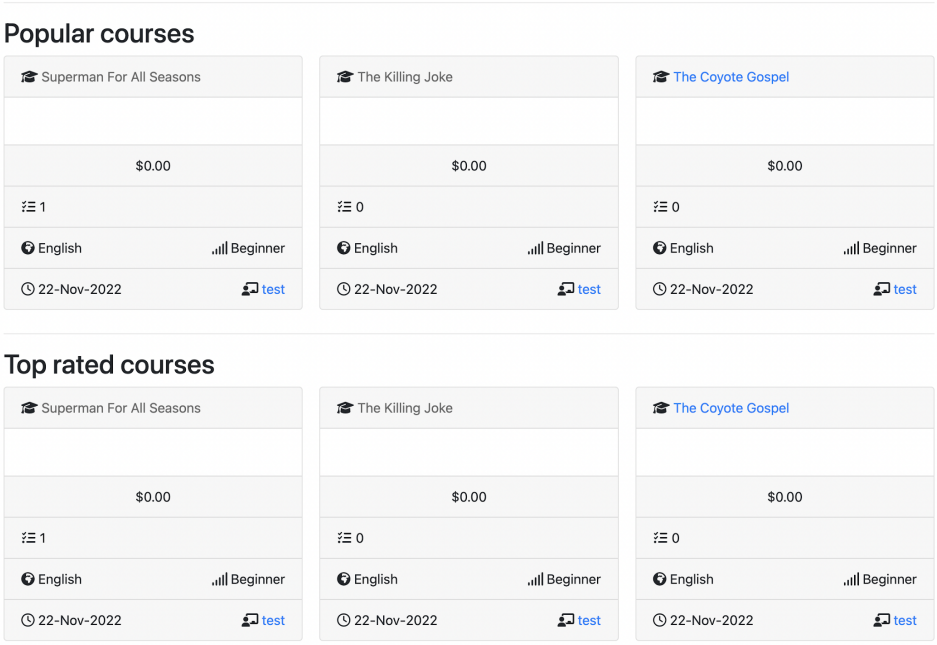

Рис.11

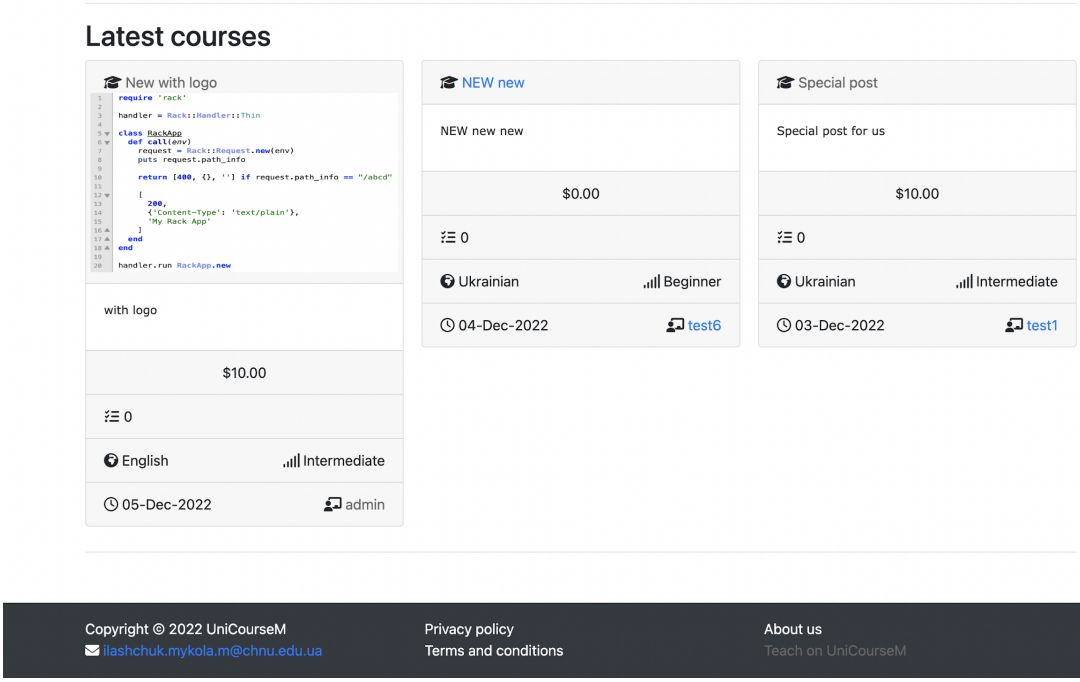

Рис.12

На головній сторінці відображається автоматичне визначення найпопулярніших курсів на сайті, найбажаніші курси та найновіші курси, які були додані (рис.11, рис.12). Також є загальна інформація про онлайн-платформу, а саме: кількість курсів, які можна вивчати, кількість викладачів на сайті та кількість студентів, які зареєстровані.

Якщо розглянути коротку інформацію про один курс (рис.12), то ми побачимо заголовок курсу, його основне фото/лого, корокий опис, скільки містить розділів, яка мова викладання даного курсу, рівень курсу, ціна (якщо є потреба), дата опублікування та ім’я користувача, який додав певний курс.

Будь-який користувач має можливість зареєструватися та авторизуватися на сайті. Для того, щоб користувач зміг зареєструватися, він має натиснути на кнопку «Реєстрація» (Sign Up), яка знаходиться у лівому верхньому куті меню (рис.13).

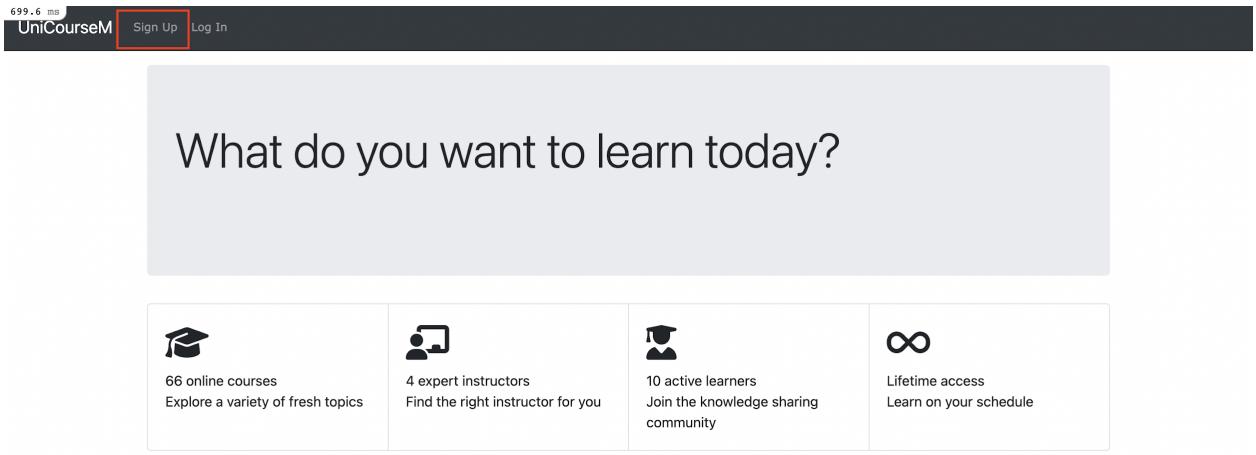

Рис.13

Після натискання кнопки «Реєстрація» з'являється форма для введення даних (рис.14) Після введення успішних даних для форми, приходить на пошту лист про підтвердження емейла, потрібно перейти за посиланням, яке є унікальним для кожного користувача при реєстрації, тому що містить зашифрований токен з інформацією про користувача.

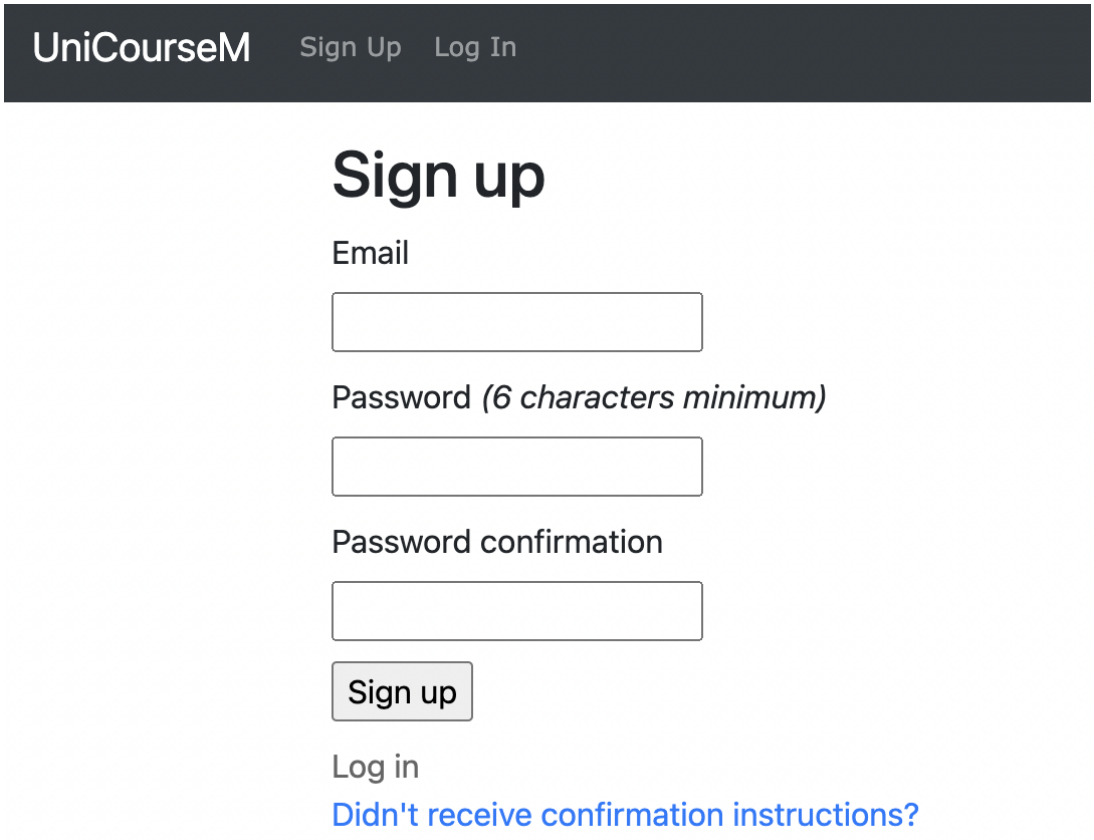

Рис.14

При введені неправильних даних форма очищується та виділяються місця, в яких було зроблено помилки (рис.15).

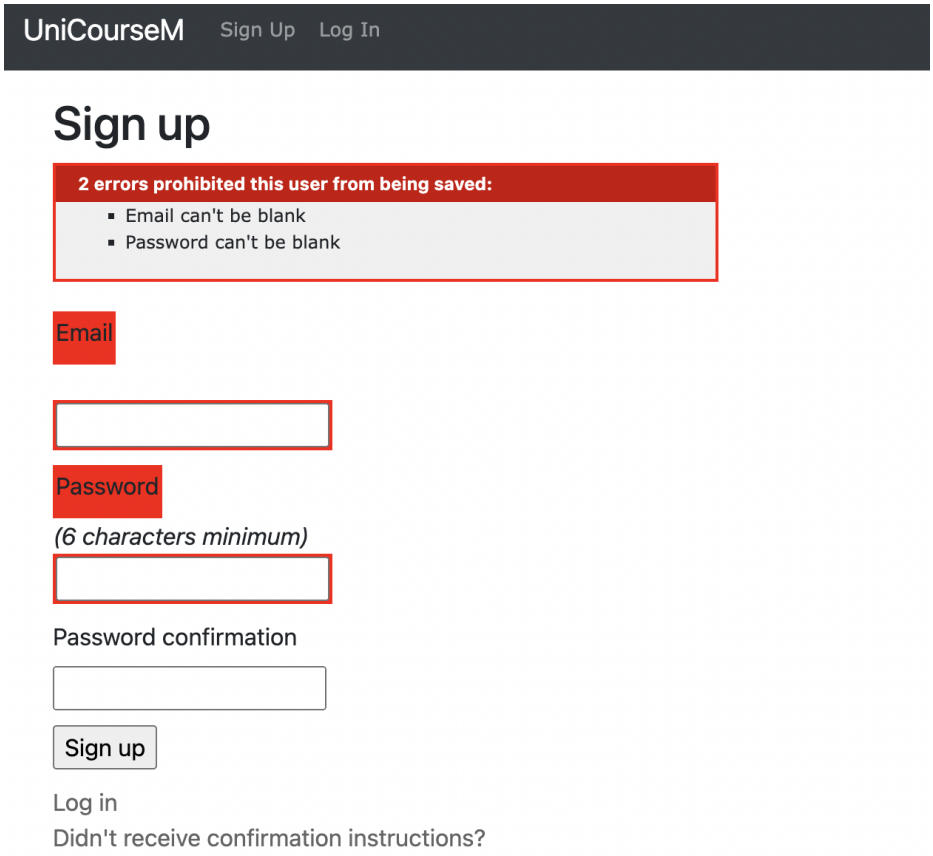

Рис.15

Якщо користувач зареєстрований, то він може авторизуватися. Це можна зробити натиснувши на кнопку «Увійти» (Log In) (рис.16).

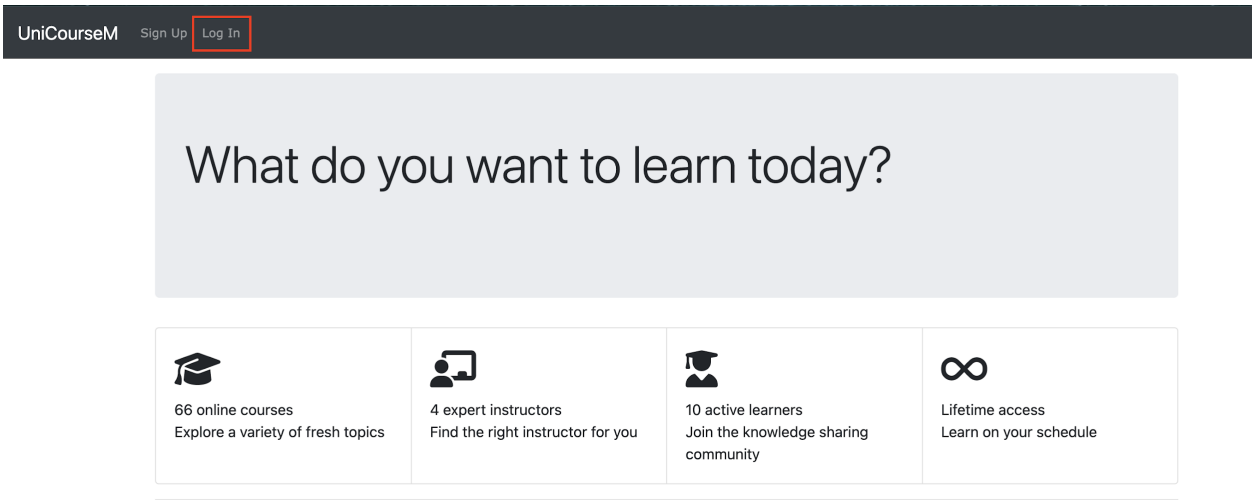

Рис.16

Після цього користувачу відображається сторінка авторизації. Користувач має внести електронну пошту та пароль вказані при реєстрації (рис.17).

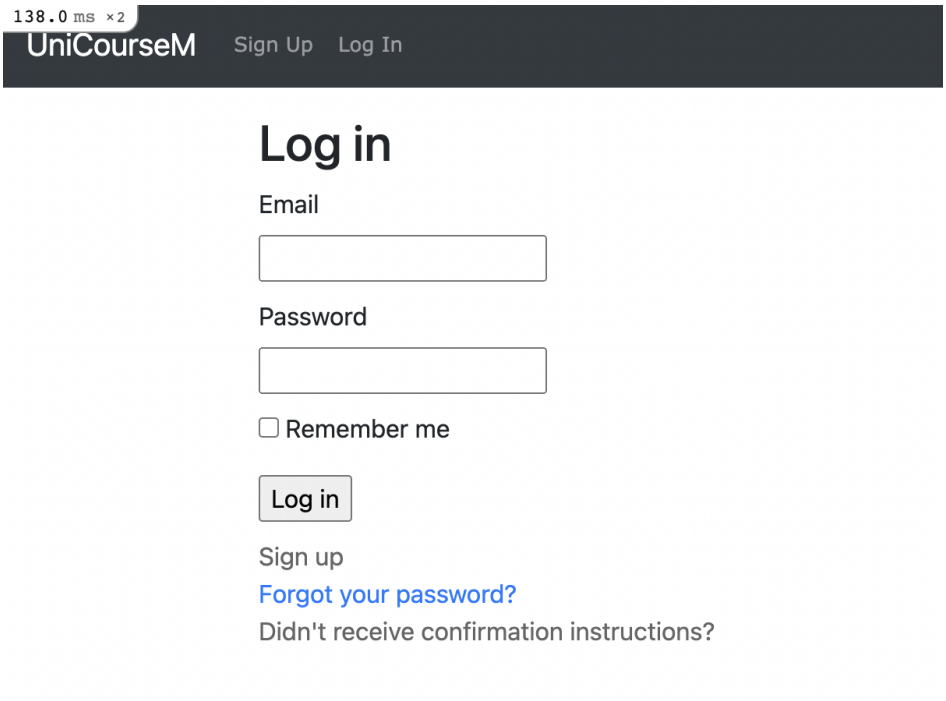

Рис.17

При введені некоректних даних користувачу відображається помилка (рис.18).

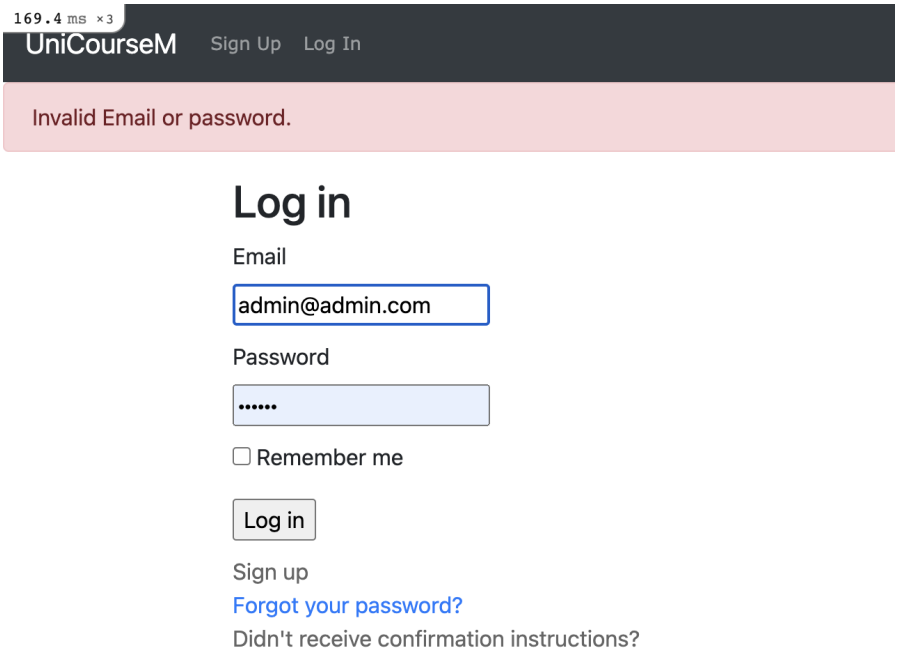

Рис.18

Після успішного входу на сайт користувачу відображається нові пункти головного меню – курси, додати свій курс, користувачі, пошук курсів на сайті, активність, ім’я користувача (username), акаунт налаштування та кнопка «Вихід» (рис.19).

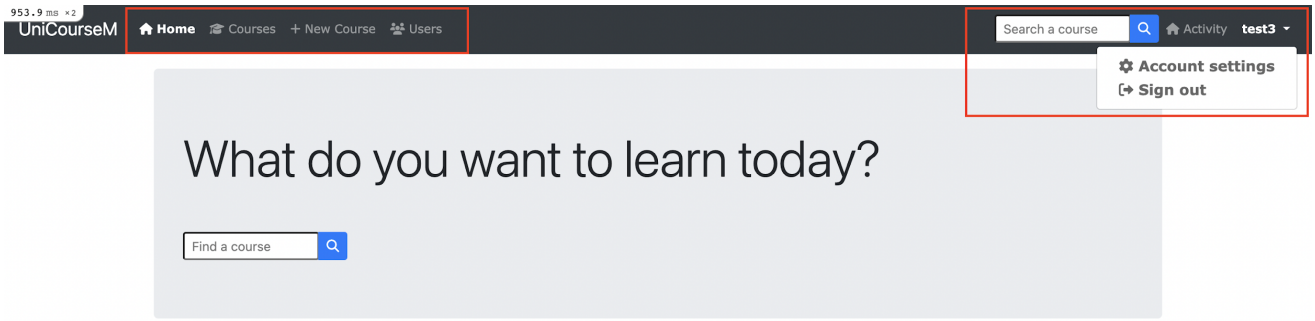

Рис.19

Якщо натиснути на пункт меню «Курси», то відкриється сторінка для перегляду всі курсів та фільтрація їх по певним критеріям: назві, опису (ключові слова), мова курсу, рівень, ціна (якщо є потреба) та username викладача (рис. 20).

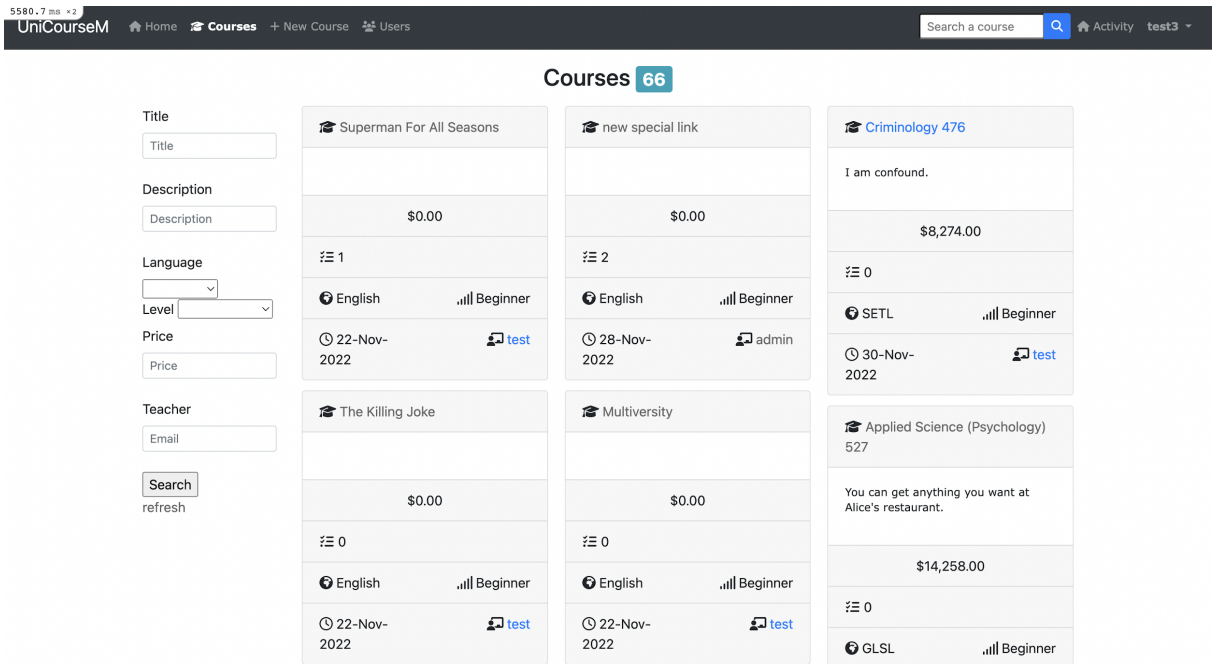

Рис.20

Натиснувши на пункт меню «Новий курс» (рис. 19) маючи роль «викладач», відкриється сторінка з формою для створення курсу (рис. 21). В

даній формі маємо такі поля: ім’я курсу, фото/лого, яке потрібно вибрати з локальної машини, опис курсу, який можливо стилізувати (жирний шрифт, підкреслення, цитата, приклад коду і т.д.), короткий опис, мова, рівень та ціна.

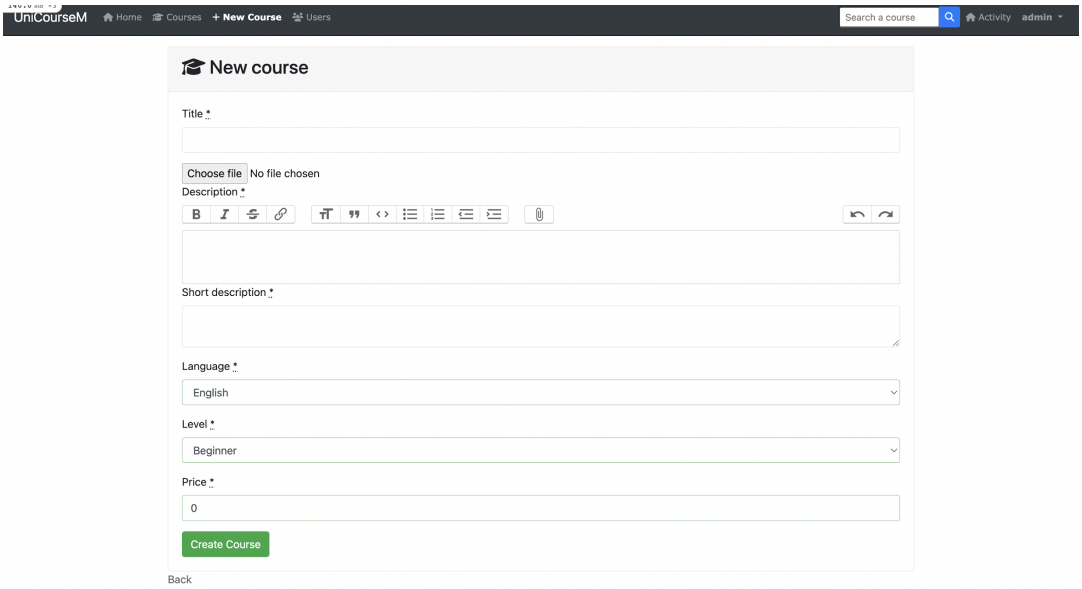

Рис. 21

Також є валідація обов’язкових полів (рис. 22), які позначені зірочкою у формі.

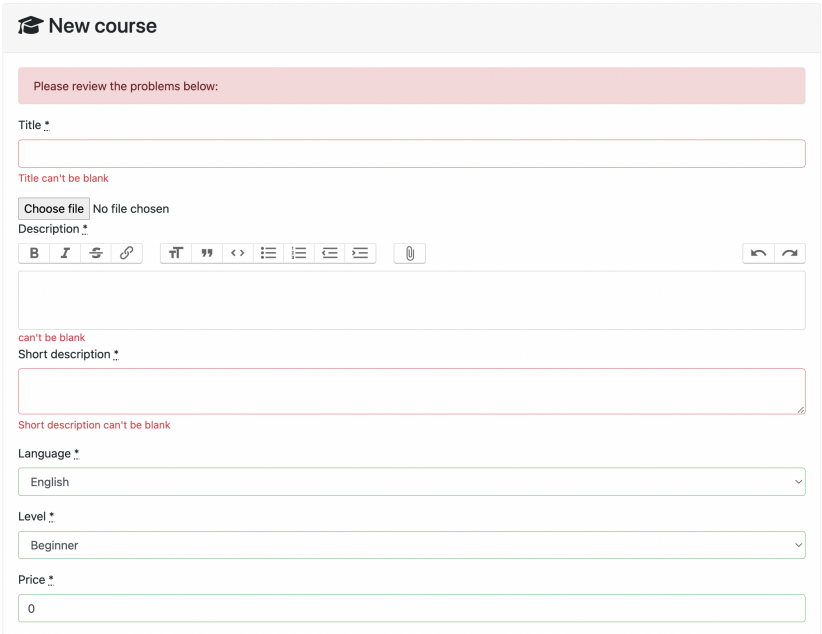

Рис. 22

Після додавання, відкривається сторінка з новоствореним курсом де можна додати секції курсу (Lessons) та переглянути всю інформацію (рис. 23).

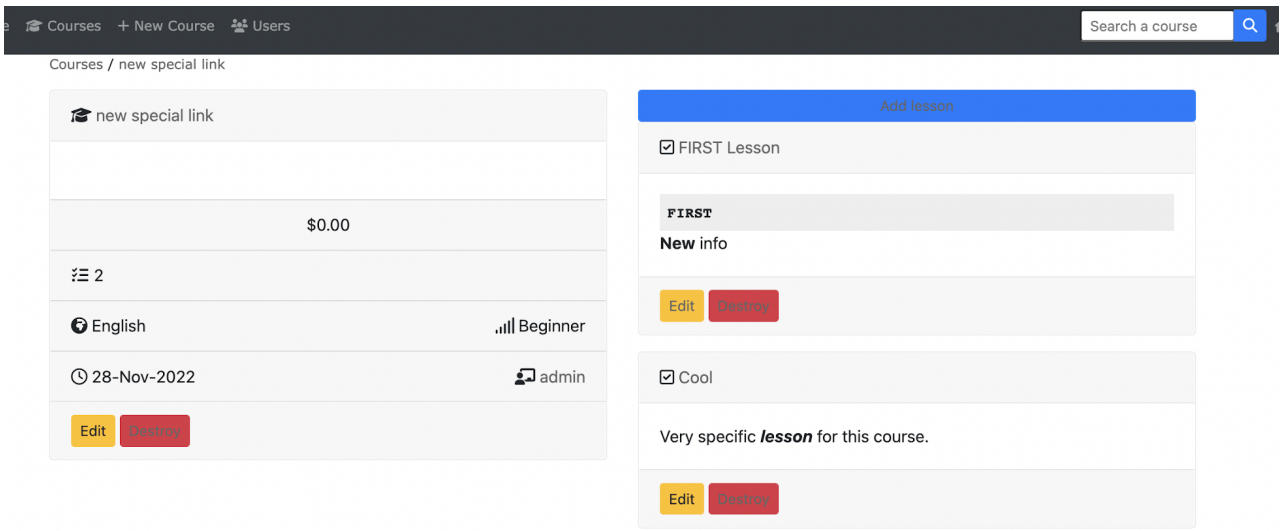

Рис. 23

Аналогічна форма для заповнення для Lessons, при натисканні кнопки «Додати Урок» (Add Lesson), де потрібно заповнити такі дані: заголовок, контекст уроку та відео/фото матеріали. Текст контексту також можна стилізувати. Є наявна валідація полів (рис. 24).

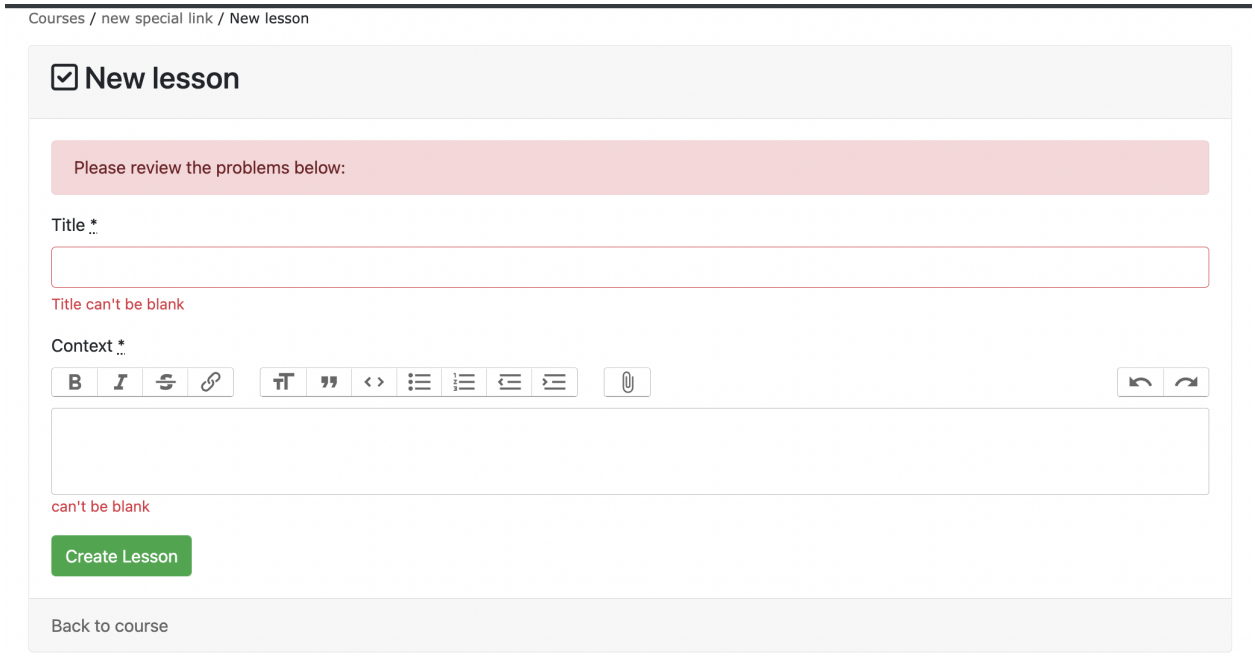

Рис. 24

Натиснувши на пункт меню «Активність» (рис. 19), можна побачити всю активність, яка є на онлайн-платформі, включаючи курси, які були додані, ким

були додані, яка назва курсу, чи можливо була певна зміна існуючого курсу та хто зробив певну операцію (рис. 25).

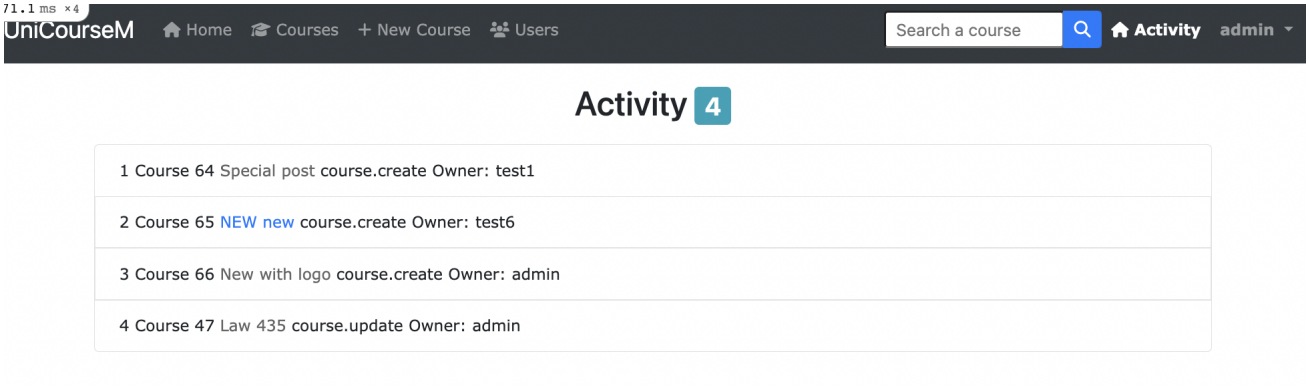

Рис. 25

Натиснувши на пункт меню у випадаючому списку «Налаштування Облікового запису» (Account Settings)(рис. 19), ми можемо побачити можливість зміни пошти користувача, зміни паролю та навіть видалити акаунт (рис. 26).

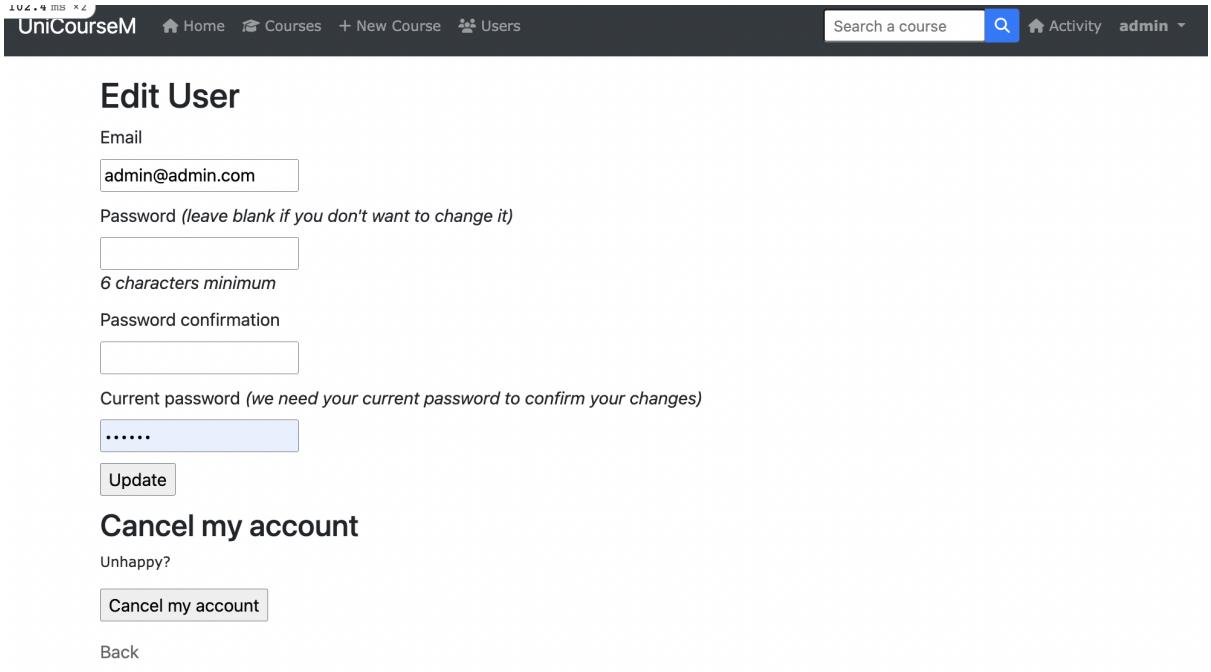

Рис. 26

2.5. Інструкція для адміністратора

Вхід у особистий кабінет для адміністратора відбувається так само як для звичайного користувача, тільки логін та пароль для ролі адміністратора є стандартним, який створюються після запуску міграцій БД для проекту.

Після входу до особистого кабінету з'являються нова сторінка на яку адмін має право на вхід: користувачі (Users). Відкриваємо її (рис.27).

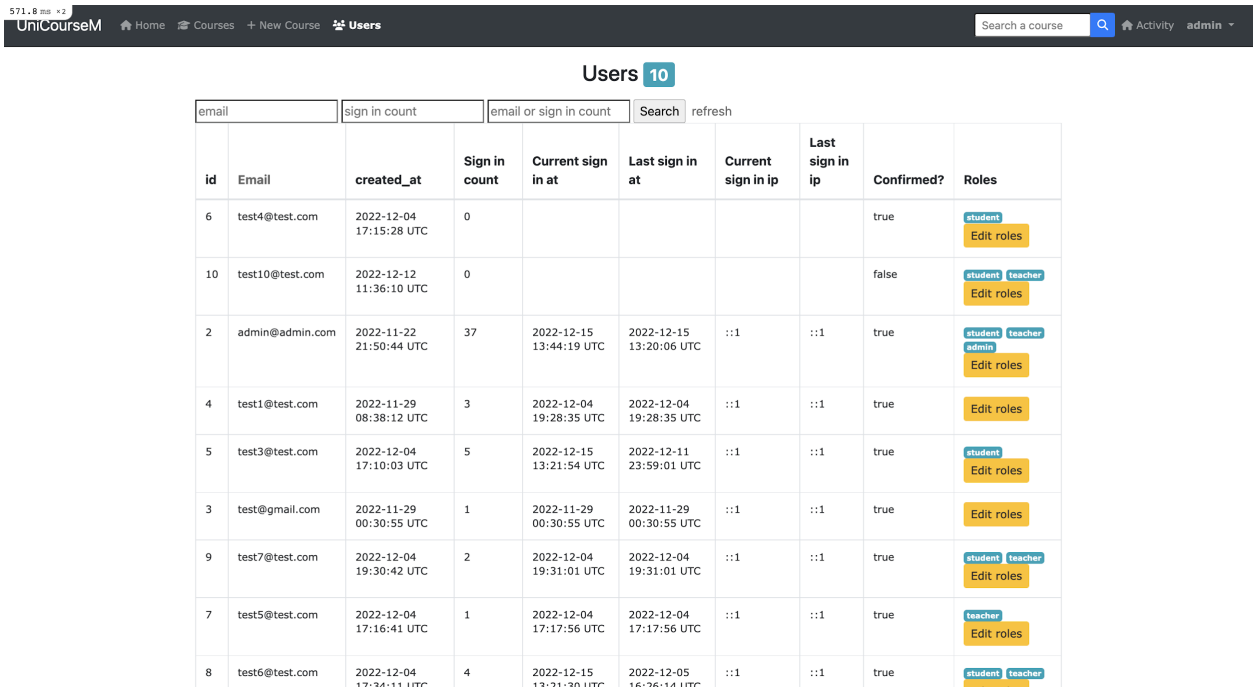

Рис. 27

На даній сторінці ми бачимо інформацію про всіх користувачів, які зареєстровані на онлайн-платформі, а саме: пошта, коли був створений акаунт, скільки разів авторизовувався користувач, чи в онлайні зараз, коли останній раз був в мережі (час та день), IP користувача коли він в онлайні, IP користувача коли останній раз заходив в систему, чи підтверджена пошта за допомогою переходу за посиланням на пошті та які ролі в кожного користувача.

Адміністратор має право змінити ролі для користувачів натиснувши на кнопку «Зміна ролей» (Edit roles).

Перейшовши на сторінку «Курси» адміністратор має право переглянути, змінити чи навіть видалити певний курс та аналогічно з уроками (Lessons) (рис.28).

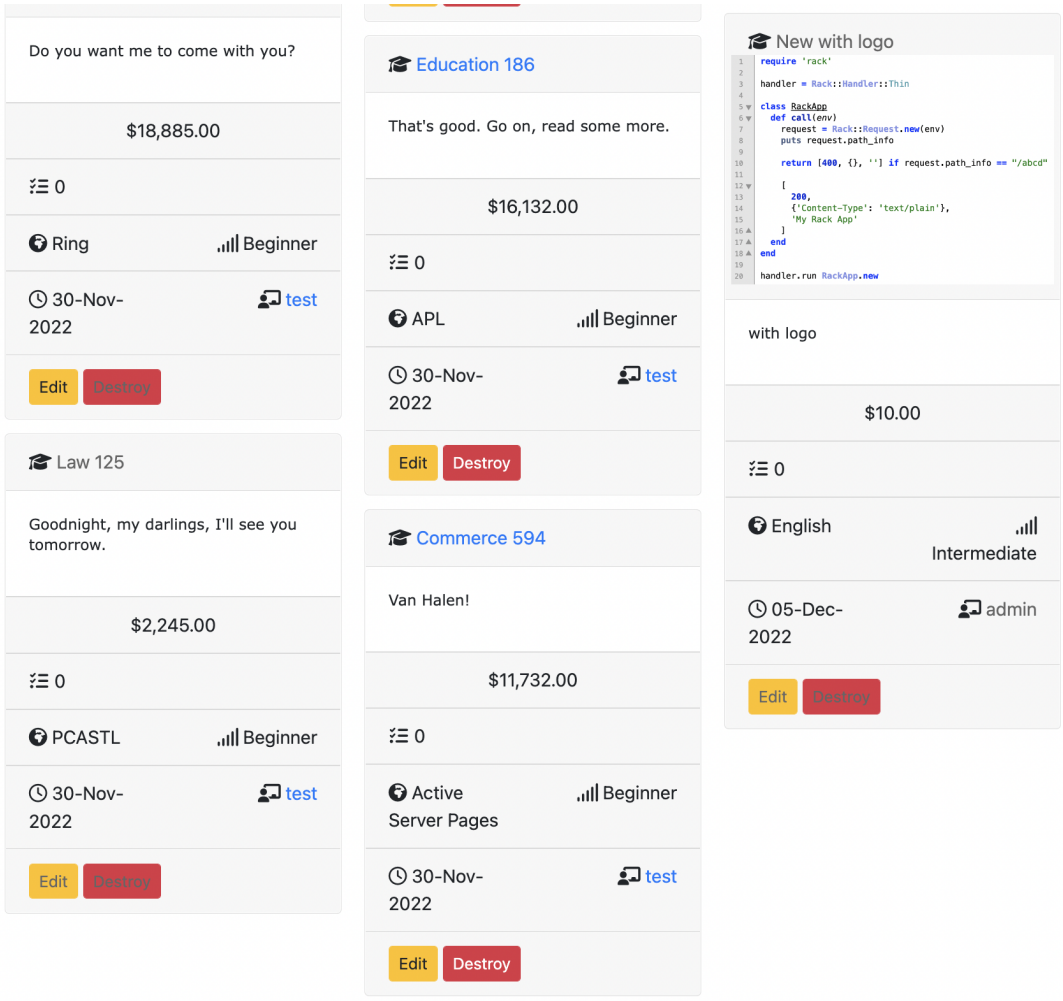

Рис. 28

Висновки

Під час виконання магістерської роботи було розроблено онлайн-платформу «UniCourseM». Створений веб-сайт дає можливість навчатися онлайн. За результатами виконання магістерської роботи можна зробити наступні висновки:

- покращено знання щодо Ruby та Ruby on Rails;
- досліджено та проаналізовано веб-сайти подібної тематики, порівняно їхні функціональні можливості та виділено їх переваги та недоліки;
- надано “майданчик” для викладачів щодо зручної публікації курсів для студентів;
- поглиблено знання у питанні хостингу додатків;
- проведено аналіз запитів до баз даних;
- усі складові проєкту ретельно протестовані перед публікацією.

Для розробки веб-сайту були обрані та засвоєні наступні технології: Ruby, JavaScript, Ruby on Rails, Bootstrap, PostgreSQL, Heroku та gem’и.

Ідея даної роботи є цікавою та актуальною, тому може розширюватися в плані написання нового функціоналу та доповнення нового дизайну.

Дана робота вважається завершеною в створенні онлайн-платформи, але в майбутньому, варто додати можливість онлайн оплати та розширення можливостей для користувача.

Вищезгадані технології є широко розповсюджені та легкими в користуванні, тому вони варті уваги, щоб вивчати їх ще глибше.

Список літератури

1. Про Ruby [Електронний ресурс]. – Режим доступу: https://www.ruby-lang.org/en/about/
2. Про багатотопоточність в Ruby [Електронний ресурс]. – Режим доступу: https://www.geeksforgeeks.org/ruby-introduction-to-multi-threading/
3. Уявіть, що ви могли б побудувати, якби вивчили Ruby on Rails [Електронний ресурс]. – Режим доступу: https://rubyonrails.org/
4. Щотижневі новини Ruby & Ruby on Rails [Електронний ресурс]. – Режим доступу: https://rubyweekly.com/
5. Lenz P. Build Your Own Ruby On Rails Web Applications. – SitePoint, 2007. – 447p.
6. Введення в Bootstrap бібліотеку [Електронний ресурс]. – Режим доступу: https://getbootstrap.com/docs/5.2/getting-started/introduction/
7. Що таке HTML? [Електронний ресурс]. – Режим доступу: https://futurenow.com.ua/shho-take-html/
8. CSS [Електронний ресурс]. – Режим доступу: https://avada-media.ua/ua/css/
9. jQuery. Загальна інформація. – [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/JQuery
10. Chaffer J., Swedberg K. Learning jQuery - Fourth Edition – R.: Packt Publishing, 2013. – 444 c.
11. Yarn: новий менеджер пакетів для JavaScript. – [Електронний ресурс]. – Режим доступу: https://engineering.fb.com/2016/10/11/web/yarn-a-new-package-manager-for-javascript/
12. Використання Webpacker у вашій програмі Ruby on Rails. – [Електронний ресурс]. – Режим доступу: https://blog.appsignal.com/2021/02/17/using-webpacker-in-your-ruby-on-rails-app-deep-dive.html
13. Репозиторій Ruby-Pg gem. – [Електронний ресурс]. – Режим доступу: https://github.com/ged/ruby-pg

14. Репозиторій Byebug gem. – [Електронний ресурс]. – Режим доступу: https://github.com/deivid-rodriguez/byebug
15. Репозиторій Haml-Rails gem. – [Електронний ресурс]. – Режим доступу: https://github.com/haml/haml-rails
16. Репозиторій Simple-Form gem. – [Електронний ресурс]. – Режим доступу: https://github.com/heartcombo/simple_form
17. Репозиторій Faker gem. – [Електронний ресурс]. – Режим доступу: https://github.com/faker-ruby/faker
18. Репозиторій Devise gem. – [Електронний ресурс]. – Режим доступу: https://github.com/heartcombo/devise
19. Репозиторій Friendly-Id gem. – [Електронний ресурс]. – Режим доступу: https://github.com/norman/friendly_id
20. Репозиторій Rolify gem. – [Електронний ресурс]. – Режим доступу: https://github.com/RolifyCommunity/rolify
21. Репозиторій Pundit gem. – [Електронний ресурс]. – Режим доступу: https://github.com/varvet/pundit
22. Документація Pagy. Вступ. – [Електронний ресурс]. – Режим доступу: https://ddnexus.github.io/pagy/index#gsc.tab=0
23. Репозиторій Wicked-PDF gem. – [Електронний ресурс]. – Режим доступу: https://github.com/mileszs/wicked_pdf
24. Функції Heroku. – [Електронний ресурс]. – Режим доступу: https://blog.heroku.com/data-in-functions

Додатки

1. Приклад побудови розділення обробки коду в 2-а потоки на мові Ruby:

```ruby
#!/usr/bin/ruby

# first method
def Geeks1
   a = 0
   while a <= 3

      puts "Geeks1: #{a}"
      sleep(1)

      a = a + 1
   end

end

# Second method
def Geeks2
   b = 0

   while b <= 3

      puts "Geeks2: #{b}"

      sleep(0.5)

      b = b + 1
   end

end

x = Thread.new { Geeks1() }

y= Thread.new { Geeks2() }

# wait for the first thread
# to finish
x.join

# wait for the second thread
# to finish
y.join


puts "Process End"
```

2. Команда для створення Ruby on Rails проекту:

```
$ bundle exec rails new bookstore
```

3. Команда для генерації моделі “Course”:

```
$ bundle exec rails generate model course
```

4. Модель “User”:

```ruby
class User < ApplicationRecord
  extend FriendlyId
  # Include default devise modules. Others available are:
  # :confirmable, :lockable, :timeoutable, :trackable and :omniauthable
  devise :database_authenticatable, :registerable,
         :recoverable, :rememberable, :validatable, :trackable,
:confirmable
  has_many :courses

  rolify
  friendly_id :email, use: :slugged
```

```
  def username
    self.email.split(/@/).first
  end

  after_create :assign_default_role

  def assign_default_role
    if User.count == 1
      self.add_role(:admin) if self.roles.blank?
      self.add_role(:teacher)
      self.add_role(:student)
    else
      self.add_role(:student) if self.roles.blank?
      self.add_role(:teacher)
    end
  end

  validate :must_have_a_role, on: :update

  def online?
    updated_at > 2.minutes.ago
  end

  private

  def must_have_a_role
    unless roles.any?
      errors.add(:roles, "must have at least one role")
    end
  end
end
```

5. Міграція моделі “User” для додавання авторизації:

```
# frozen_string_literal: true

class DeviseCreateUsers < ActiveRecord::Migration[6.1]
  def change
    create_table :users do |t|
      t.string :email,              null: false, default: ""
      t.string :encrypted_password, null: false, default: ""

      t.string   :reset_password_token
      t.datetime :reset_password_sent_at

      t.datetime :remember_created_at
      t.timestamps null: false
    end

    add_index :users, :email,                unique: true
    add_index :users, :reset_password_token, unique: true
  end
end
```

6. Команда для генерації «тіла» Lessons:

```
$ bundle exec rails g scaffold lessons title context:text
course:references
```

7. Контролер “UsersController”:

```
class CoursesController < ApplicationController
  before_action :set_course, only: %i[show edit update destroy]

  def index
    @ransack_courses = Course.ransack(params[:courses_search],
search_key: :courses_search)
    @courses = @ransack_courses.result.includes(:user)
  end

  def show
    @lessons = @course.lessons
  end

  def new
    @course = Course.new
    authorize @course
  end

  def edit
    authorize @course
  end

  def create
    @course = Course.new(course_params)
    authorize @course
    @course.user = current_user

    respond_to do |format|
      if @course.save
        format.html { redirect_to course_url(@course), notice: "Course
was successfully created." }
        format.json { render :show, status: :created, location: @course
}
      else
        format.html { render :new, status: :unprocessable_entity }
        format.json { render json: @course.errors, status:
:unprocessable_entity }
      end
    end
  end

  def update
    respond_to do |format|
      if @course.update(course_params)
        format.html { redirect_to course_url(@course), notice: "Course
was successfully updated." }
        format.json { render :show, status: :ok, location: @course }
      else
        format.html { render :edit, status: :unprocessable_entity }
        format.json { render json: @course.errors, status:
:unprocessable_entity }
      end
    end
  end

  def destroy
    authorize @course
    @course.destroy

    respond_to do |format|
      format.html { redirect_to courses_url, notice: "Course was
```

```
successfully destroyed." }
      format.json { head :no_content }
    end
  end

  private

  def set_course
    @course = Course.friendly.find(params[:id])
  end

  def course_params
    params.require(:course).permit(:title, :description,
:short_description, :price, :language, :level, :avatar)
  end
end
```

8. Правила відображення певних даних LessonPolicy:

```
class LessonPolicy < ApplicationPolicy
  class Scope < Scope
    def resolve
      scope.all
    end
  end

  def show?
    @user.has_role?(:admin) || @record.course.user_id == @user.id
  end

  def edit?
    @record.course.user_id == @user.id
  end

  def update?
    @record.course.user_id == @user.id
  end

  def new?
    @user.has_role?(:teacher)
  end

  def create?
    @record.course.user_id == @user.id
  end

  def destroy?
    @record.course.user_id == @user.id
  end
end
```

9. View partial для відображення курсів _course.html.haml:

```
card
  .card-header
    .fa.fa-graduation-cap
    = link_to course.title, course_path(course)
    - if course.avatar.attached?
      .row
        = image_tag course.avatar, height: '200px', width: '100%'
  .card-body
    %small= simple_format(course.short_description)
  .card-footer
    .text-center
      = number_to_currency(course.price)
```

```
  .card-footer
    .fa.fa-tasks
    = course.lessons.count
  .card-footer
    .row
      .col-md-6
        .fa.fa-globe-africa
        = course.language
      .col-md-6
        .text-right
          .fa.fa-signal
          = course.level
  .card-footer
    .row
      .col-md-6
        .far.fa-clock
        = course.created_at.strftime('%d-%b-%Y')
      .col-md-6
        .text-right
          .fa.fa-chalkboard-teacher
          = link_to course.user.username, user_path(course.user)
  - if current_user && policy(course).edit?
    .card-footer
      = link_to 'Edit', edit_course_path(course), class: 'btn btn-sm
btn-warning'
      = link_to 'Destroy', course, method: :delete, data: { confirm:
'Are you sure?' }, class: 'btn btn-sm btn-danger'
```

10. Параметри для підключення БД до різних середовищ:

```
default: &default
  adapter: postgresql
  encoding: unicode

  pool: <%= ENV.fetch("RAILS_MAX_THREADS") { 5 } %>

development:
  <<: *default
  database: uni_course_m_development

test:
  <<: *default
  database: uni_course_m_test

production:
  <<: *default
  database: uni_course_m_production
  username: uni_course_m
  password: <%= ENV['UNI_COURSE_M_DATABASE_PASSWORD'] %>
```

11. Тестові дані, які створюються для заповнення БД даними:

```
admin = User.new(email: "admin@admin.com", password: "123456",
password_confirmation: "123456")
admin.skip_confirmation!
admin.save!

user = User.new(email: "test@test.com", password: "123456",
password_confirmation: "123456")
user.skip_confirmation!
user.save!

30.times do
  Course.create!([{
    title: Faker::Educator.course_name,
```

```
    description: Faker::TvShows::GameOfThrones.quote,
    user_id: User.find_by(email: "test@test.com"),
    short_description: Faker::Quote.famous_last_words,
    language: Faker::ProgrammingLanguage.name,
    level: 'Beginner',
    price: rand(1000..20000)
  }])
end
```

12. Файл з повної структурою БД у форматі DSL:

```
ActiveRecord::Schema.define(version: 2022_12_09_141235) do
 enable_extension "plpgsql"

 create_table "action_text_rich_texts", force: :cascade do |t|
   t.string "name", null: false
   t.text "body"
   t.string "record_type", null: false
   t.bigint "record_id", null: false
   t.datetime "created_at", precision: 6, null: false
   t.datetime "updated_at", precision: 6, null: false
   t.index ["record_type", "record_id", "name"], name:
"index_action_text_rich_texts_uniqueness", unique: true
 end

 create_table "active_storage_attachments", force: :cascade do |t|
   t.string "name", null: false
   t.string "record_type", null: false
   t.bigint "record_id", null: false
   t.bigint "blob_id", null: false
   t.datetime "created_at", null: false
   t.index ["blob_id"], name:
"index_active_storage_attachments_on_blob_id"
   t.index ["record_type", "record_id", "name", "blob_id"], name:
"index_active_storage_attachments_uniqueness", unique: true
 end

 create_table "active_storage_blobs", force: :cascade do |t|
   t.string "key", null: false
   t.string "filename", null: false
   t.string "content_type"
   t.text "metadata"
   t.string "service_name", null: false
   t.bigint "byte_size", null: false
   t.string "checksum", null: false
   t.datetime "created_at", null: false
   t.index ["key"], name: "index_active_storage_blobs_on_key", unique:
true
 end

 create_table "active_storage_variant_records", force: :cascade do |t|
   t.bigint "blob_id", null: false
   t.string "variation_digest", null: false
   t.index ["blob_id", "variation_digest"], name:
"index_active_storage_variant_records_uniqueness", unique: true
 end

 create_table "activities", force: :cascade do |t|
   t.string "trackable_type"
   t.bigint "trackable_id"
   t.string "owner_type"
   t.bigint "owner_id"
   t.string "key"
   t.text "parameters"
   t.string "recipient_type"
```

```
    t.bigint "recipient_id"
    t.datetime "created_at", null: false
    t.datetime "updated_at", null: false
    t.index ["owner_id", "owner_type"], name:
"index_activities_on_owner_id_and_owner_type"
    t.index ["owner_type", "owner_id"], name:
"index_activities_on_owner_type_and_owner_id"
    t.index ["recipient_id", "recipient_type"], name:
"index_activities_on_recipient_id_and_recipient_type"
    t.index ["recipient_type", "recipient_id"], name:
"index_activities_on_recipient_type_and_recipient_id"
    t.index ["trackable_id", "trackable_type"], name:
"index_activities_on_trackable_id_and_trackable_type"
    t.index ["trackable_type", "trackable_id"], name:
"index_activities_on_trackable_type_and_trackable_id"
  end

  create_table "courses", force: :cascade do |t|
    t.string "title"
    t.text "description"
    t.datetime "created_at", precision: 6, null: false
    t.datetime "updated_at", precision: 6, null: false
    t.bigint "user_id", null: false
    t.string "slug"
    t.text "short_description"
    t.string "language", default: "English", null: false
    t.string "level", default: "Beginner", null: false
    t.integer "price", default: 0, null: false
    t.index ["slug"], name: "index_courses_on_slug", unique: true
    t.index ["user_id"], name: "index_courses_on_user_id"
  end

  create_table "friendly_id_slugs", force: :cascade do |t|
    t.string "slug", null: false
    t.integer "sluggable_id", null: false
    t.string "sluggable_type", limit: 50
    t.string "scope"
    t.datetime "created_at"
    t.index ["slug", "sluggable_type", "scope"], name:
"index_friendly_id_slugs_on_slug_and_sluggable_type_and_scope", unique:
true
    t.index ["slug", "sluggable_type"], name:
"index_friendly_id_slugs_on_slug_and_sluggable_type"
    t.index ["sluggable_type", "sluggable_id"], name:
"index_friendly_id_slugs_on_sluggable_type_and_sluggable_id"
  end

  create_table "lessons", force: :cascade do |t|
    t.string "title"
    t.text "context"
    t.bigint "course_id", null: false
    t.datetime "created_at", precision: 6, null: false
    t.datetime "updated_at", precision: 6, null: false
    t.string "slug"
    t.index ["course_id"], name: "index_lessons_on_course_id"
    t.index ["slug"], name: "index_lessons_on_slug", unique: true
  end

  create_table "roles", force: :cascade do |t|
    t.string "name"
    t.string "resource_type"
    t.bigint "resource_id"
    t.datetime "created_at", precision: 6, null: false
```

```
    t.datetime "updated_at", precision: 6, null: false
    t.index ["name", "resource_type", "resource_id"], name:
"index_roles_on_name_and_resource_type_and_resource_id"
    t.index ["resource_type", "resource_id"], name:
"index_roles_on_resource"
  end

  create_table "users", force: :cascade do |t|
    t.string "email", default: "", null: false
    t.string "encrypted_password", default: "", null: false
    t.string "reset_password_token"
    t.datetime "reset_password_sent_at"
    t.datetime "remember_created_at"
    t.datetime "created_at", precision: 6, null: false
    t.datetime "updated_at", precision: 6, null: false
    t.integer "sign_in_count", default: 0, null: false
    t.datetime "current_sign_in_at"
    t.datetime "last_sign_in_at"
    t.inet "current_sign_in_ip"
    t.inet "last_sign_in_ip"
    t.string "confirmation_token"
    t.datetime "confirmed_at"
    t.datetime "confirmation_sent_at"
    t.string "unconfirmed_email"
    t.string "slug"
    t.index ["confirmation_token"], name:
"index_users_on_confirmation_token", unique: true
    t.index ["email"], name: "index_users_on_email", unique: true
    t.index ["reset_password_token"], name:
"index_users_on_reset_password_token", unique: true
    t.index ["slug"], name: "index_users_on_slug", unique: true
  end

  create_table "users_roles", id: false, force: :cascade do |t|
    t.bigint "user_id"
    t.bigint "role_id"
    t.index ["role_id"], name: "index_users_roles_on_role_id"
    t.index ["user_id", "role_id"], name:
"index_users_roles_on_user_id_and_role_id"
    t.index ["user_id"], name: "index_users_roles_on_user_id"
  end

  add_foreign_key "active_storage_attachments", "active_storage_blobs",
column: "blob_id"
  add_foreign_key "active_storage_variant_records",
"active_storage_blobs", column: "blob_id"
  add_foreign_key "courses", "users"
  add_foreign_key "lessons", "courses"
end
```

13. Приклад тесту для CoursesControllerTest контролера:

```
require "test_helper"

class CoursesControllerTest < ActionDispatch::IntegrationTest
  setup do
    @course = courses(:one)
  end

  test "should get index" do
    get courses_url
    assert_response :success
  end

  test "should get new" do
```

```
    get new_course_url
    assert_response :success
  end

  test "should create course" do
    assert_difference('Course.count') do
      post courses_url, params: { course: { description:
@course.description, title: @course.title } }
    end

    assert_redirected_to course_url(Course.last)
  end

  test "should show course" do
    get course_url(@course)
    assert_response :success
  end

  test "should get edit" do
    get edit_course_url(@course)
    assert_response :success
  end

  test "should update course" do
    patch course_url(@course), params: { course: { description:
@course.description, title: @course.title } }
    assert_redirected_to course_url(@course)
  end

  test "should destroy course" do
    assert_difference('Course.count', -1) do
      delete course_url(@course)
    end

    assert_redirected_to courses_url
  end
end
```