МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

«Веб-платформа для онлайн курсів»

Завідувач
випускаючої кафедри **Довбиш А.С.**

Керівник роботи **Проценко О.Б.**

Студента групи ІН-63 **Бакакін Д.А.**

СУМИ 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую_____

Зав. кафедрою Довбиш А.С.

"_____" _____2020 р.

ЗАВДАННЯ

до випускної роботи

Студента четвертого курсу, групи ІН-63 спеціальності "Інформатика" денної форми навчання Бакакіна Даніїла Андрійовича.

Тема: «Веб-платформа для онлайн курсів»

Затверджена наказом по СумДУ

№_____ от _____2020 р.

Зміст пояснювальної записки: 1) огляд літератури за темою; 2) постановка завдання й формування завдань дослідження; 3) опис основних положень і моделі веб-ресурсу; 4) реалізація веб-ресурсу; 5)аналіз отриманих результатів.

Дата видачі завдання "_____"_____2020 р.

Керівник випускної роботи _____ Проценко О.Б.

Завдання приняв до виконання _____ Бакакін Д.А.

РЕФЕРАТ

Записка: 55 стор., 15 рис., 1 додаток, 10 літературних джерел.

Об’єкт дослідження — веб-система корпоративного онлайн-навчання.

Мета роботи — розробити і програмно реалізувати веб-ресурс, для підвищення кваліфікації та збільшення продуктивності працівників компанії.

Результати — розроблений веб-ресурс за допомогою HTML/CSS, Javasctipt, Python, фреймворку Django. Веб-ресурс реалізований як платформа корпоративного онлайн-навчання .

ВЕБ-ПЛАТФОРМА, PYTHON, БАЗА ДАНИХ, DJANGO,
КОРПОРАТИВНІ ОНЛАЙН-ТРЕНІНГИ.

ЗМІСТ

ВСТУП

Оскільки технології розвиваються з високою швидкістю, а з кожним роком їх вплив збільшується на різні сфери життя кожної людини все більше, то сфера освіти також зазнає свого впливу від технологій. Ноутбуки, планшети, онлайн-платформи, інтерактивні заняття вже стали частиною сучасної онлайн-освіти. Проте в освіті починає з'являтися така галузь як онлайн-навчання та повне проходженнянавчання в режимі онлайн. Оффлайн-освіта поступово відходить на другий план та, можливо, не витримає конкуренції з онлайн-навчанням.

Ідея онлайн-освіти стала настільки популярною, що її підхопили й вітчизняні навчальні заклади. Так, ще у 2015 році Міністерство освіти і науки України на базі Рішельєвського ліцею (Одеса) розпочало експеримент із залучення до навчального процесу сучасних засобів комунікації. Було розроблено курси для додаткового навчання, також надаються матеріали для підготовки до олімпіад, іспитів та ЗНО. Цікаво, що доступ до курсів та вебінарів мають усі учні Одеської області.

Це набагато зручніше та швидше, аніж відвідувати курси підвищення кваліфікації. Тим паче, що вибір в онлайн-школі значно більший.

Ідея зі створення корпоративного онлайн-навчання, особисто для мене, є надзвичайно цікавою, але вона не нова.

У всьому світі корпоративне навчання користується великим попитом, серед компаній, фірм, організацій, що бажають досягти відчутних позитивних результатів у своїй роботі.

Під корпоративним навчанням розуміють підвищення освіти й одержання нових навичок та вмінь співробітниками однієї компанії. Метою корпоративного навчання є підвищення ефективності роботи кожного співробітника окремо і всієї компанії в цілому. Керівництво компанії

встановлює цілі та вирішує завдання, учасників процесу навчання, його вигляд і спосіб проведення.

Все більше бізнес-структур переймаються важливістю ведення корпоративного навчання. За матеріалами досліджень західних фахівців при збільшенні витрат на корпоративне об'єднання всього на 10% загальна ефективність роботи компанії виростає на 9%.

Щоб бізнес був успішним, потрібно постійно працювати над рівнем кваліфікації персоналу, тому що людські ресурси - фактор, який має прямий вплив на ефективність бізнес-процесів будь-якої компанії. Але якісне навчання персоналу вимагає тимчасових витрат і, природно, додаткових фінансових вкладень, що є перешкодою для проведення корпоративного навчання в регулярному режимі. Однак відповідно до світових трендів, стає можливим цілодобове навчання на основі найбільш релевантної та актуальної інформації за допомогою використання сучасних гаджетів

В результаті проведеної роботи буде отримано повністю працездатний сайт, на якому можна знайти потрібний предмет і курс від досвідченого спеціаліста,. Також користувач зможе розмістити свої матеріали з одного чи іншого предмету. Це гарний сайт для бажаючих познайомитись з новими сферами діяльності. Щоб розпочати навчання не потрібно їхати із міста або за кордон. Достатньо мати лише доступ до інтернету.

На відкритому ринку слухач курсів сам вибирає, чого йому вчитися. У корпоративному навчанні онлайн-курси найчастіше призначаються. У цьому випадку замовником виступає не слухач, а компанія, яка навчає співробітників, щоб вирішити бізнес-завдання.

Іноді проекти з корпоративного онлайн-навчання розглядаються як складна довга інвестиція: компанії шукають платформу, проводять тендери і не вирішуються вибрати. Більш того, завдання впровадження навчання на всю компанію може виявитися занадто громіздкою. Можна спробувати почати з безкоштовних інструментів для організації онлайн-навчання

(Google-диск, онлайн-опитувальники для тестів, Zoom, Hangout або інші відкриті аналоги для відео-навчання). Якщо пілотний проект запустився, можна його розвивати і масштабувати.

Веб-система корпоративного онлайн-навчання допоможе значно розширити та вдосконалити професійні навички працівників.

1 ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Загальна інформація про веб-сайти

Веб-сайт - це колекція пов'язаних мережевих веб-ресурсів, таких як веб-сторінки, мультимедійний вміст, які зазвичай ідентифікуються загальним доменним ім'ям і публікуються принаймні на одному веб-сервері. Примітними прикладами є wikipedia.org, google.com і amazon.com, тощо.

Доступ до веб-сайтів здійснюється через загальнодоступну мережу Інтернет-протоколу (IP), наприклад Інтернет, або приватну локальну мережу (LAN) за допомогою уніфікованого локатора ресурсів (URL), який ідентифікує сайт.

Веб-сайти можуть мати багато функцій і можуть використовуватися в різних модифікаціях; веб-сайт може бути особистим веб-сайтом, корпоративним веб-сайтом для компанії, урядовим веб-сайтом, веб-сайтом організації та ін. Веб-сайти, як правило, присвячені певній темі або цілям, починаючи від розваг та соціальних мереж до надання новин та освіти. Усі загальнодоступні веб-сайти спільно становлять Всесвітню павутину, а приватні веб-сайти, такі як веб-сайт компанії для своїх співробітників, зазвичай є частиною інтрамережі.

Веб-сторінки, які є будівельними блоками веб-сайтів, є документами, які, як правило, складаються у вигляді звичайного тексту, поєднаного з інструкціями форматування мови гіпертекстової розмітки (HTML, XHTML). Вони можуть включати елементи з інших веб-сайтів з відповідними якорями розмітки. Доступ до веб-сторінок здійснюється за допомогою протоколу передачі гіпертексту (HTTP), який може додатково використовувати шифрування (HTTP Secure, HTTPS) для забезпечення безпеки та конфіденційності для користувача. Застосування користувача, часто веб-браузер, надає вміст сторінки відповідно до своїх інструкцій з розмітки HTML на дисплейному терміналі.

Гіперпосилання між веб-сторінками передає читачеві структуру сайту і направляє навігацію по сайту, яка часто починається з домашньої сторінки, що містить каталог веб-контенту сайту. Деякі веб-сайти вимагають реєстрації користувача або підписки на доступ до вмісту. Приклади веб-сайтів підписки включають багато ділових сайтів, новинних веб-сайтів, веб-сайтів академічних журналів, ігрових веб-сайтів, веб-сайтів для обміну файлами, дошок оголошень, веб-пошти, веб-сайтів соціальних мереж, веб-сайтів, що надають дані про фондовий ринок у реальному часі, а також сайти, що надають різні інші послуги. Кінцеві користувачі можуть отримувати доступ до веб-сайтів на різних пристроях, включаючи настільні та портативні комп'ютери, планшетні комп'ютери, смартфони та смарт-телевізори.

1.2 Система управління навчанням

Система управління навчанням(LMS) - це платформа для електронного навчання. Ключові принципи її роботи криються в самій абревіатурі.

Learning - навчання. За допомогою LMS створює єдину базу електронних курсів і навчальних матеріалів. Така база - справжнє джерело знань по вашій темі. Завдяки їй ви зберігаєте і нарощуєте внутрішню експертизу компанії.

Management - управління. Керувати в LMS можна курсами, а можна учнями.

На відміну від файлообмінників, LMS - це не просто звалище файлів, а добре організована система, де ви керуєте процесом. Для старту навчання досить додати співробітників і призначити їм курси.

Прийняли на роботу новачків? Вишліть їм запрошення в електронний клас для проходження вступного курсу. Просіли продажу? Надішліть менеджерів попрактикуватися в роботі з віртуальними клієнтами. Адже управління - це ще і про зв'язок успіхів у навчанні та рішення щоденних бізнес-задач.

System - електронна система. Навіть якщо ваші співробітники перебувають у різних містах, навчить їх всіх, не виходячи з власного офісу. До того ж, LMS автоматизує всю саму нудну і монотонну роботу: перевірка тестів, збір статистики і підготовка звітів.

Виходить, що LMS - це щось на зразок вашого власного онлайн-університету. Система допомагає створювати і зберігати електронні курси, забезпечує учням доступ до них і допомагає вам оцінити результати.

Очевидні переваги використання LMS в тому, що така система заощаджує час і гроші на виїздах тренерів, допомагає витримувати єдині стандарти роботи в філіях, складати індивідуальні плани розвитку та отримувати наочну статистику успішності.

Нижче наведені приклади завдань, які компанії ефективно вирішують за допомогою LMS:

- адаптація новачків
- Регулярне навчання співробітників
- Навчання співробітників в філіях
- проведення атестацій
- Підготовка кадрового резерву
- Створення єдиної внутрішньої бази знань
- Навчання партнерів і клієнтів

З кожним днем онлайн-навчання стає все більше популярним не лише у світі, але й в Україні. Адже завдяки різноманітним проектам всі охочі можуть стати студентами. Головне — це бажання вчитись та доступ до інтернету. Перший аналог – Український сервіс Prometheus позиціонує себе як громадський проект масових відкритих онлайн-курсів. Незважаючи на те, що з’явився він відносно недавно (в 2014 році), одразу зумів стати популярним серед людей, які прагнуть нових знань. Завдяки цій платформі кожен в Україні, незалежно від віку, соціального стану та місця проживання,

може стати студентом Тут ви знайдете безкоштовні навчальні курси університетського рівня від викладачів КНУ, КПІ та Києво-Могилянської академії.

На рисунках 1.1 – 1.3 представлені деякі сторінки сайтів аналогів.

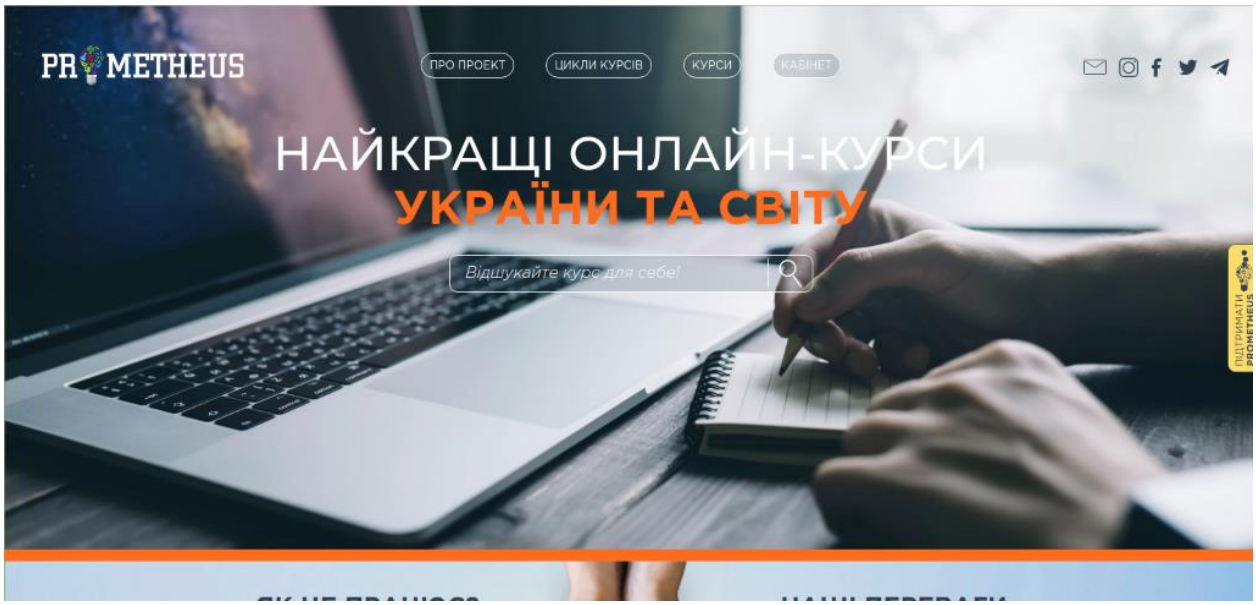

Рисунок 1.1 – Головна сторінка Prometheus

Наступний аналог – Coursera — освітній проект, який пропонує своїм слухачам сотні безкоштовних онлайн-курсів з різних предметів від провідних американських університетів. У разі успішного закінчення,студент отримує сертифікат про проходження курсу. Студенти самі обирають час для навчання, переглядають відеолекції, виконують домашні завдання та перевіряють виконані роботи інших студентів.

Викладачі встановлюють мінімальну кількість балів, які має набрати студент, тому багато слухачів “відсіюються” під час навчання. Найкоротші курси тривають 3 тижні, найпоширеніші — 6-тижневі, однак є й такі, які тривають 10 та більше тижнів.

Зважаючи на те, що відеолекції читають англійською мовою, найпопулярніші з них субтитровані українською мовою.

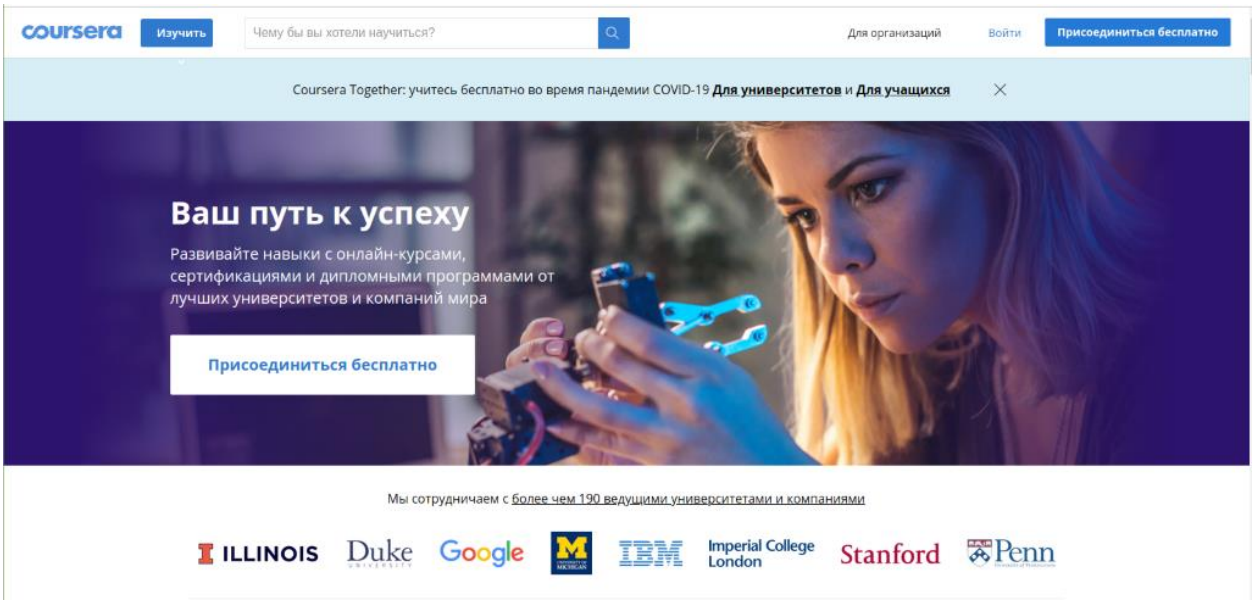

Рисунок 1.2 – Головна сторінка сайту Coursera

Наступний аналог – TED (Technology, Entertainment, Design) — некомерційний проект, який кожного року в Единбурзі та Лонг-Біч збирає зі всього світу науковців, бізнесменів, політиків, активістів. Мета конференції — поширити серед суспільства унікальні та цікаві ідеї. Після конференції їхні промови з’являються на сайті TED, переглядати які може кожен охочий.

На сайті можна знайти більше 2 тисяч відео, до більшої частини з яких є субтитри українською мовою, а деякі — озвучені нею. Неважливо, чим ви займаєтесь в житті, різноманітність тем на сайті вражає, а тому кожен може знайти цікаву для себе промову. Тут є відповідь на запитання як спілкуються бактерії, як подолати лінь, як отримати максимум від онлайн-освіти та багато іншого.

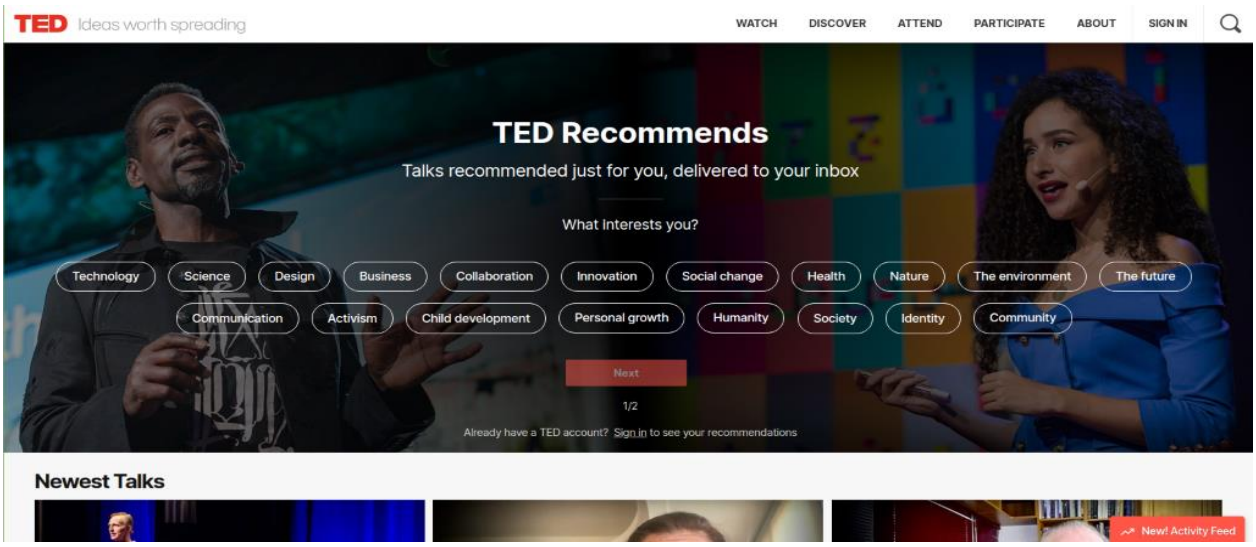

Рисунок 1.3 – Головна сторінка сайту TED.

1.3 Постановка задачі

Метою цієї роботи є створення програми, побудоваяної таким чином щоб процес отримання знань був простим і зрозумілим.Створена програма повинна бути незалежна від пристроїв та операційних систем.Головний зміст даної програми(сайту) полягає в тому, щоб максимально спростити процедуру отримання знань.До функціональних вимог входить наявність must-have функцій, таких як управління користувачами і курсами, вибір моделі навчання (чисто онлайн-навчання, змішане онлайн-офлайн або ж під керівництвом інструктора), підтримка та створення навчального контенту.

На сайті будуть присутні такі розділи, як безпосередньо предмети, курси, модулі, створення модулів та предметів, запис на курс. Також доступна функція надання прав на створення курсу користувачу.

Основні вимоги для сайту:

1) Можливість знаходження потрібної інформації щодо запису на курс;
2) Перегляд матеріалів в режимі онлайн;

3) Сучасний дизайн;
4) Стабільність роботи сайту;
5) Зручність використання сайту;
6) Наповнення сайту контентом;
7) Структура, тобто зручне розміщення інформації на сайті;
8) Зручні інтерфейси створення курсів, тесті;
9) Доступ до списку предметів, фахівця які він викладає;
10) Можливість створювати акаунти фахівців та учнів;

2 ПРОГРАМНІ ЗАСОБИ ДЛЯ РЕАЛІЗАЦІЇ ЗАДАЧІ

2.1 Вибір дизайну сайту

Матеріальний дизайн – (англ. Material design; кодова назва "Quantum Paper") — принципи дизайну сайтів, програмного забезпечення і застосунків від компанії Google. Вперше представлений на конференції Google I/O 25 червня 2014 року. Ідея дизайну полягає в інтерфейсі, поведінка і вигляд якого наслідують правила поведінки і вигляду паперових карток в реальному житті. Існує також визначення, що матеріальний дизайн є візуальною чи дизайн мовою.

Особливості дизайну:

Дослідження, зроблене дизайнерами Google, змінює підхід до інтерфейсу і зробити його реальним. Інтерфейс перетворюється на живу приємну людині взаємодію.

Матеріальний дизайн за допомогою системи управління тінями

створює візуальну ілюзію простору між застосунком та екраном пристрою.

Була збільшена реалістичність анімації за допомогою прискорення та пригальмовування рухів, пружного стрибання об'єктів.

Анімація є ключевым компонентом Material Design, але вона не може бути там просто заради руху. Анімація повинна відбуватися в єдиному середовищі, служить для того, щоб було легше сфокусуватися на об’єкті і повинна включати в себе зрозумілі та прості переходи.

2.2 Вибір мов програмування

При створенні сайту для макету взята мова розмітки гіпертекстових документів HTML, а також мова, яка відповідає за візуальне представлення документів користувачу CSS. Для написання web-додатків (сценаріїв), що виконуються на Web-сервері буде використовуватися Python. Додатково буде

в меншій мірі використовуватися JavaScript з метою зробити web-сторінки “живими”.

HTML5 – сучасна версія мови гіпертекстової розмітки, яка отримала ряд доповнень та нових можливостей. До складу робочої групи її розробників входять представники таких гігантів IT-сфери, як Google, Apple, IBM, Microsoft та інші.

Багато експертів вважають HTML5 не продовженням розвитку мови HTML, а новою відкритою програмною платформою, яка застосовується для створення веб-інтерфейсів, що використовують текстові та мультимедійні інструменти (відеоролики, графіка, аудіозаписи та інші елементи). При цьому у п'ятій версії зберігається зворотна сумісність і читабельність коду для користувачів.

Хорошим доповненням для HTML стали CSS (Cascading Style Sheets) – каскадні таблиці стилів, які містять опис особливостей зовнішнього вигляду веб-документа, створеного на основі інструментів гіпертекстової розмітки. В основному CSS використовується для візуального оформлення веб-сторінок. Тобто, HTML задає структуру ресурсу, а таблиці стилів визначають те, як він буде виглядати.

Саме CSS відповідають за привабливість і дизайн сторінок. Ця технологія дозволяє «навести красу». Вже судячи з назви, вона працює зі стилями (кольорами, шрифтами, формами і т. д.)

CSS може застосовуватися не тільки в мовах розмітки HTML та XHTML, але і працювати в XML-форматі (включаючи документи SVG і HUL).

CSS3 – третя версія таблиці каскадних стилів, яка значно розширила можливості попередніх поколінь. Особливість CSS3 полягає в тому, що з її допомогою можна створювати анімовані елементи без допомоги JS. Вона включає підтримку різних градієнтів і тіней, використовує нові форми згладжування і т. д.

Ядро JavaScript включає цілий ряд функцій, які дають наступні можливості:

- Зберігати дані в змінних;
- Активувати частину коду у відповідності з певними сценаріями, які здійснюються на сторінці сайту;
- Створювати контент, який оновлюється автоматично;
- Управляти мультимедійними можливостями (працювати з відео, анімувати зображення).
- Усього кілька рядків коду скрипта роблять дивовижні речі. Саме тому мова настільки популярна серед розробників.

Головним фактором мови Python є надійність. Python повинен надати програмісту засоби для швидкого і ефективного вирішення поставлених завдань. Практичний характер Python обумовлений важливими характеристиками:

- надійність;
- структура;
- читабельність;
- простотою;
- ефективністю;
- безпекою;
- гнучкістю;
- універсальність;
- управління пакетами;
- Велика спілка розробників.

Існує ще одна «характеристика», яка робить Pyhon особливо привабливим: він поширюється безкоштовно! Причому, з відкритими початковими кодами (Open Source).

2.3 Середовище розробки PyCharm IDE

Інтегроване середовище розробки (IDE) (англ. Integrated Development Environment) - система програмних засобів, яка використовується програмістами для розробки програмного забезпечення.

Підтримка основних фреймворків

PyCharm ідеально підходить для роботи з Django, Pyramid, Flask и Tornado та інших фреймворків.

Всі інструменти для написання Pyhon коду

Редактор фактично "отримує" код і глибоко розуміє його структуру, підтримуючи всі можливості мови Pyhon для сучасних і застарілих проектів. Він забезпечує найкраще завершення коду, рефакторинг, запобігання помилок на льоту та багато іншого.

Включені Front-end технології

Також можна скористатися передовими технологіями інтерфейсу, такими як HTML, CSS, JavaScript, TypeScript, Angular, React та Vue.js з доступними рефакторингами, налагодженням і модульним тестуванням, дивитися зміни негайно в браузері завдяки Live Edit.

Вбудовані інструменти розробника

Виконувати багато рутинних завдань безпосередньо з середовища IDE, завдяки інтеграції систем керування версіями, підтримці віддаленого розгортання, баз даних / SQL, інструментах командного рядка, Docker, Composer, Django Rest Api Client та багатьох інших інструментів.

Інтелектуальна допомога в кодуванні

Сотні перевірок піклуються про перевірку коду під час введення тексту, аналізуючи весь проект. Автодоповнення коду, аранжування та форматування, швидкі виправлення та інші функції допомагає записати акуратний код, який легко підтримувати.

Навігація смарт-кодом

ReSharper розширює можливості вбудованого автодоповнення PyCharm. Автодоповнення в ReSharper звужує список запропонованих варіантів зі зворотнім відліком, підтримує скорочення lowerCamelHumps, пропонує імена змінних і полів в залежності від імен їх типів і використовуваних правил іменування, автоматично імпортує вибрані типи і методи розширення.

Легке налагодження та тестування

Графічний відладчик PyCharm робить процес максимально простим, наочно візуаліруя налагодження в зручному форматі. С PyCharm ви як вдома і зможете швидко почати користуватися основними функціями налагодження.

2.4 Фреймворк Django

З Django ви можете перенести веб-додатки з концепції на запуск за лічені години. Django бере на себе більшу частину турбот веб-розробки, тому ви можете зосередитися на написанні свого застосування без необхідності винаходити велосипед. Це безкоштовно і з відкритим вихідним кодом.

- Django був розроблений, щоб допомогти розробникам якомога швидше перенести додатки з концепції в доробку.
- Django включає в себе десятки доповнень, які ви можете використовувати для вирішення типових завдань веб-розробки. Django піклується про аутентифікації користувачів, адмініструванні контенту, картах сайтів, RSS-каналах і багатьох інших завданнях - прямо з коробки.
- Django серйозно ставиться до безпеки і допомагає розробникам уникнути багатьох поширених помилок безпеки, таких як SQL-ін'єкція, міжсайтовий скриптинг, підробка міжсайтових запитів і перехоплення. Його система аутентифікації користувачів забезпечує безпечний спосіб управління обліковими записами користувачів і паролями.

- Деякі з найбільш завантажених місць на планеті використовують можливості Django для швидкого і гнучкого масштабування для задоволення найвищих вимог трафіку.
- Компанії, організації та уряди використовували Django для створення самих різних речей - від систем управління контентом до соціальних мереж і наукових обчислювальних платформ.
- Поділ бізнес-логіки і візуальної частини на рівні архітектури.
- SEO-дружність.
- Можливість розширення.
- Розвинена інфраструктура: велика кількість бібліотек і плагінів.
- Численне і дружнє співтовариство, завдяки якому легко шукати відповіді на складні питання.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Проектування веб-системи

Для того, щоб розпочати безпосередньо створення сайту необхідно виконати наступні пункти:

- Будування UML діаграми;
- Будування DFD діаграми;
- Встановити PyCharm IDE;
- Встановити Django;
- Підключити базу даних ;
- Оновлення пакетів за допомогою pip;

UML - мова графічного опису для об'єктного моделювання в області розробки програмного забезпечення, моделювання бізнес-процесів, системного проектування та відображення організаційних структур.

В процесі проектування система розглядається з різних точок зору за допомогою моделей, різні уявлення яких подаються у формі діаграм.

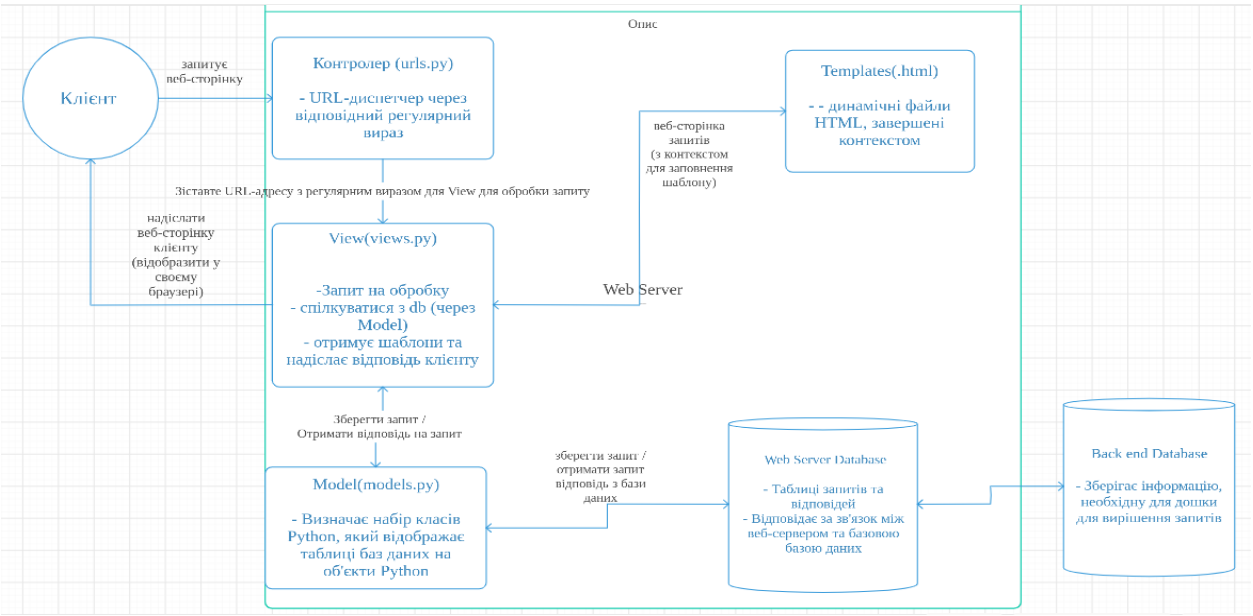

Рисунок 3.1 – Інформаційна модель веб-системи

Діаграма Прецедентів візуально зображає різноманітні сценарії взаємодії між акторами (користувачами) і прецедентами (випадками використання); описує функціональні аспекти системи (бізнес логіку). Діаграми Прецедентів відіграють важливу роль не тільки у комунікації між збирачами вимог до проекту і потенційними користувачами. Діаграми Прецедентів дописані бізнес логікою і детальними специфікаціями прецедентів, як джерельна інформація, успішно використовують учасники розробки проекту на всіх його фазах (зародження, дизайн, програмування, тестування, документування).

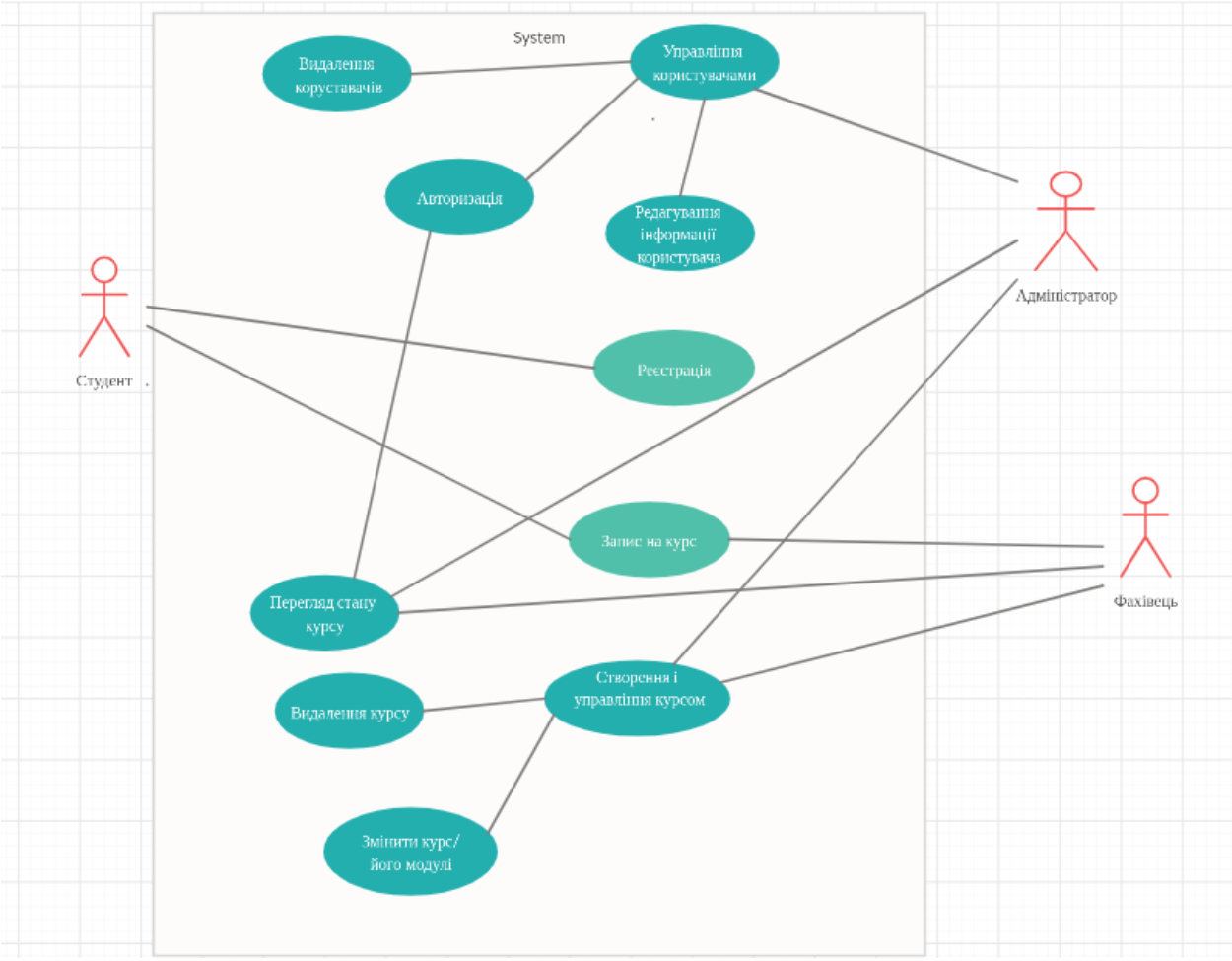

Рисунок 3.2 - Use Case Diagram

Контекстна діаграма в поєднанні з переліком системних відображує принцип роботи системи. Для систем зі складними зв'язками між об'єктами

важливо більш детально представлятимуть взаємини між об'єктами. У цих діаграмах об'єкти представляються прямокутниками, а зв'язки між ними - стрілками, на яких розташовані ромби. У прямокутниках і ромбах записані імена об'єктів і зв'язків. Тип зв'язку та її напрямок визначаються за допомогою стрілок на початку і в кінці лінії зв'язку[4].

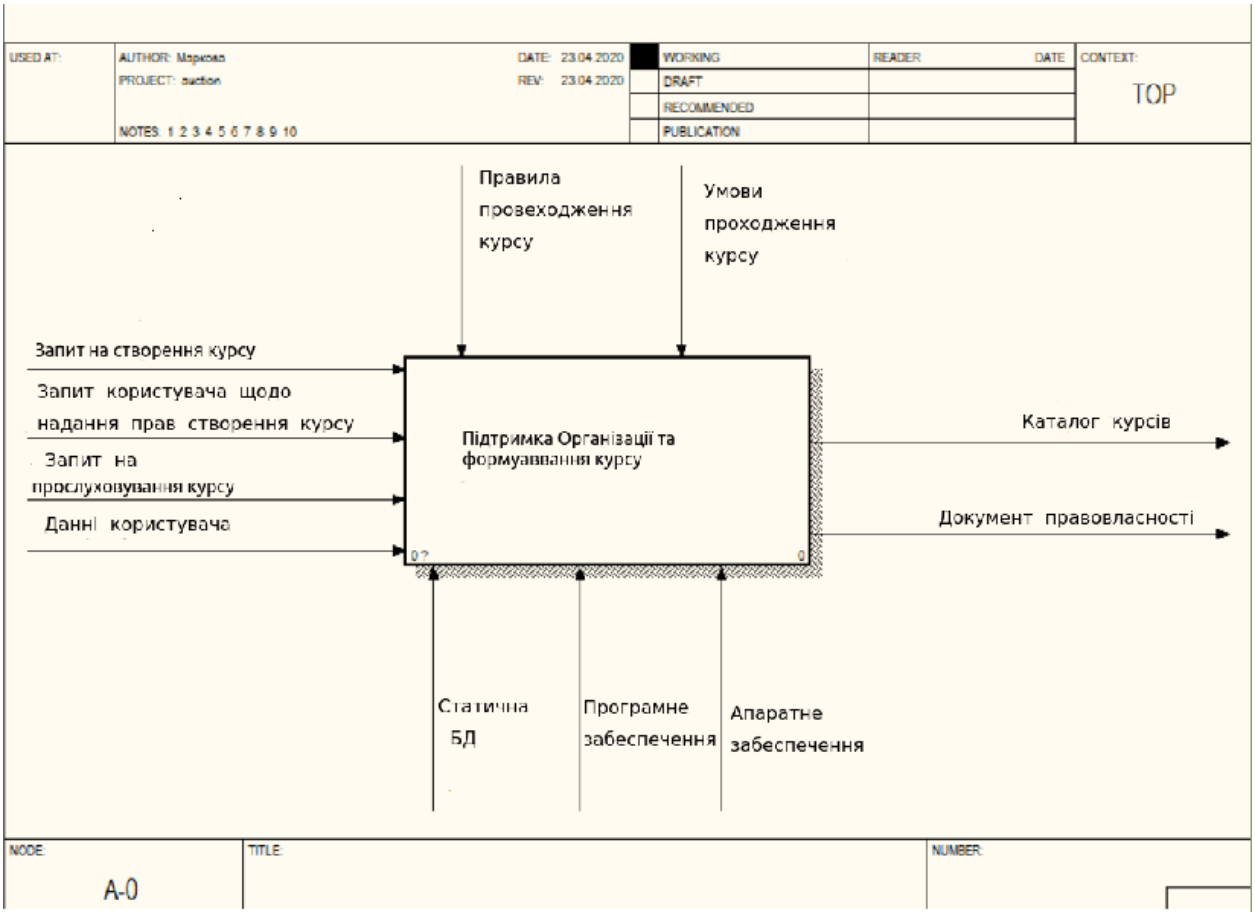

Рисунок 3.3 - Контекстна діаграма

DFD розглядає систему як сукупність предметів. Контекстна діаграма часто включає роботи і зовнішні посилання. Включення зовнішніх посилань в контекстну діаграму не скасовує вимоги методології чітко визначити мету, область і єдину точку зору на моделируемую систему.

Ця діаграма містить тільки одну функцію і один або декілька зовнішніх об'єктів, які обмінюються потоками з даною функцією. Хоча під час пізніх

стадій розробки функція буде розкладатися на складові, на контекстній діаграмі не показані підфункції, внутрішні потоки і зберігання даних. Ця діаграма корисна для відображення області або меж системи в цілому і об'єктів поза системою, з якими вона взаємодіє. Не розглядаються потоки даних між зовнішніми об'єктами, оскільки вони знаходяться поза області опису системи[4].

Діаграми потоків даних (Data flow diagramming, DFD) використовуються для опису документообігу і обробки інформації. Подібно до IDEF0, DFD представляє модельну систему як мережу пов'язаних між собою робіт. Їх можна використовувати як доповнення до моделі IDEF0 для наочного відображення поточних операцій документообігу в корпоративних системах обробки інформації.

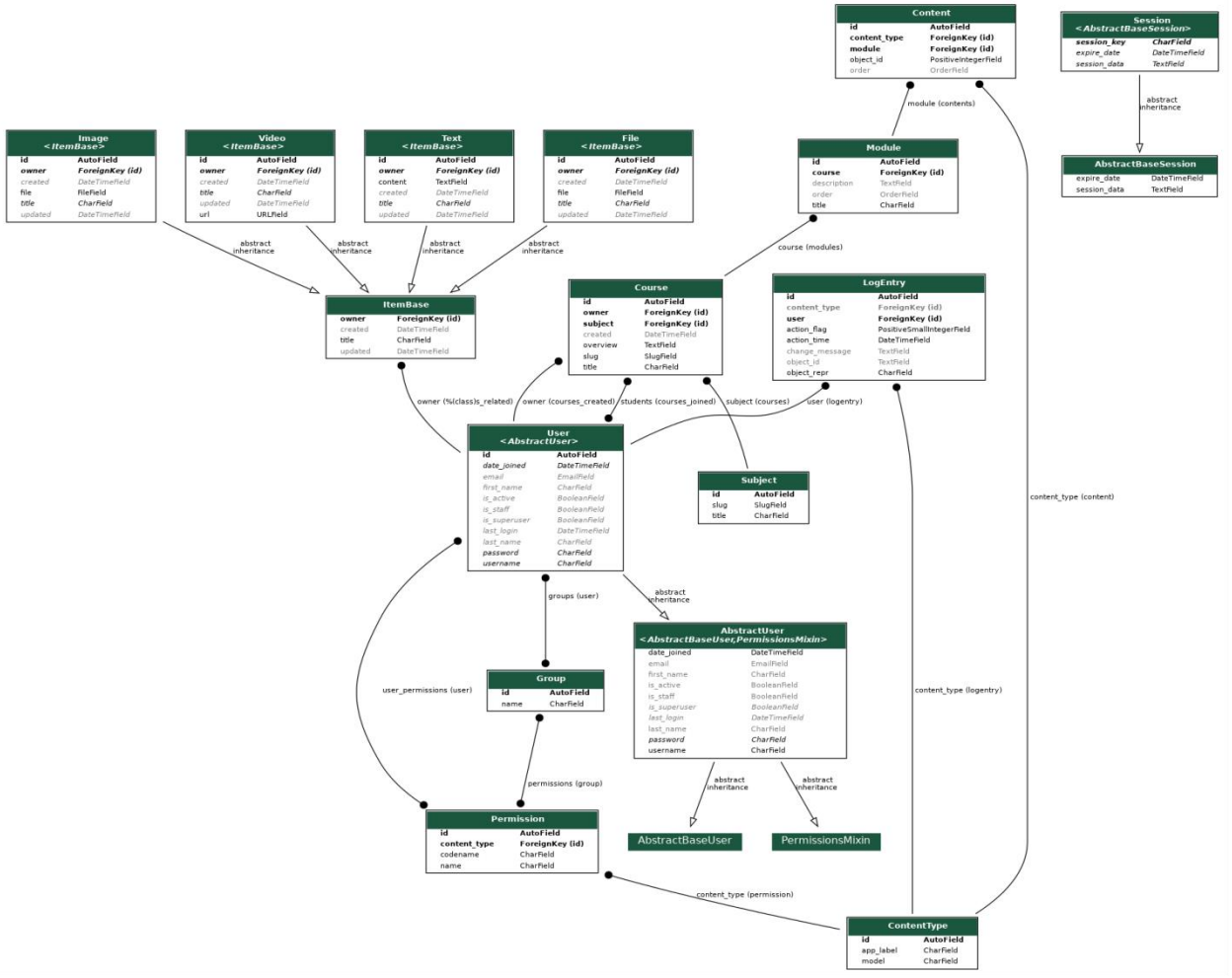

Рисунок 3.3 - DFD діаграма

Інсталяція редактору не займає багато часу. Для цього необхідно перейти на офіційну сторінку JetBrains і завантажити PyCharm IDE.

Після завантаження встановити його на комп'ютер в декілька кроків. Django має декілька системних вимог. Для початку треба встановити віртуальне середовище.

Virtualenv ізолювати ваше Python/Django середовище, для кожного проекту. Це означає, що будь-які зміни, внесені на одному сайті не вплинуть на будь-які інші, які ви також розробляєте.

Після активації віртуального середовища треба інсталювати в нього Django

Django використовує irtualenv для управління своїми залежностями. Не використовуйте глобальне оточення Python для залежностей вашого проекту, тому що це може призвести до виникнення конфліктів залежностей. Python не вміє працювати з кількома версіями пакетів одночасно. Це стане проблемою, якщо різним проектам потрібні різні, несумісні версії одного пакета.

Команда python manage.py runserver використовується для запуску програми з використанням сервера. Команда python manage.py виконує опції: хост може використовуватися для зміни адреси програми (за замовчуванням це 127.0.0.1 і 8000 порт. Параметр port можна використовувати для зміни порту, на який програма реагує (за умовчанням це 8000). Також python manage.py створює та застосовує міграції бази данних[8].Для роботи з базами даних в проекті Django в файлі settings.py визначено параметр DATABASES, який за замовчуванням виглядає наступним чином:

Це є вільним програмним забезпеченням і має на меті бути легким для Конфігурація використовуваної бази даних в даному випадку складається з двох параметрів. Параметр ENGINE вказує на використовуваний движок для доступу до БД. В даному випадку це вбудований пакет

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

django.db.backends.sqlite3. Другий параметр - NAME вказує на шлях до бази даних. Після першого запуску проекту в ньому за умовчанням буде створено файл db.sqlite3, який власне і буде використовуватися в якості бази даних. Microsoft SQL або PostgreSQL. В Django реалізовано об'єктно-реляційне відображення (ORM), яке забезпечує взаємодію додатки з базами даних (БД). ORM автоматично передає дані з БД, наприклад, PostgreeSQL або MySQL, в об'єкти, які використовуються в коді програми.ORM прискорює розробку прототипів і готових веб-додатків на Django[8]. Розробнику навіть не потрібно знати мову, який використовується для взаємодії з базами даних.Також ORM дозволяє швидко перемикатися між базами даних з мінімальними змінами коду. Наприклад, ви можете використовувати SQLite на локальному сервері, а потім переключитися на MySQL на production-сервері. Однак для мінімізації помилок краще використовувати одну базу даних під час розробки і в продакшені.Згідно з вимогами до сайту створено базу данних**.**

Вбазістворено 5 таблиць: Subject, Course, Module, Content, ItemBase. Таще 4 таблицідлязберіганнярізноготипуінформації: Text, File, Image, Video.

В данному випадку міграції подібні до керування версіями для бази даних, що дозволяє легко змінювати та використовувати схему бази даних програми. Міграції, як правило, поєднуються з конструктором схем Django, щоб легко створити схему бази даних програми. Міграції вирішують проблему, коли, наприклад, вручну потрібно додати стовпець до схеми локальної бази данних.

```
class Subject(models.Model):
  title = models.CharField(max_length=250)
 slug = models.SlugField(max_length=200, unique=True)
 class Meta:
   ordering = ['title']
 def __str__(self):
    return self.title
```

Таблиця предметів

```
class Course(models.Model):
      owner = models.ForeignKey(User,
                        related_name='courses_created',
                        on_delete=models.CASCADE)
   subject = models.ForeignKey(Subject,
                         related_name='courses',
                         on_delete=models.CASCADE)
   title = models.CharField(max_length=250)
   slug = models.SlugField(max_length=250, unique=True)
   overview = models.TextField()
   created = models.DateTimeField(auto_now_add=True)
   students = models.ManyToManyField(User,
                              related_name='courses_joined',
                              blank=True)
   class Meta:
      ordering = ['-created']
   def __str__(self):
```

```
        return self.title
```

Таблиця курсів

3.2Опис основного функціоналу веб-сайту

Для того, щоб мати можливість стати случем курсу або його керівником потрібно зареєструватися:

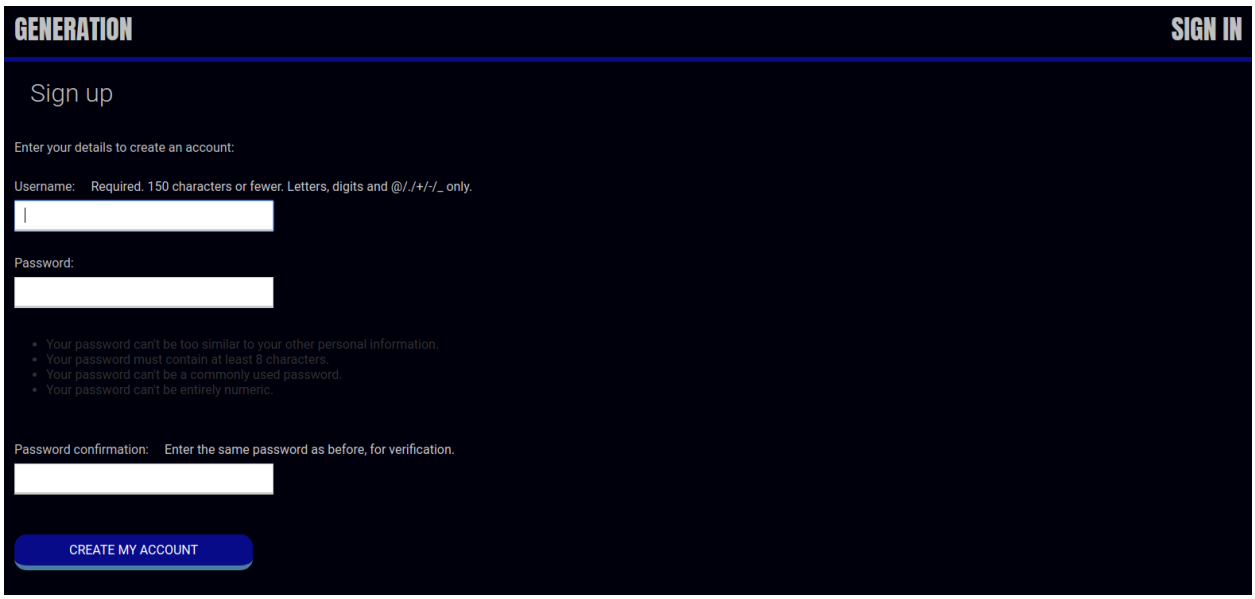

Рисунок 3.4 – Форма авторизації

```
{% extends "base.html" %}
{% blocktitle %}
Sign up
{% endblock %}
{% block content %}
<h1>
        Sign up
</h1>
<div class="module">
<p>Enter your details to create an account:</p>
<form action="" method="post">
```

```
        {{ form.as_p }}
        {% csrf_token %}
<p><input type="submit" value="Create my account"></p>
</form>
</div>
{% endblock %}
```

На даній сторінці реалізована реєстрація та авторизація на сайті.

Рисунок 3.5 – Форма авторизації

На даній сторінці також реалізовано валідацію введених даних користувача для входу на сайт.

```
{% extends "base.html" %}
{% block title %}Log-in{% endblock %}
{% block content %}
<h1>Log-in</h1>
<div class="module">
   {% if form.errors %}
```

```
<p>Your username and password didn't match. Please try again.</p>
   {% else %}
<p>Please, use the following form to log-in:</p>
   {% endif %}
<div class="login-form">
<form action="{% url 'login' %}" method="post">
      {{ form.as_p }}
      {% csrf_token %}
<input type="hidden" name="next" value="{{ next }}" />
<p><input type="submit" value="Log-in"></p>
</form>
</div>
</div>
{% endblock %}
```

Рисунок 3.6 – Головна сторінка авторизованого користувача

Користувач може додавати до списку прослуховування курси, та переглядати їх вміст.

Якщо користувач охочий до створення свого курсу адміністратор може надати йому права групи фахівців в адміністративній панелі. Він буде мати права створювати курси, модулі, створювати та прикріпити відео лекції та додавати матеріал у різному форматі.

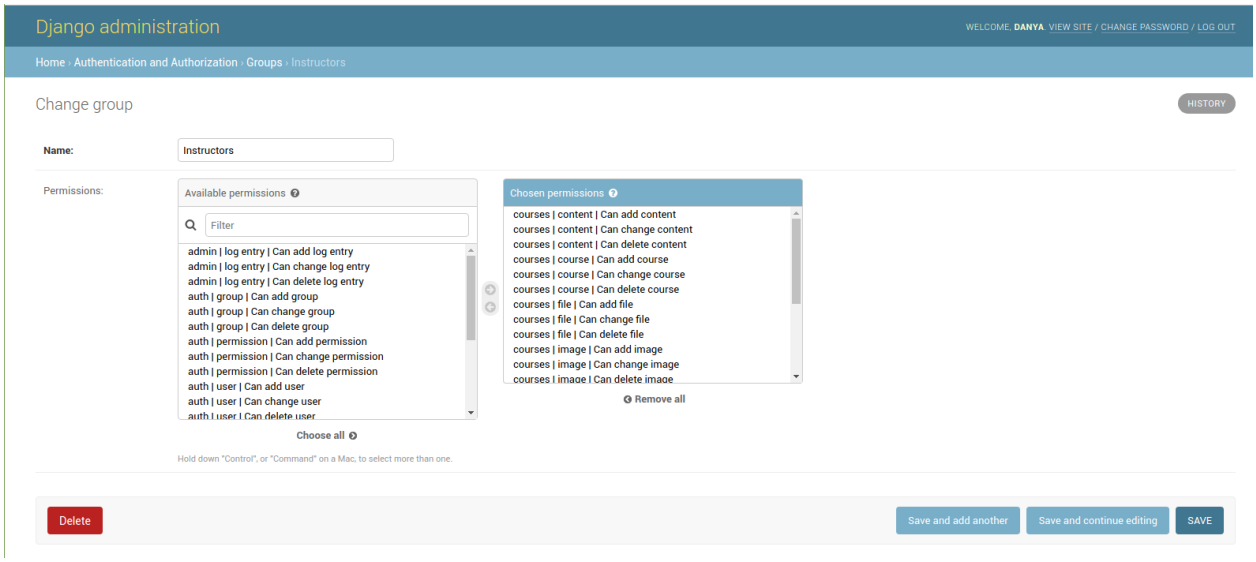

Рисунок 3.7 – Права користувачів на сайті

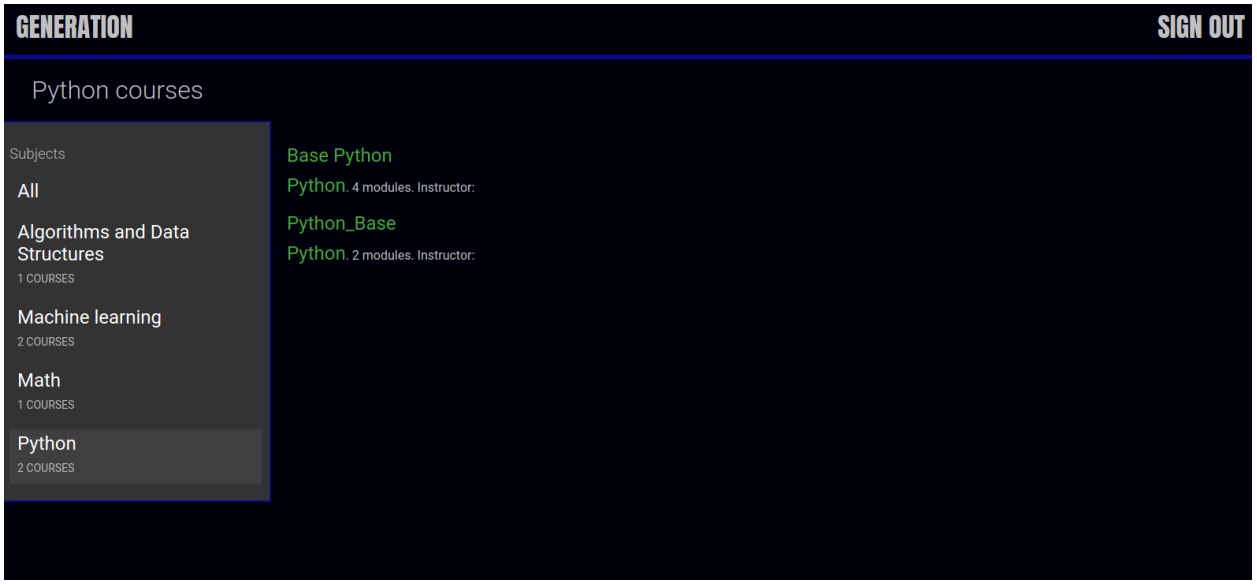

Рисунок 3.8 – Сторінка вибраного курсу

```
{% extends "base.html" %}
{% block title %}
  {% if subject %}
    {{ subject.title }} courses
  {% else %}
    All courses
  {% endif %}
{% endblock %}
{% block content %}
<h1>
  {% if subject %}
    {{ subject.title }} courses
  {% else %}
    All courses
  {% endif %}
</h1>
<div class="contents">
<h3>Subjects</h3>
<ul id="modules">
<li {% if not subject %}class="selected"{% endif %}>
<a href="{% url 'course_list' %}">All</a>
</li>
    {% for s in subjects %}
<li {% if subject == s %}class="selected"{% endif %}>
<a href="{% url 'course_list_subject' s.slug %}">
            {{ s.title }}
<br><span>{{ s.total_courses }} courses</span>
```

```
</a>
</li>
        {% endfor %}
</ul>
</div>
<div class="module">
   {% for course in courses %}
        {% with subject=course.subject %}
<h3><a href="{% url 'course_detail' course.slug %}">{{ course.title }}</a></h3>
<p>
<a href="{% url 'course_list_subject' subject.slug %}">{{ subject }}</a>.
                {{ course.total_modules }} modules.
                Instructor: {{ course.owner.get_full_name }}
</p>
        {% endwith %}
   {% endfor %}
</div>
{% endblock %}
```

Якщо тема курсу зацікавила користувача, система пропонує ознайомитись з вмістом курсу.

Рисунок 3.9 – Опис курсу

Перед тим, як стати користувачем з роллю «слухач курсу», користувач має можливість ознайомитись з описом курсу (див. рис.3.9).

```
{% extends "base.html" %}
{% block title %}
  {{ object.title }}
{% endblock %}
{% block content %}
  {% with subject=course.subject %}
<h1>
      {{ object.title }}
</h1>
<div class="module">
<h2>Overview</h2>
<p>
<a href="{% url 'course_list_subject' subject.slug %}">{{ subject.title }}</a>.
        {{ course.modules.count }} modules.
        Instructor: {{ course.owner.get_full_name }}
```

```
</p>
        {{ object.overview|linebreaks }}
        {% if request.user.is_authenticated %}
<form action="{% url 'student_enroll_course' %}" method="post">
            {{ enroll_form }}
            {% csrf_token %}
<input type="submit" class="button" value="Enroll now">
</form>
        {% else %}
<a href="{% url 'student_registration' %}" class="button">
            Register to enroll
</a>
        {% endif %}
</div>
   {% endwith %}
{% endblock %}
```

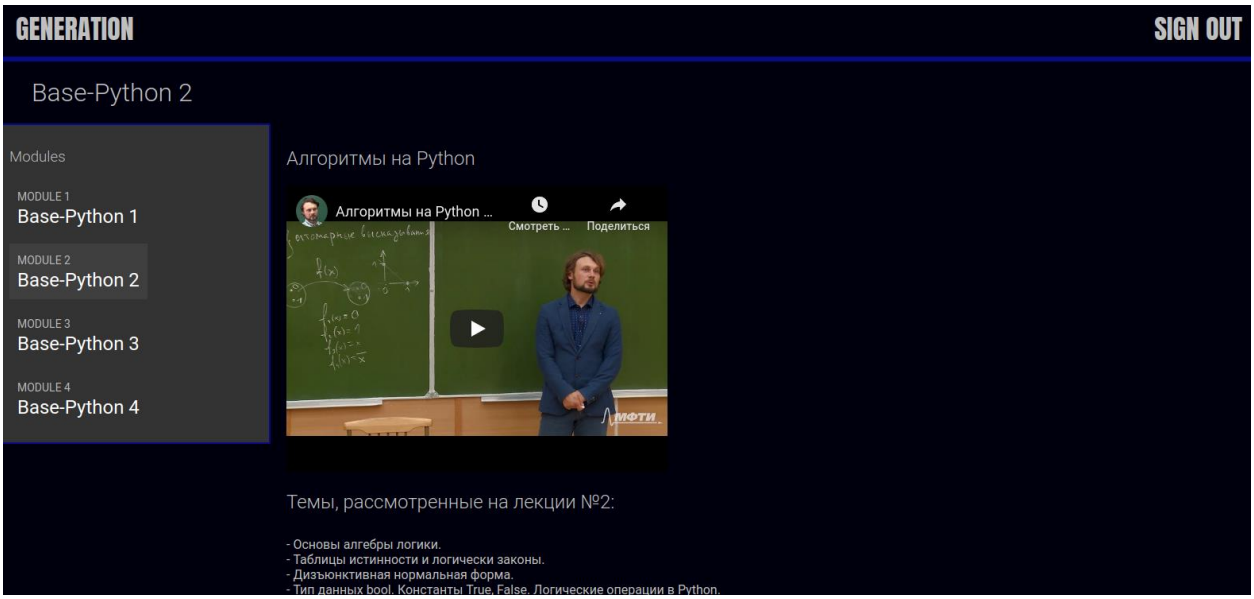

Рисунок 3.10 – Вміст курсу

```
{% extends "base.html" %}
{% block title %}
  {% if subject %}
    {{ subject.title }} courses
  {% else %}
    All courses
  {% endif %}
{% endblock %}
{% block content %}
<h1>
  {% if subject %}
    {{ subject.title }} courses
  {% else %}
    All courses
  {% endif %}
</h1>
<div class="contents">
<h3>Subjects</h3>
<ul id="modules">
<li {% if not subject %}class="selected"{% endif %}>
<a href="{% url 'course_list' %}">All</a>
</li>
    {% for s in subjects %}
<li {% if subject == s %}class="selected"{% endif %}>
<a href="{% url 'course_list_subject' s.slug %}">
            {{ s.title }}
<br><span>{{ s.total_courses }} courses</span>
```

```
</a>
</li>
        {% endfor %}
</ul>
</div>
<div class="module">
   {% for course in courses %}
        {% with subject=course.subject %}
<h3><a href="{% url 'course_detail' course.slug %}">{{ course.title }}</a></h3>
<p>
<a href="{% url 'course_list_subject' subject.slug %}">{{ subject }}</a>.
                {{ course.total_modules }} modules.
                Instructor: {{ course.owner.get_full_name }}
</p>
        {% endwith %}
   {% endfor %}
</div>
{% endblock %}
```

Після переходу користувач зможе ознайомитись з вмістом кожного модуля. Число модулів не обмежено лсистемою і створюється фахівцем - розробником курса.

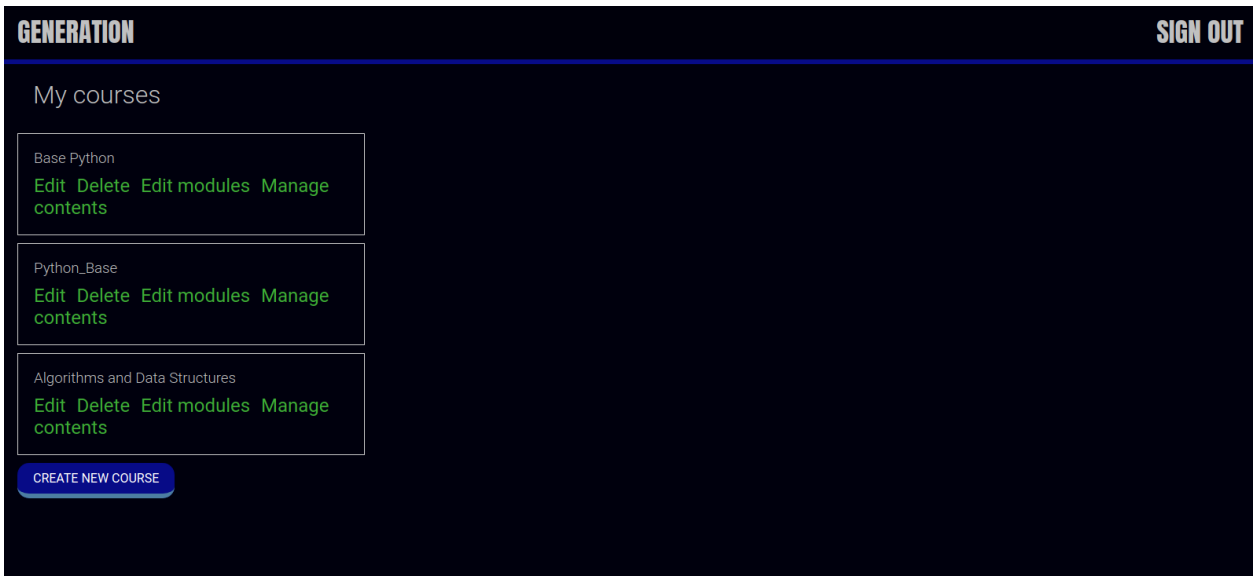

Рисунок 3.10 – Сторінка фахівця

Сторінка управління контентом фахівця, саме тут він має інструменти для створення курсів та додавання учбових матеріалів.

```
{% extends "base.html" %}
{% block title %}My courses{% endblock %}
{% block content %}
<h1>My courses</h1>
<div class="module">
   {% for course in object_list %}
<div class="course-info">
<h3>{{ course.title }}</h3>
<p>
<a href="{% url 'course_edit' course.id %}">Edit</a>
<a href="{% url 'course_delete' course.id %}">Delete</a>
<a href="{% url 'course_module_update' course.id %}">Edit modules</a>
        {% if course.modules.count > 0 %}
```

```
<a href="{% url 'module_content_list' course.modules.first.id %}">Manage contents</a>
        {% endif %}
</p>
</div>
  {% empty %}
<p>You haven't created any courses yet.</p>
  {% endfor %}
<p>
<a href="{% url 'course_create' %}" class="button">Create new course</a>
</p>
</div>
{% endblock %}
```

Рисунок 3.11 – Сторінка створення курсу

Слід зазначити, що користувач з правами фахівця може створювати курс, але не предмет. Предмет може створювати лише супер користувач з доступом до адміністративній панелі сайту.

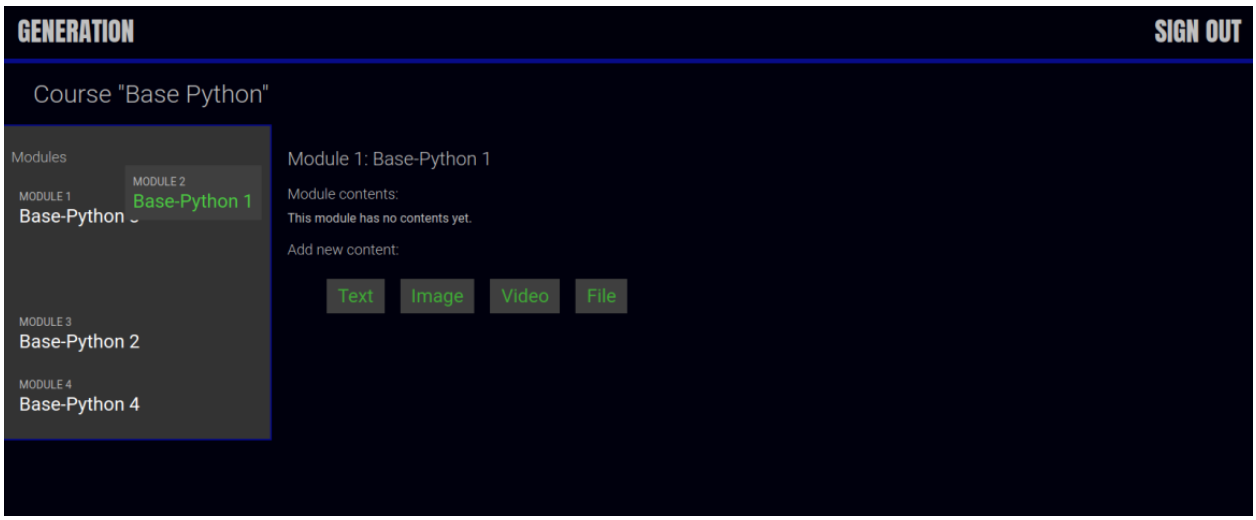

Рисунок 3.12 – Сторінка управління контентом

За допомогою JS можна змінювати порядок модулів на курса.

```
{% extends "base.html" %}
{% block title %}
 {% if object %}
  Edit content "{{ object.title }}"
 {% else %}
  Add a new content
 {% endif %}
{% endblock %}
{% block content %}
<h1>
  {% if object %}
   Edit content "{{ object.title }}"
  {% else %}
   Add a new content
  {% endif %}
</h1>
```

```
<div class="module">
<h2>Course info</h2>
<form action="" method="post" enctype="multipart/form-data">
    {{ form.as_p }}
    {% csrf_token %}
<p><input type="submit" value="Save content"></p>
</form>
</div>
{% endblock %}
```

З першог опогляду в шаблоні управління контентом немає JavaScript коду. Тоді встає логічне запитання, як працює функція змінення порядку модулів. За цей функціонал відповідає перший рядок коду {% extends "base.html" %}.

Всі відомі фреймворки використовують наслідування шаблонів і тому в головному шаблоні base.html, для підключення JavaScript потрібно створити блок domready в кінці головного шаблону. Всі відомі фреймворки використовують наслідування шаблонів і тому в головному шаблоні base.html, для підключення JavaScript потрібно створити блок domready в кінці головного шаблону.

```
<script>
$(document).ready(function() {
   {% block domready %}
   {% endblock %}
 });
</script>
```

3.3 Кешуваннявеб-сайту

Головнаособливістьсучаснихвеб-сайтіввтому, щовонидинамічні. Коженраз, колиКористувачзапитуєсторінку, веб-сервервиробляєкупудій, відзапитівдобазиданихдообробкишаблонів, щобстворитисторінку, якупобачитьвідвідувачвашогосайту. Це вимагає більше витрат, з точки зору використання ресурсів, ніж статична видача файлу з файлової системи. Для більшості веб додатків такі зайві обчислення не особливо помітні. Ці програми не є сайтами washingtonpost.com або slashdot.org. Це сайти невеликого або середнього розміру з невеликим трафіком. Але для більш великих сайтів стає дуже важлива економія ресурсів.З цього моменту в справу вступає кешування.Кешування означає збереження результатів дорогого обчислення, щоб уникнути його повторного обчислення наступного разу.

Django поставляється з надійною системою кешування, яка дозволяє вам зберігати динамічні сторінки так, що не буде потрібно їх створювати для кожного запиту. Для зручності, Django дає можливість тонкого управління рівнем кешування: ви можете кешувати результат роботи уявлення, ви можете кешувати тільки ті шматки, які важко обчислювати або ви можете кешувати весь ваш сайт[9].

Mamcached - найшвидший і ефективний тип кешу, доступний Django, Memcached є кешем, який повністю розташовується в оперативній пам'яті, він був розроблений для LiveJournal.com і пізніше переведений в опенсорс компанією Danga Interactive. Він використовується такими сайтами як Facebook і Wikipedia для зниження навантаження на базу даних і значного збільшення продуктивності сайту.

Memcached працює як демон і захоплює певний обсяг оперативної пам'яті. Його завданням є представлення швидкого інтерфейсу для додавання, отримання і видалення певних даних в кеші. Всі дані зберігаються

прямо в оперативній пам'яті, таким чином немає ніякого додаткового навантаження на базу даних або файлову систему.

Щоб налаштувати кешування потрібно внести корективи в настройки проекту, а саме в файл settings.py.

Рівні кешування

- Django підтримує кешування даних на декількох рівнях, представлених нижче:
- низькорівневий API - надає можливість кешувати найменшу одиницю обчислень (запити або обчислення);
- рівень оброблювачів - кешуються результати обробки одногоHTTP-запиту;
- рівень шаблонів - застосовується для додавання в кеш результату генерації HTML-шаблону або його фрагмента;
- рівень сайту - кешує весь сайт

Для покращення працездатності сайту було вирішено кешувати на рівні обробників та шаблонів для перегляду матеріалу користувачами.

Для кешування обробників потрібно лише додати cache_page (60 * 15) до шляху обобника у файлі urls.py та огорнути {% cache 600 module_contents module %} у такий блок у ту частину шаблону яка відповідає за відображення курсів. Таким чином користувачу не буде потрібно чекати відображення матеріалів які він переглядав 15 хвилин назад.

3.4 Налаштування Django REST Framework

До цього моменту переглядати вміст сайту міг кожен. Для того щоб анонімний користувач не зміг записатись на курс було вирішено задіяти Django REST Framework.

За допомогою налаштування REST_FRAMEWORK можна додатково настроїти фреймворк Django REST, який підтримує безліч різних параметрів

для перевизначення поведінки за замовчуванням. наприклад,настройка DEFAULT_PERMISSION_CLASSES відповідає за здатність за замовчуванням для доступу до читання, створення, зміни і видалення об'єктів. Ми вказали клас DjangoModelPermissionsOrAnonReadOnly, тому анонімні користувачі зможуть тільки переглядати, але не змінювати дані, а авторизовані матимуть доступ до всіх чотирьох дій.

Спочатку потрібно перетворити наші дані в формат JSON. Для цього потрібно створити файл serializers та импортувати до нього базу даних і серіалізувати її.

Серіалізатор дозволяють складні дані, такі як querysets і екземплярів моделі повинні бути перетворені в рідні типи даних Python, який може бути легко відтворювані в JSON, XMLілі інших типів контенту. Серіалізатор також забезпечують десеріалізацію, що дозволяє перетворювати проаналізовані дані назад в складні типи після попередньої перевірки вхідних даних.

Був створений файл permissions.py

permissions - є засобом контролю і регулювання доступу до певних функцій на рівні системи і пристрої

```
from rest_framework.permissions import BasePermission
class IsEnrolled(BasePermission):
    def has_object_permission(self, request, view, obj):
        return obj.students.filter(id=request.user.id).exists()
```

Цей класс перевіряє, чи є студент слухачем курсу за допомогою атрибуту students .

ВИСНОВКИ

У результаті виконаної роботи було розроблено систему управління навчанням, яка вирішує проблему створення та управлінні курсами та предметами для підвищення кваліфікації працівників.

Розроблена веб-система вирішує такі завдання як: створення користувачів, створення та перегляд курсу, запис слухачів на курс, а також доступ до матеріалів з будь якого місця на планеті за допомогою мережі інтернет.

У ході дослідження системи управління навчанням було виявлено що інвестування у розвиток співробітників є найкращим вкладом у розвиток компанії.

Системазабезпечує зручний пошук інформації, простий та інтуїтивно зрозумілий інтерфейс допоможе фахівцям та користувачам оволодіти системою, а для останніх були прибрані всі перешкоди у здобутті навичок.

Оскільки він був створений за допомогою Django сайт зможе витримувати велику кількість співробітників які водночас переглядають одні ті самі матеріали. Система добре захищена що є дуже важливим у корпоративній етиці.

СПИСОК ЛІТЕРАТУРИ

1. Федоров А.Г. JavaScript для всех. - М.: Высшая школа, 2010. - 384 с.
2. Дунаев В Web-программирование для всех - М. Высшая школа,2018.- 560 с.
3. Павловская Е.Э.Основы дизайна и композиции, современные концепции, М.:Юрайт, 2019. – 350с.
4. Технічна енциклопедія TechTrend [Електронний ресурс]. – Режим доступу: http://techtrend.com.ua/index.php?newsid=3511
5. Бизли Д., Джонс Б. К. Python. Книга рецептов / пер. с анг. Б. В. Уварова. – М.: ДМК Пресс, 2019. – 648 с.
6. Добірка безкоштовних онлайн-сервісів для навчання українською [Електронний ресурс].– Режим доступу: https://mizky.com/article/222/online-education-ua
7. Дистанційне навчання як сучасна освітня технологія [Електронний ресурс] : матеріали міжвузівського вебінару (м. Вінниця, 31 березня 2017 р.) / відп. ред. Л.Б.Ліщинська. – Вінниця : ВТЕІ КНТЕУ, 2017. – 102 с.
8. Django documentation [Електронний ресурс]. – Режим доступу: https://docs.djangoproject.com/en/3.0/
9. Система кешування Django [Електронний ресурс]. – Режим доступу:https://djbook.ru/rel1.8/topics/cache.html
10. DJANGO REST FRAMEWORK [Електронний ресурс]. https://www.django-rest-framework.org/
11. Кириченко А., Кришталь А. HTML5 + CSS3. Основи сучасного web-дизайну. -СПб:Питер, 2018. – 533с.
12. Мердок М., Мюллер Т. Вибух навчання. Дев'ять правил ефективного віртуального класу. М:Альпина Паблишер, 2012. – 190 с.
13. Форс Дж., Біссекс П., Уеслі Дж. Django. Розробка веб-додатків на Python – К.: НМК-Трейд, 2009. – 300с.

14. Метіз Е. Вивчаємо Python. Програмування ігор, візуалізація даних, веб-додатки.. –СПб:Питер, 2020. – 450 с.
15. Automate the Boring Stuff with Python: Practical Programming for Total Beginners - https://www.livelib.ru/book/1001380318-automate-the-boring-stuff-with-python-practical-programming-for-total-beginners-al-sweigart

ДОДАТОК

Views.py

```
from django.shortcuts import render
from django.urls import reverse_lazy
from django.views.generic.list import ListView
from django.views.generic.edit import CreateView, UpdateView, DeleteView

from .models import Course
from django.contrib.auth.mixins import LoginRequiredMixin, PermissionRequiredMixin

from django.shortcuts import redirect, get_object_or_404
from django.views.generic.base import TemplateResponseMixin, View
from .forms import ModuleFormSet
from django.forms.models import modelform_factory
from django.apps import apps
from .models import Module, Content
from braces.views import CsrfExemptMixin, JsonRequestResponseMixin
from django.db.models import Count
from .models import Subject
from django.views.generic.detail import DetailView

from students.forms import CourseEnrollForm

from django.core.cache import cache


class CourseDetailView(DetailView):
    model = Course
    template_name = 'courses/course/detail.html'

    def get_context_data(self, **kwargs):
        context = super(CourseDetailView, self).get_context_data(**kwargs)
        context['enroll_form'] = CourseEnrollForm(
        initial={'course':self.object})
        return context
```

```
class CourseListView(TemplateResponseMixin, View):
    model = Course
    template_name = 'courses/course/list.html'

    def get(self, request, subject=None):
        subjects = cache.get('all_subjects')
        if not subjects:
            subjects = Subject.objects.annotate(
                        total_courses=Count('courses'))
            cache.set('all_subjects', subjects)
        all_courses = Course.objects.annotate(
                            total_modules=Count('modules'))
        if subject:
            subject = get_object_or_404(Subject, slug=subject)
            key = 'subject_{}_courses'.format(subject.id)
            courses = cache.get(key)
            if not courses:
                courses = all_courses.filter(subject=subject)
                cache.set(key, courses)
        else:
            courses = cache.get('all_courses')
            if not courses:
                courses = all_courses
                cache.set('all_courses', courses)
        return self.render_to_response({'subjects': subjects,
                                        'subject': subject,
                                        'courses': courses})
class ContentOrderView(CsrfExemptMixin,
                       JsonRequestResponseMixin,
                       View):
    def post(self, request):
        for id, order in self.request_json.items():
            Content.objects.filter(id=id,
                       module__course__owner=request.user) \
                       .update(order=order)
        return self.render_json_response({'saved': 'OK'})
```

```
class ModuleOrderView(CsrfExemptMixin,
                      JsonRequestResponseMixin,
                      View):
    def post(self, request):
        for id, order in self.request_json.items():
            Module.objects.filter(id=id,
                course__owner=request.user).update(order=order)
        return self.render_json_response({'saved': 'OK'})

class ContentDeleteView(View):

    def post(self, request, id):
        content = get_object_or_404(Content,
                                    id=id,
                                    module__course__owner=request.user)
        module = content.module
        content.item.delete()
        content.delete()
        return redirect('module_content_list', module.id)



class ModuleContentListView(TemplateResponseMixin, View):
    template_name = 'courses/manage/module/content_list.html'

    def get(self, request, module_id):
        module = get_object_or_404(Module,
                                   id=module_id,
                                   course__owner=request.user)

        return self.render_to_response({'module': module})

class ContentCreateUpdateView(TemplateResponseMixin, View):
    module = None
    model = None
    obj = None
    template_name = 'courses/manage/content/form.html'
```

```
def get_model(self, model_name):
    if model_name in ['text', 'video', 'image', 'file']:
        return apps.get_model(app_label='courses',
                              model_name=model_name)
    return None

def get_form(self, model, *args, **kwargs):
    Form = modelform_factory(model, exclude=['owner',
                                             'order',
                                             'created',
                                             'updated'])
    return Form(*args, **kwargs)

def dispatch(self, request, module_id, model_name, id=None):
    self.module = get_object_or_404(Module,
                                    id=module_id,
                                    course__owner=request.user)
    self.model = self.get_model(model_name)
    if id:
        self.obj = get_object_or_404(self.model,
                                     id=id,
                                     owner=request.user)
    return super(ContentCreateUpdateView,
        self).dispatch(request, module_id, model_name, id)

def get(self, request, module_id, model_name, id=None):
    form = self.get_form(self.model, instance=self.obj)
    return self.render_to_response({'form': form,
                                    'object': self.obj})

def post(self, request, module_id, model_name, id=None):
    form = self.get_form(self.model,
                         instance=self.obj,
                         data=request.POST,
                         files=request.FILES)
    if form.is_valid():
        obj = form.save(commit=False)
```

```
            obj.owner = request.user
            obj.save()
            if not id:
                # new content
                Content.objects.create(module=self.module,
                                       item=obj)
            return redirect('module_content_list', self.module.id)

        return self.render_to_response({'form': form,
                                        'object': self.obj})

    def get(self, request, module_id, model_name, id=None):
        form = self.get_form(self.model, instance=self.obj)
        return self.render_to_response({'form': form, 'object': self.obj})

    def post(self, request, module_id, model_name, id=None):
        form = self.get_form(self.model,
                             instance=self.obj,
                             data=request.POST,
                             files=request.FILES)
        if form.is_valid():
            obj = form.save(commit=False)
            obj.owner = request.user
            obj.save()
            if not id:

                Content.objects.create(module=self.module, item=obj)
            return redirect('module_content_list', self.module.id)
        return self.render_to_response({'form': form, 'object': self.obj})


class CourseModuleUpdateView(TemplateResponseMixin, View):
    template_name = 'courses/manage/module/formset.html'
    course = None

    def get_formset(self, data=None):
        return ModuleFormSet(instance=self.course, data=data)
```

```
    def dispatch(self, request, pk):
        self.course = get_object_or_404(Course,
                                        id=pk,
                                        owner=request.user)
        return super(CourseModuleUpdateView, self).dispatch(request, pk)

    def get(self, request, *args, **kwargs):
        formset = self.get_formset()
        return self.render_to_response({'course': self.course,
                                        'formset': formset})

    def post(self, request, *args, **kwargs):
        formset = self.get_formset(data=request.POST)
        if formset.is_valid():
            formset.save()
            return redirect('manage_course_list')
        return self.render_to_response({'course': self.course,
                                        'formset': formset})

class OwnerMixin(object):
    def get_queryset(self):
        qs = super(OwnerMixin, self).get_queryset()
        return qs.filter(owner=self.request.user)

class OwnerEditMixin(object):
    def form_valid(self, form):
        form.instance.owner = self.request.user
        return super(OwnerEditMixin, self).form_valid(form)

class OwnerCourseMixin(OwnerMixin, LoginRequiredMixin):
    model = Course
    fields = ['subject', 'title', 'slug', 'overview']
    success_url = reverse_lazy('manage_course_list')

class OwnerCourseEditMixin(OwnerCourseMixin, OwnerEditMixin):
    fields = ['subject', 'title', 'slug', 'overview']
    success_url = reverse_lazy('manage_course_list')
    template_name = 'courses/manage/course/form.html'
```

```
class ManageCourseListView(OwnerCourseMixin, ListView):
    template_name = 'courses/manage/course/list.html'

class CourseCreateView(PermissionRequiredMixin,
                       OwnerCourseEditMixin,
                       CreateView):
    permission_required = 'courses.add_course'

class CourseUpdateView(PermissionRequiredMixin,
                       OwnerCourseEditMixin,
                       UpdateView):
    permission_required = 'courses.change_course'

class CourseDeleteView(PermissionRequiredMixin,
                       OwnerCourseMixin,
                       DeleteView):
    template_name = 'courses/manage/course/delete.html'
    success_url = reverse_lazy('manage_course_list')
    permission_required = 'courses.delete_course'

class ManageCourseListView(ListView):
    model = Course
    template_name = 'courses/manage/course/list.html'

    def get_queryset(self):
        qs = super(ManageCourseListView, self).get_queryset()
        return qs.filter(owner=self.request.user)
```