НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет
Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:
Завідувач кафедри
_____ Олександр КОВАЛЬ
«___»_____2021 р.

Дипломна робота

на здобуття ступеня бакалавра

спеціальності 122 «Комп'ютерні науки»

освітня програма «Комп'ютерний моніторинг та геометричне моделювання процесів і систем»

на тему: «Система обліку робіт студентів із можливістю віддаленої компіляції вихідного коду»

Виконав:
студент IV курсу, групи ТМ-71
Іваниця Євгеній Ігорович _____

Керівник:
Доцент, к.т.н.,
Ходаковський Олексій Володимирович _____

Консультант:
_____

Рецензент:
_____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ — 2021 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

спеціальності 122 «Комп’ютерні науки»

освітня програма «Комп’ютерний моніторинг та геометричне моделювання процесів і систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр КОВАЛЬ
(підпис)
”___”_____2021р.

ЗАВДАННЯ
на дипломну роботу студенту

Іваниці Євгенію Ігоровичу
(прізвище, ім’я, по батькові)

1. Тема роботи _Система обліку робіт студентів із можливістю віддаленої компіляції вихідного коду_____

керівник роботи____Ходаковський Олексій Володимирович, к.т.н., доцент___
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”24” травня 2021р. № **1267-с**

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи ___мови програмування Python та JavaScript, фреймворк Django, середовище розробки PyCharm, СУБД SQLite, мова розмітки гіпертексту HTML, каскадні таблиці стилів CSS._____

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) __проаналізувати існуючі системи управління навчанням та обліку робіт студентів; розробити веб-платформу для обліку робіт студентів із можливістю віддаленої компіляції вихідного коду засобами вбудованого онлайн-компілятора.

5. Перелік ілюстративного матеріалу

Постановка задачі, діаграма прецедентів системи, модель бази даних, засоби розробки, сторінки реєстрації та авторизації, модуль викладача, модуль студента, модуль адміністрування, модуль онлайн-компілятора, висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання ”___” _____2020 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	15.10.2020	
2.	Вивчення та аналіз задачі	16.10.2020-16.12.2020	
3.	Розробка архітектури та загальної структури системи	09.03.2021-23.03.2021	
4.	Розробка структур окремих підсистем	24.03.2021-10.04.2021	
5.	Програмна реалізація системи	14.04.2021-10.05.2021	
6.	Оформлення пояснювальної записки	18.05.2021-02.06.2021	
7.	Захист програмного продукту	17.05.2021	
8.	Передзахист	26.05.2021	
9.	Захист	16.06.2021	

Студент _____ _____Іваниця Є. І._____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ __Ходаковський О. В._____
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

У роботі розглянуті теоретичні та практичні аспекти розробки системи обліку робіт студентів із можливістю віддаленої компіляції вихідного коду.

Метою роботи є створення веб-платформи для надсилання та обліку робіт студентів із можливістю віддаленої компіляції вихідного коду засобами вбудованого онлайн-компілятора.

В результаті було розроблено систему, що складається із модулів студента, викладача, адміністрування та онлайн-компілятора. Всі модулі системи було протестовано в умовах, максимально наближених до реального навчального процесу.

Пояснювальна записка складається зі вступу, п’яти розділів, висновку, списку використаних джерел; містить 50 сторінок, 20 рисунків та 3 додатки. Список використаних джерел включає 1 бібліографічних найменувань.

Ключові слова: веб-платформа, онлайн-компілятор, облік робіт, студент, викладач, курс, завдання, рішення.

ABSTRACT

The paper considers theoretical and practical aspects of developing a system of accounting for students' work with the possibility of remote compilation of source code.

The aim of the work is to create a web platform for sending and accounting of students' works with the possibility of remote compilation of source code by means of a built-in online compiler.

As a result, a system consisting of student, teacher, administration and online compiler modules was developed. All modules of the system were tested in conditions as close as possible to the real learning process.

The explanatory note consists of an introduction, five sections, a conclusion, a list of sources used; contains 50 pages, 20 figures and 3 appendices. The list of used sources includes 14 bibliographic names.

Keywords: web platform, online compiler, work accounting, student, teacher, course, tasks, solutions.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

JS — JavaScript

CSS — Cascading Style Sheets (Каскадні таблиці стилів)

HTML — HyperText Markup Language (Мова розмітки гіпертексту)

LMS — Learning management system (Система управління навчальною діяльністю)

UML — Unified Modeling Language (Уніфікована мова моделювання)

IDE — Integrated development environment (Інтегроване середовище розробки)

ВСТУП

Проблема ефективної організації навчального процесу у зручному форматі є надзвичайно актуальною в контексті стрімкого поширення технологій та автоматизації процесів у всіх сферах діяльності людства. Ефективна організація навчання та здачі робіт учнями чи студентами здатна зекономити велику кількість часу та значно спростити виконання організаційних моментів навчальної системи як для учнів та студентів, так і для викладачів.

Задля подолання вищеописаної проблеми, на сьогоднішній день розробляються сотні систем управлення навчанням. Система управління навчанням – це програмна платформа, призначена для створення, управління та поширення курсів, завдань та програм електронного навчання. Системи управління навчанням призначені не лише для навчання в Інтернеті. Систему управління вмістом для управління та відстеження всіх видів навчання можна використовувати і офлайн. Подібні системи використовуються як в навчальних закладах вищої та середньої освіти, так і в корпоративних цілях для навчання співробітників [1].

Системи управління навчанням вперше використовувались у вищих навчальних закладах, але стали популярними в інших сферах, включаючи корпоративне навчання. Вони допомагають зростаючій кількості навчальних закладів та організацій навчати та керувати своїми студентами за допомогою моделі, яка не вимагає фізичної участі.

Кількість інформації, яка може бути створена та поширена за допомогою системи управління навчанням, величезна. Порівняно із традиційними способами навчання, системи управління навчанням можуть також включати в себе відео матеріали, веб-семінари та інтерактивні елементи для кращого закріплення матеріалу студентами [2].

Системи управління навчанням повинні бути інтуїтивно зрозумілими, простими у використанні та здатними до масштабування. Однак, це і є головними недоліками багатьох існуючих систем.

Основною метою даної роботи є розробка системи обліку робіт студентів, яка б дозволяла додатково компілювати та запускати вихідні коди програм безпосередньо засобами у самої системі. Така система значно спрощує процес здачі та перевірки робіт, крім того, можливість віддалено компілювати вихідні коди програм буде актуальною для дисциплін, безпосередньо пов’язаних із програмуванням, які набувають все більшої і більшої популярності у зв’язку із зростанням попиту на кадри у сфері інформаційних технологій та програмування. Вищеописаний механізм здатний зекономити значну кількість часу як для учнів та студентів, так і для викладачів [3].

1. ЗАДАЧА СИСТЕМИ ОБЛІКУ РОБІТ СТУДЕНТІВ ІЗ МОЖЛИВІСТЮ ВІДДАЛЕНОЇ КОМПІЛЯЦІЇ ВИХІДНИХ КОДІВ ПРОГРАМ

Метою даної роботи є створення онлайн-платформи, яка дозволятиме проводити здачу та облік рішень студентів до задач, які їм ставить викладач. Додатковою задачею роботи є розробка відокремленого модуля системи – онлайн-компілятора, який може використовуватись для компіляції вихідних кодів програм, які будуть надсилатись студентами як частина рішення завдання.

Система розроблятиметься переважно мовою програмування Python із застосуванням фреймворку Django і буде складатись із клієнтського модуля викладача, клієнтського модуля студента, системи адміністрування, модуля онлайн-компілятора, веб-сервера, серверного додатку та бази даних.

Перелік задач, які необхідно буде розв’язати для досягнення поставленої мети даної роботи:

1. Провести аналіз існуючих рішень, які реалізують системи управління навчанням, проаналізувати їх функціонал та можливості, визначити основні технічні переваги та недоліки кожної системи.
2. Створити деталізовану модель майбутньої системи обліку робіт студентів із можливістю віддаленої компіляції вихідних кодів програм у вигляді схем та діаграм UML.
3. Визначити найбільш підходящі технічні засоби та технології розробки системи.
4. Розробити всі модулі системи, базуючись на створених моделях та обраних засобах розробки.
5. Провести тестування системи.

Клієнтський модуль викладача повинен реалізовувати наступні можливості в рамках системи:

- Створення, перегляд, редагування та видалення курсів (предметів);
- Створення, перегляд, редагування та видалення завдань, які викладач створює для окремого курсу;
- Наповнення завдань контентом: текстом, зображеннями, відео, або іншими файлами, які б дозволяли зручно переглянути весь матеріал по конкретній темі та виконати відповідне завдання студентам;
- Перегляд рішень студентів по кожному завданню.

Необхідно створити клієнтський модуль студента, який дозволяв би виконувати наступні дії у системі:

- Перегляд інформації про курс (предмет);
- Перегляд інформації та контенту по завданням для кожного курсу;
- Створення, перегляд, редагування та видалення рішень по кожному завданню;
- Додавання контенту із описом до кожного рішення: супроводжувальний текст, зображення, відео, файли будь-якого формату, вихідні коди програм для того, аби сформовані рішення підходили до завдань будь-якого формату.

Наступний модуль – система адміністрування. У цьому модулі необхідно реалізувати наступні можливості:

- Перегляд, редагування та видалення будь-яких даних в системі (курсів, завдань, рішень);
- Додавання, редагування та видалення користувачів системи;
- Встановлення та вилучення прав доступу (студент, викладач, адміністратор) для зареєстрованих користувачів системи;
- Перегляд історії змін у модулі адміністрування.

Онлайн-компілятор повинен працювати окремим відокремлений модулем у системі і реалізовувати наступний функціонал :

- Компіляція та виконання вихідних кодів програм, написаних основними мовами програмування (C, C++, C#, Go, Java, JavaScript, Pascal, PHP, Python);
- Користувацький ввід даних перед виконанням програми;
- Вивід результатів компіляції;
- Вбудований текстовий редактор для написання вихідних кодів програм безпосередньо у модулі онлайн-компілятора;
- Зміна теми текстового редактора (світла і темна);
- Завантаження написаних вихідних кодів на пристрій користувача;

Вищеописані користувацькі модулі повинні бути реалізовані окремими додатками фреймворку Django. Інтерфейс системи реалізовуватиметься засобами HTML, CSS, JavaScript. Серверна частина – мовою програмування Python на основі фреймворку Django та із використанням СУБД SQLite.

2. ОПИС АНАЛОГІЧНИХ СИСТЕМ У СФЕРІ УПРАВЛІННЯ НАВЧАЛЬНИМ ПРОЦЕСОМ

На сьогодні існує дуже велика кількість систем обліку робіт студентів та систем управління навчанням. Перш за все, це пов’язано зі зростанням попиту на дистанційну освіту та високою ефективністю подібних систем у освітній сфері. Але у кожної із них є свої недоліки та особливості, які не підходять для всіх напрямків освіти.

2.1. Система управління навчанням Moodle

Moodle – це система управління навчанням з відкритим кодом (LMS), яка забезпечує власне навчальне середовище для студентів та учнів. Викладачі можуть використовувати Moodle для створення уроків, управління курсами та взаємодії з викладачами та студентами. Студенти можуть використовувати Moodle для перегляду календаря занять, здачі завдань, проходження тестів.

Moodle використовується тисячами навчальних закладів по всьому світу для забезпечення організованого та централізованого інтерфейсу для електронного навчання. Викладачі та адміністратори можуть створювати віртуальні класи та керувати ними, в яких студенти можуть отримувати доступ до відео, документів та тестів. Груповий чат дозволяє студентам спілкуватися з викладачем та іншими студентами в безпечному середовищі.

Для створення навчального середовища Moodle програмне забезпечення Moodle потрібно завантажити та встановити на веб-сервері. Платформа Moodle є відкритим кодом і побудована з використанням модульної конструкції. Індивідуальні користувачі, такі як викладачі та студенти, можуть зареєструвати обліковий запис на

сервері Moodle і отримати доступ до вмісту через веб-інтерфейс або програму "Moodle Desktop".

Moodle є прикладом LAMP додатку. LAMP спочатку розшифровувався як Linux, Apache, MySQL та Perl. З часом різні компоненти абревіатури змінилися. Наприклад, PHP став основною мовою для додатків LAMP. Однак заголовок закріпився, посилаючись на програми, написані мовами веб-сценаріїв, використовуючи базу даних SQL для зберігання інформації [4].

Переваги системи управління навчанням Moodle:

- Moodle має зручний інтерфейс, це означає, що для ознайомлення із системою, студентам та викладачам потрібно витратити мінімум часу.
- Підтримує всі новітні стандарти електронного навчання, такі як SCORM та Tin Can / xAPI. Крім того, дозволяє завантажувати існуючі документи та відео, а також обмінюватися матеріалами та подіями між курсами або навчальними шляхами.
- Окрім надання функцій в Інтернеті, платформа здатна підтримувати навчання в режимі офлайн і включає функції для відстеження, запису та оцінки офлайн-подій разом із онлайновими.
- Може підтримувати тести та письмові роботи як частину навчального процесу.
- Включає функції для відстеження та фіксації прогресу учнів через навчальний шлях.
- Moodle включає не лише записи та особисті дані тих, хто навчається, але також і контент, який має власний характер або має комерційну цінність. Moodle LMS має функцію безпечного входу для захисту конфіденційних даних.
- Платформа є досить гнучкою в налаштуванні та багатофункціональною. На сьогодні світовою спільнотою розроблено більш ніж 500 плагінів для Moodle.

Недоліки та проблеми:

- Перша велика проблема полягає в тому, що Moodle ще не досить допрацьований, щоб справлятися з великими проектами. Система може не працювати ефективно з великими школами або служити способом проведення всіх занять у місті.
- Існують досить великі проблеми зі стабільністю роботи системи. Веб-сайт може іноді закриватися, або працювати зі збоями, доставляючи проблеми та незручності студентам та викладачам.
- Зазвичай для управління та обслуговування функціональних можливостей системи потрібна команда технічних працівників, яка виконуватиме оновлення функцій, виправлення помилок, обслуговування серверів, зберігання та безпеку.
- Управління користувачами системи може бути досить великою проблемою.

2.2. Система управління навчанням Google Classroom

Google Classroom – це система управління навчанням, яка належить до пакету Google Apps for Education, метою якого є сприяння цифровому навчанню в класі.

За допомогою цієї платформи вчителі можуть створити власний онлайн-клас, залучити учнів до участі у їхньому класі та роздати домашнє завдання. Викладачі та студенти можуть обговорити свої домашні завдання на платформі. Вчителі можуть легко контролювати успіхи кожного учня.

Школи також можуть створити власний обліковий запис Google Apps for Education, щоб повністю використовувати це рішення. Inbox by Gmail може розміщувати повідомлення в Google Classroom, дозволяючи як викладачам, так і студентам визначати критичні оновлення та основні моменти. Вчителі можуть структурувати потоки класів, включаючи додаткові предмети в посади. Вчителі та

студенти отримують свободу фільтрувати потоки, щоб вони могли шукати певні предмети. Google Classroom – це велика перевага для батьків. Вчителі можуть надіслати батькам короткий опис результатів своєї дитини. Більше того, батьки можуть отримувати у своєму електронному листі автоматизовані зведення про роботу учнівських класів та інші важливі оголошення [5].

Переваги платформи Google Classroom:

- Проста у використанні та доступна з усіх пристроїв.
- Однією з найбільших переваг Google Classroom є Google Docs. Ці документи зберігаються в Інтернеті та передаються необмеженій кількості людей, тож при створенні оголошення або завдання за допомогою документа Google, ваші студенти можуть негайно отримати доступ до них через свій Google Drive.
- Google Classroom дає можливість запропонувати свою онлайн-підтримку учням.
- Чистий та зручний інтерфейс, кожна окрема деталь дизайну проста, інтуїтивна та зручна для користувача.
- Студенти та учні можуть коментувати конкретні місця в межах фотографій для різноманітних онлайн-курсів. Крім того, ви можете створювати URL-адреси для цікавих коментарів та використовувати їх для подальшого обговорення в Інтернеті.
- Викладачі також можуть приєднатися до Google Classroom як учні, що означає, що можна створити Google Classroom і використовувати його для зустрічей викладачів, обміну інформацією або професійного розвитку.

Недоліки та проблеми:

- Налаштування класу може бути простим, але для кожного учня необхідно створювати свій аккаунт. Для того, щоб зробити клас повністю приватним, ви повинні створити нові облікові записи, зокрема для себе, через домен Google Classroom.

- Через закрите та приватне середовище важко або неможливо ділитися предметами з широким шкільним співтовариством, громадськістю чи батьками.
- Незважаючи на те, що він реєструє оцінки учнів, Google Classroom не містить повноцінної книги з оцінками. Можна лише експортувати Google Таблиці до універсальних файлів баз даних, сумісних з іншими програмами книг класів.

2.3. Система управління навчанням Docebo

Docebo – постачальник хмарних рішень для електронного навчання, система управління навчанням SaaS (LMS). Він пропонує екосистему модулів та функціональних можливостей, які можуть бути адаптовані до різних середовищ. Docebo підтримує багато популярних методів навчання, включаючи мобільні, соціальні та змішані навчальні ініціативи [6].

Переваги системи управління навчанням Docebo:

- Простота завантаження та перетворення файлів.
- Docebo дозволяє завантажувати навчальні матеріали та ділитися ними у різних форматах, включаючи файли PDT, PPT, відео, SCORM та Tin Can. Файли PPT, PDF, ODP та PPTX можна конвертувати у слайди або послідовність знімків. Пакети SCORM створюються за допомогою спеціального програмного забезпечення та завантажуються у форматі ZIP. Docebo також перетворює відеофайли у формат MP4.
- Інтеграція з популярним програмним забезпеченням. Для веб-конференцій можна інтегруватися з Teleskill, WebEx та Adobe Connect, для Системи управління вмістом (CMS) існують WordPress, Drupal та Joomla, для соціальних мереж – LinkedIn, Facebook та Twitter, а для взаємодії з клієнтами в Інтернеті Docebo легко інтегрується з Vivocha.

- Функції єдиного входу (SSO) та функції єдиного входу зі сторонніми програмами.
- Продаж курсів за допомогою функцій електронної комерції Docebo.

Недоліки та проблеми:

- Не досить зручний інтерфейс, який не завжди буде зрозумілим новим користувачам.
- Висока складність в адмініструванні та обслуговуванні системи.
- Відсутність розширеного API, який надавав би доступ до всіх об'єктів на стороні LMS та маніпулювання ними.
- У стандартному пакеті системи включено досить мало функціональних можливостей.
- Значна кількість помилок та недоліків, які не виправляються дуже довго, що свідчить про погану підтримку системи.

3. ЗАСОБИ РЕАЛІЗАЦІЇ ПРОГРАМНОЇ СИСТЕМИ ОБЛІКУ РОБІТ СТУДЕНТІВ ІЗ МОЖЛИВІСТЮ ВІДДАЛЕНОЇ КОМПІЛЯЦІЇ ВИХІДНОГО КОДУ

3.1. Вибір технологій та їх обґрунтування

Основою системи є архітектура фреймворку Django мови програмування Python. Django – це фреймворк з відкритим кодом для серверних веб-додатків на основі Python – однієї з найкращих мов веб-розробки. Основними його перевагами є простота, гнучкість, надійність та масштабованість.

База даних веб-платформи керується системою управління базами даних SQLite, яка реалізує швидкий, автономний, високонадійний, повнофункціональний механізм баз даних SQL.

Інтерфейс користувача розроблено засобами HTML, CSS та меншою мірою мови програмування JavaScript.

Розробка повністю велась у спеціальному інтегрованому середовищі розробки Python – PyCharm.

3.2. Мова програмування Python

Python – інтерпретована, об’єктно-орієнтована мова програмування високого рівня з динамічною семантикою. Високорівневі вбудовані структури даних у поєднанні з динамічним набором тексту та динамічним прив'язуванням роблять його дуже привабливим для швидкої розробки додатків, а також для використання в якості мови сценаріїв або для з'єднання існуючих компонентів.

Простий, легкий у вивченні синтаксис Python підкреслює читабельність і, отже, зменшує витрати на обслуговування програми. Python підтримує модулі та пакети, що заохочує модульність програми та повторне використання коду. Інтерпретатор Python та велика стандартна бібліотека доступні у вихідній або двійковій формі безкоштовно для всіх основних платформ і можуть вільно поширюватися [7].

Інтерпретатор Python легко розширити за допомогою нових функцій та типів даних, реалізованих на C або C ++ (або інших мовах, які можна викликати з C). Python також підходить як мова розширення для налаштовуваних додатків.

Python можна використовувати для створення веб-додатків на стороні сервера. Для цього може використовуватись як чистий Python, так і велика кількість вже існуючих веб-фреймворків на Python.

Python не використовується у веб-браузері. Мовою, яка виконується у таких браузерах, як Chrome, Firefox, Opera та Internet Explorer, є JavaScript. Найкращим варіантом у такій ситуації є використання комбінацію Python та JavaScript. Python виконується на стороні сервера, а JavaScript завантажується в клієнт і запускається веб-браузером.

3.3. Веб-фреймворк Django

Django – це фреймворк веб-додатків з відкритим кодом, який написаний мовою Python. Цей фреймворк використовується для побудови внутрішньої (серверної) частини веб-програми. Початкова версія Django випущена 15 липня 2005 року компанією Django Software Foundation.

Основними перевагами Django є максимально спрощене створення веб-додатків, керованих базами даних. Структура фреймворку підкреслює багаторазовість та автономність її компонентів, менше коду, швидкий розвиток проекту та принцип "не повторюватися" [8].

Django використовує архітектуру MTV (Model-Template-View). MVT, як правило, дуже схожий на MVC (Model-View-Controller). Різниця між MVC та MVT тут полягає в тому, що сам Django виконує роботу контролера в архітектурі MVC за допомогою шаблонів.

На рисунку 3.3 подано загальну структуру роботи системи на основі Django.

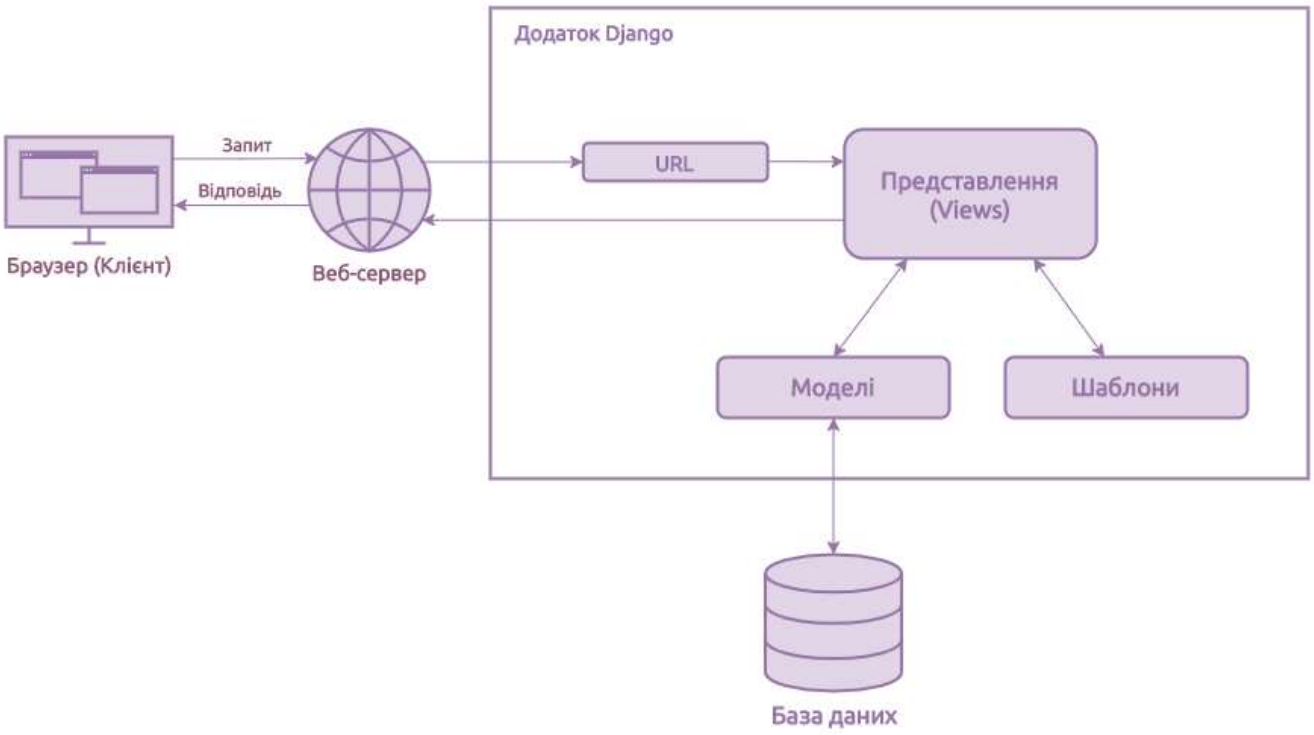

Рисунок 3.3 – Загальна схема роботи системи із додатком Django

На рисунку 3.3 показано робочий цикл архітектури Django MVC (Model-View-Controller).

Відповідно до схеми, модель – це частина програми, яка виступає посередником між інтерфейсом веб-сайту та базою даних. Модель є структурою даних, що стоїть за цілим додатком, і представлена базою даних. Модель – це компонент, який є частиною бізнес-логіки в архітектурі Django.

Представлення – це компонент, який містить логіку інтерфейсу в архітектурі Django. представлення насправді є інтерфейсом користувача програми. Вони представлені файлами HTML/ CSS/ Javascript. Як правило, вміст надходить з компонента моделей.

Контролер, як випливає з назви, є основним компонентом управління. Це означає, що контролер обробляє взаємодію користувача та вибирає представлення відповідно до моделі. Основне завдання контролера – вибрати компонент подання відповідно до взаємодії користувача, а також застосувати компонент моделі [9].

Django – це, в основному, платформа MTV (Model-Template-View). Вона використовує шаблони для представлень та контролера. Шаблон відноситься до представлення у архітектурі MVC та, по суті, контролює вміст, який відображатиметься, та те, як він відображатиметься для користувача.

Таким чином, код Python буде у поданнях та моделях, а HTML/ CSS/ Javascript – у шаблонах.

3.4. Мова програмування JavaScript

JavaScript – це динамічна мова програмування, яка найчастіше використовується як частина веб-сторінок, реалізації яких дозволяють клієнтському сценарію взаємодіяти з користувачем та створювати динамічні сторінки. Це інтерпретована мова програмування з об’єктно-орієнтованими можливостями.

JavaScript, в силу своєї легкості при виконанні, найчастіше використовується у клієнтській частині додатків. Сценарій повинен бути включений у документ HTML або посилатися на нього, щоб код інтерпретувався браузером. Це означає, що веб-сторінка не повинна бути статичним HTML, але може включати програми, які взаємодіють з користувачем, керують браузером та динамічно створюють вміст HTML [10].

Клієнтський механізм JavaScript надає багато переваг перед традиційними серверними сценаріями CGI. JavaScript можна використовувати для захоплення ініційованих користувачем подій, таких як клацання кнопок, навігація посиланнями та інші дії, які користувач ініціює явно або неявно.

3.5. Середовище розробки PyCharm

PyCharm – це спеціальне інтегроване середовище розробки Python (IDE), що забезпечує широкий спектр важливих інструментів для розробників Python. Розроблений чеською компанією JetBrains (раніше відомою як IntelliJ).

Забезпечує аналіз коду, графічний налагоджувач, інтегрований тестер одиниць, інтеграцію із системами контролю версій (VCSes).

Крім того, PyCharm забезпечує розробку засобами Django, Flask та Pyramid. Крім того, він повністю підтримує HTML (включаючи HTML5), CSS, JavaScript та XML: ці мови входять до складу IDE за допомогою плагінів і вмикаються за замовчуванням. Підтримку для інших мов та фреймворків також можна додати за допомогою плагінів.

PyCharm – це крос-платформне середовище розробки, яке працює на Windows, macOS та Linux.

3.6. Мова розмітки гіпертексту HTML

HTML – це абревіатура, що розшифровується як Hyper Text Markup Language, яка використовується для створення веб-сторінок та веб-програм.

HyperText означає просто "текст у тексті". Текст, що має в собі посилання, є гіпертекстом. HyperText – це спосіб зв’язування двох або більше веб-сторінок (HTML-документів) між собою.

Мова розмітки – це комп’ютерна мова, яка використовується для застосування конвенцій щодо розмітки та форматування до текстового документа. Мова розмітки робить текст більш інтерактивним та динамічним. Він може перетворювати текст на зображення, таблиці, посилання тощо [11].

Отже, HTML – це мова розмітки, яка використовується для створення привабливих веб-сторінок за допомогою стилю і яка виглядає у приємному форматі у веб-браузері. Документ HTML складається з безлічі тегів HTML, і кожен тег HTML містить різний вміст.

3.7. Каскадні таблиці стилів CSS

Каскадні таблиці стилів (CSS) – це мова таблиці стилів, яка використовується для опису зовнішнього вигляду та форматування документа, написаного мовою розмітки.

Зазвичай він використовується з HTML для зміни стилю веб-сторінок та користувацьких інтерфейсів. Він також може використовуватися з будь-якими документами XML, включаючи звичайні XML, SVG та XUL.

CSS використовується разом з HTML та JavaScript на більшості веб-сайтів для створення інтерфейсів для веб-програм та інтерфейсів користувачів для багатьох мобільних додатків.

Визначення стилів CSS зберігаються у зовнішніх файлах CSS, тому можна змінити весь веб-сайт, змінивши лише один файл.

CSS надає більш детальні атрибути, ніж звичайний HTML, для визначення зовнішнього вигляду веб-сайту.

До CSS теги, такі як шрифт, колір, стиль фону, вирівнювання елементів, межі та розміри, повинні були повторюватися на кожній веб-сторінці. Це дуже сповільнювало швидкодію завантаження сайтів. Для вирішення цієї проблеми був створений CSS [12].

Рекомендації CSS постійно переглядаються та еволюціонують, щоб задовольнити запити та вимоги зростаючої аудиторії під керівництвом W3C, організації, відповідальної за розробку та підтримку мови.

3.8. Система управління базами даних SQLite

SQLite – це реляційна система управління базами даних (СУБД), що міститься в бібліотеці C і реалізує швидкий, автономний, високонадійний, повнофункціональний механізм баз даних SQL.

SQLite – це найбільш використовуваний механізм баз даних у світі. Він вбудований у всі мобільні телефони та більшість комп’ютерів і входить до складу безлічі інших програм, якими люди користуються щодня.

Формат файлу SQLite стабільний, крос-платформний і зворотно сумісний. Файли бази даних зазвичай використовуються як контейнери для передачі багатого вмісту між системами та як довгостроковий архівний формат даних.

Вихідний код SQLite знаходиться у загальному доступі і може використовуватись будь-яких цілей.

SQLite реалізує більшість стандартів SQL, як правило, дотримуючись синтаксису PostgreSQL [13].

SQLite – популярний вибір як програмне забезпечення для вбудованих баз даних для локального/ клієнтського зберігання в прикладних програмах, таких як веб-браузери. Це, мабуть, найбільш широко розгортаний механізм баз даних, оскільки він використовується серед багатьох широко поширених браузерів, операційних систем та вбудованих систем. SQLite має прив'язки до багатьох мов програмування, в тому числі, SQLite є СУБД за замовчуванням у Django.

3.9. Висновки до розділу

У даному розділі описано мови програмування та технології, використані при розробці веб-платформи для здачі та обліку робіт студентів із можливістю віддаленої компіляції вихідних кодів програм засобами вбудованого онлайн-компілятора.

За основу системи було взято фреймворк Django мови програмування Python. Основними його перевагами є простота, гнучкість, надійність та масштабованість при розробці веб-додатків.

Для керування базою даних системи було обрано систему управління базами даних SQLite, яка реалізує швидкий, автономний, високонадійний, повнофункціональний механізм баз даних SQL.

Для створення інтерфейсу користувача обрано стандартну зв’язку засобів HTML, CSS та мови програмування JavaScript.

В якості IDE було обрано PyCharm – спеціальне інтегроване середовище мови програмування Python із величезним спектром інструментів для розробки.

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

4.1. Структура проекту

Система обліку робіт студентів складається із користувацького веб-додатку (модуль викладача, модуль студента, модуль адміністратора, модуль онлайн-компілятора), веб-сервера, серверного додатка Django та бази даних.

Основою системи є серверний додаток Django, в якому реалізована вся логіка роботи системи, взаємодія із базою даних через моделі, логіка роботи інтерфейсу через представлення, зберігаються шаблони для користувацького веб-додатку кожного із модулів. На рисунку 4.1 зображено узагальнену структуру файлової системи додатку Django.

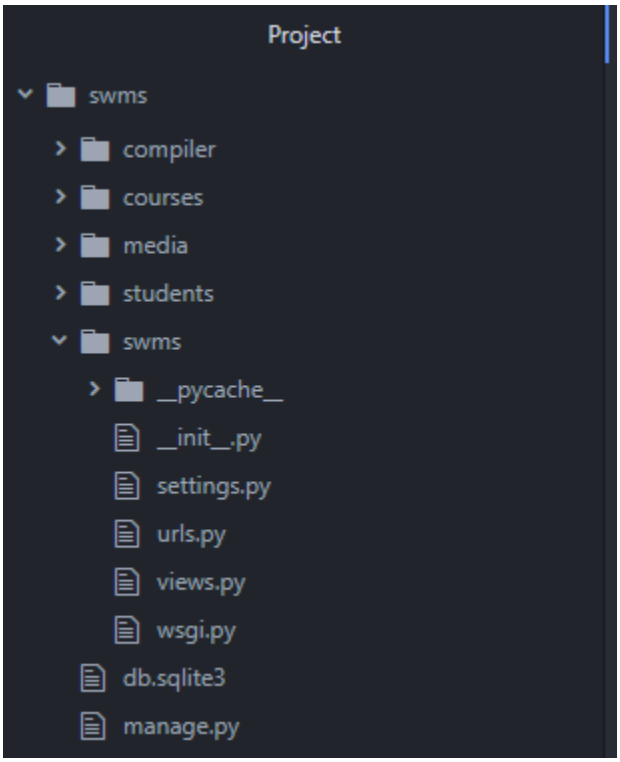

Рисунок 4.1 – Узагальнена структура файлової системи серверного додатку Django

Модуль swms є головною частиною всього додатку. Він включає в себе всі глобальні налаштування (settings.py), URL-адреси додатка (urls.py) та представлення (views.py).

Модуль compiler реалізує відокремлений додаток онлайн-компілятора. Безпосередньо компіляція виконується засобами API HackerEarth.

Модуль courses повністю реалізує весь механізм функціонування курсів, завдань, рішень та контенту в ньому, тобто є так званою системою управління контентом (модуль викладача та адміністратора). Крім того, у цьому модулі здійснена реалізація всіх моделей (models.py) додатку.

Модуль students реалізує модуль студента.

Тека media використовується для зберігання усього контенту, який завантажується на сервер.

Файл db.sqlite3 – файл бази даних.

Файл manage.py – утиліта командного рядка, яка дозволяє взаємодіяти з проектом Django різними способами.

4.2. Діаграма прецедентів

Діаграма прецедентів (варіантів використання) – це діаграма поведінки мови моделювання UML. Діаграми варіантів використання моделюють функціональність системи за допомогою акторів та варіантів використання [14].

Варіанти використання - це набір дій, послуг та функцій, які система повинна виконувати. У цьому контексті "система" – це щось, що розробляється або експлуатується, наприклад, веб-сайт.

"Актори" – це люди або організації, що діють за певною роллю в системі. Стандартні види відносин між акторами і подіями:

- асоціації (association);
- розширення (extend);

- узагальнення (generalization);
- включення (include).

Діаграма прецедентів розробленого застосунку зображена на рисунку 4.2.

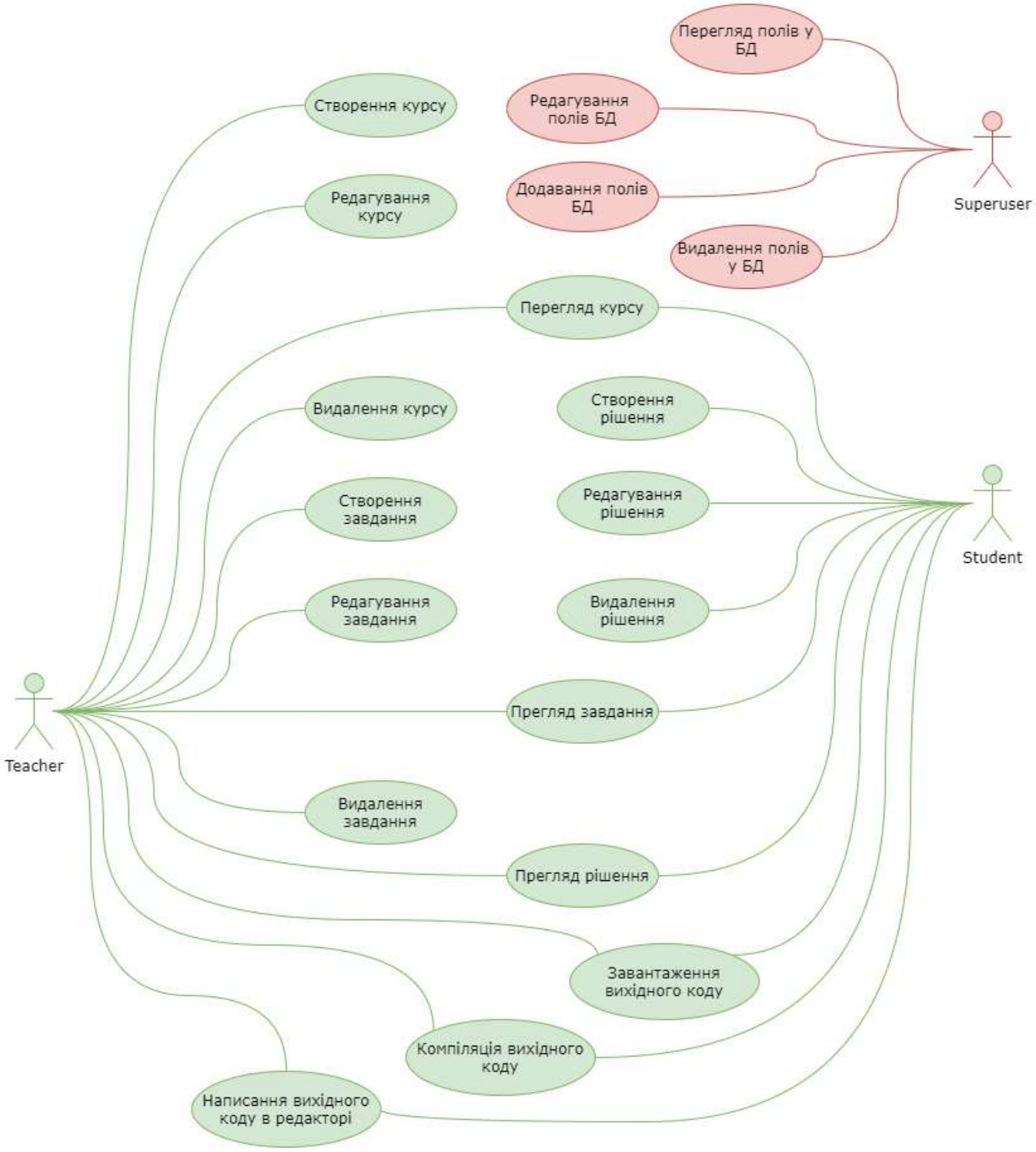

Рисунок 4.2 – Діаграма прецедентів системи

На діаграмі (рисунок 4.2) видно, що акторами системи є студент (student), викладач (teacher) та адміністратор (superuser).

Червоним кольором виділено прецеденти та відношення, які відносяться до відокремленої підсистеми адміністрування, а зеленим – прецеденти та відношення, які відносяться до користувацького модуля викладача, модуля студента та відокремленого модуля онлайн-компілятора.

4.3. Опис структури бази даних

Для зберігання даних про користувачів курси, завдання, рішення та контент розроблена веб-платформа використовує реляційну базу даних, створену на базі системи управління базами даних SQLite засобами Python веб-фреймворку Django. Для комунікації із базою даних Django використовує засоби моделей та представлень архітектури MTV, яка є основою фреймворку.

База даних складається із 10 таблиць:

- Курс (Course) – містить інформацію про курс (ідентифікатор курсу, викладач, назва, слоган, опис, дата і час створення, студенти курсу) та зв’язане із таблицею користувача (User) відношеннями “один до одного” (викладач) та “один до багатьох” (студенти).
- Завдання (Task) – включає інформацію про завдання (ідентифікатор завдання, курс, назва, опис, дата і час створення) та зв’язане із відповідним курсом відношенням “один до багатьох”.
- Рішення (Solution) – містить інформацію про рішення (ідентифікатор, виконавець, завдання, дата і час створення) та зв’язане із відповідним завданням відношенням “один до багатьох”.
- Вміст (Content) – містить інформацію про вміст, яким наповнюються завдання і рішення.
- Користувач (User) – описує користувача системи.

- Вихідний код (SourceCode) – описує вихідний код програми як формат вмісту.
- Текст (Text) – описує текст як формат вмісту.
- Відео (Video) – описує відео файл як формат вмісту.
- Зображення (Image) – описує зображення як формат вмісту.
- Файл (File) – описує файл (будь-якого формату даних) як формат вмісту.

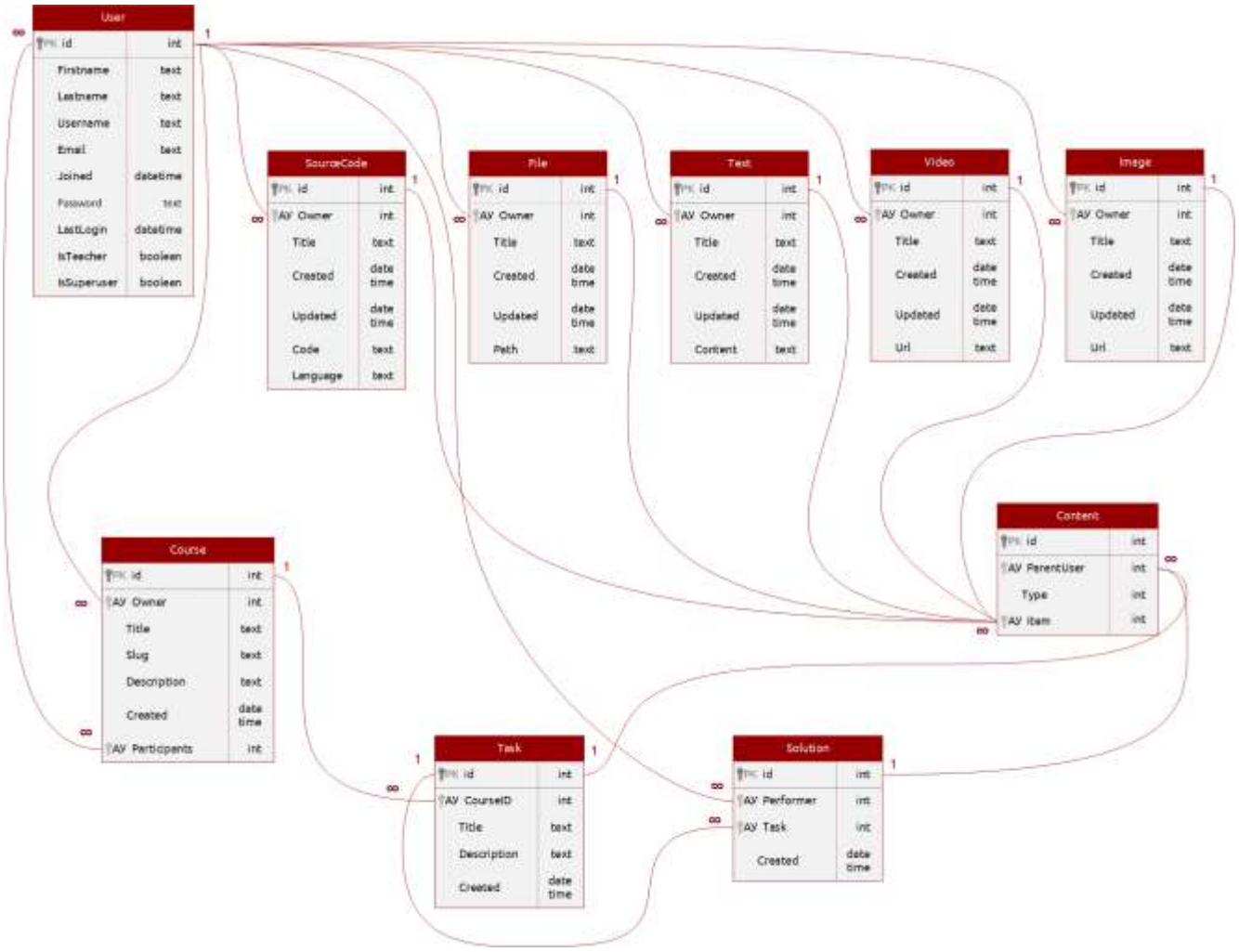

Рисунок 4.3 – Схема бази даних системи

На рисунку 4.3 зображено повну схему бази даних всього проекту. Відповідними позначеннями на схемі зображені відношення, їх типи та ключові поля таблиць.

4.4. Висновки до розділу

У даному розділі описано архітектуру веб-платформи, файлову структуру програмного забезпечення у розгорнутому вигляді, діаграму прецедентів (варіантів використання) системи, що описує набір дій, послуг та функцій, які система повинна виконувати та базу даних системи із описом всіх десяти її полів та засобів, що були використані при розробці.

5. МЕТОДИКА РОБОТИ КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

5.1. Системні вимоги

Оскільки створена система є веб-додатком, то для повноцінної роботи з усіма її користувацькими модулями необхідно лише встановити будь-який доступний браузер незалежно від використовуваного пристрою та мати стабільне підключення до мережі Інтернет.

5.2. Робота із модулем викладача

Розроблений Для роботи у модулі викладача, необхідно авторизуватись у системі. Сторінка авторизації зображена на рисунку 5.1.1.

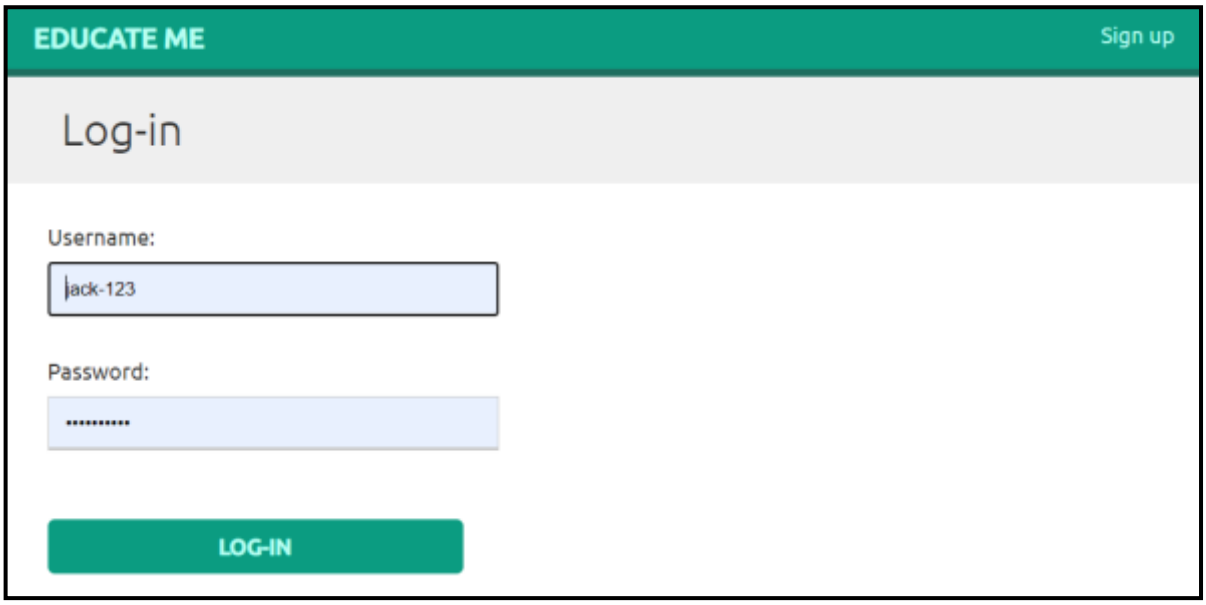

Рисунок 5.1.1 – Сторінка авторизації

Після авторизації, якщо поточний користувач має права викладача, відбудеться автоматичне перенаправлення у модуль, де буде відображено список усіх курсів викладача (рисунок 5.1.2).

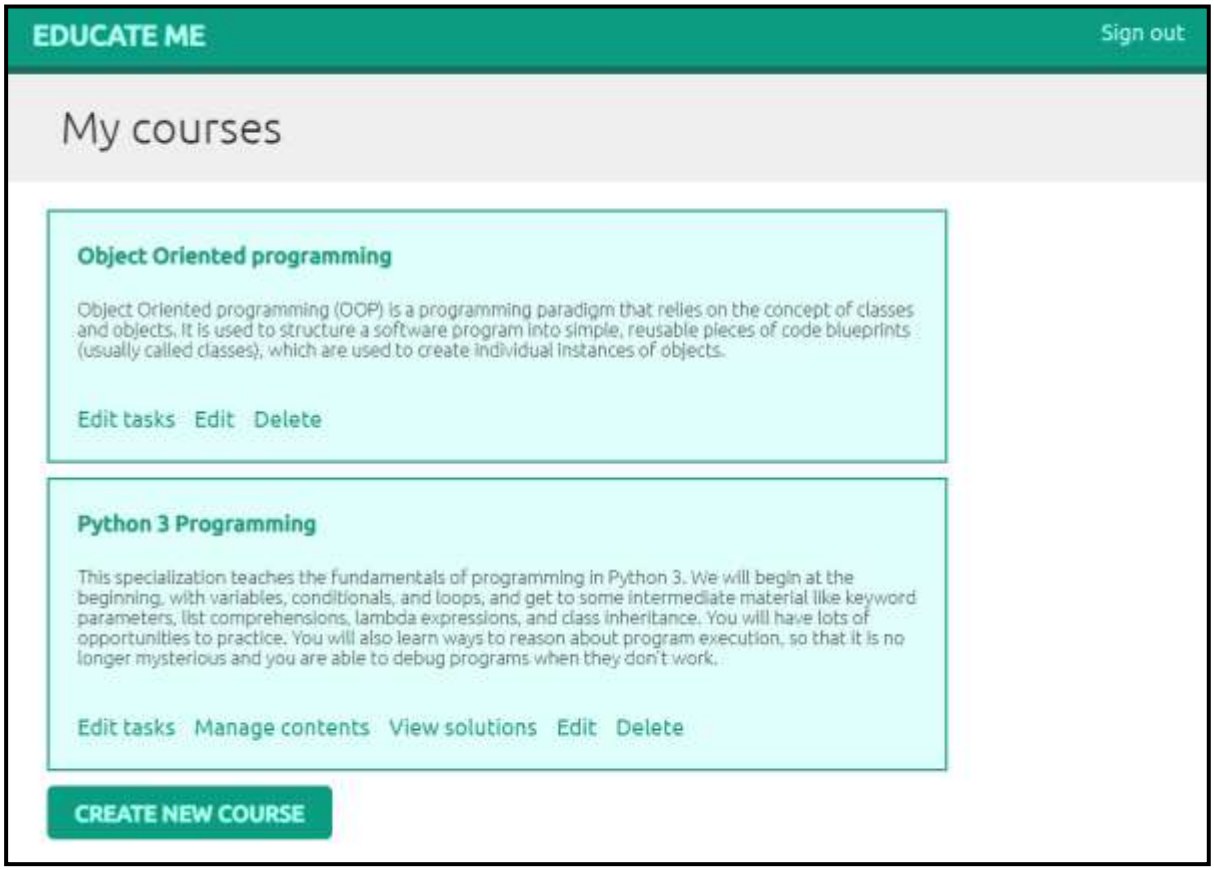

Рисунок 5.1.2 – Сторінка зі списком курсів у модулі викладача

Після перенаправлення на сторінку зі списком курсів у модулі викладача, можна створити новий курс, або виконати маніпуляції над уже існуючими. Форма створення нового курсу зображення на рисунку 5.1.3. Форма редагування курсу є аналогічною до створення.

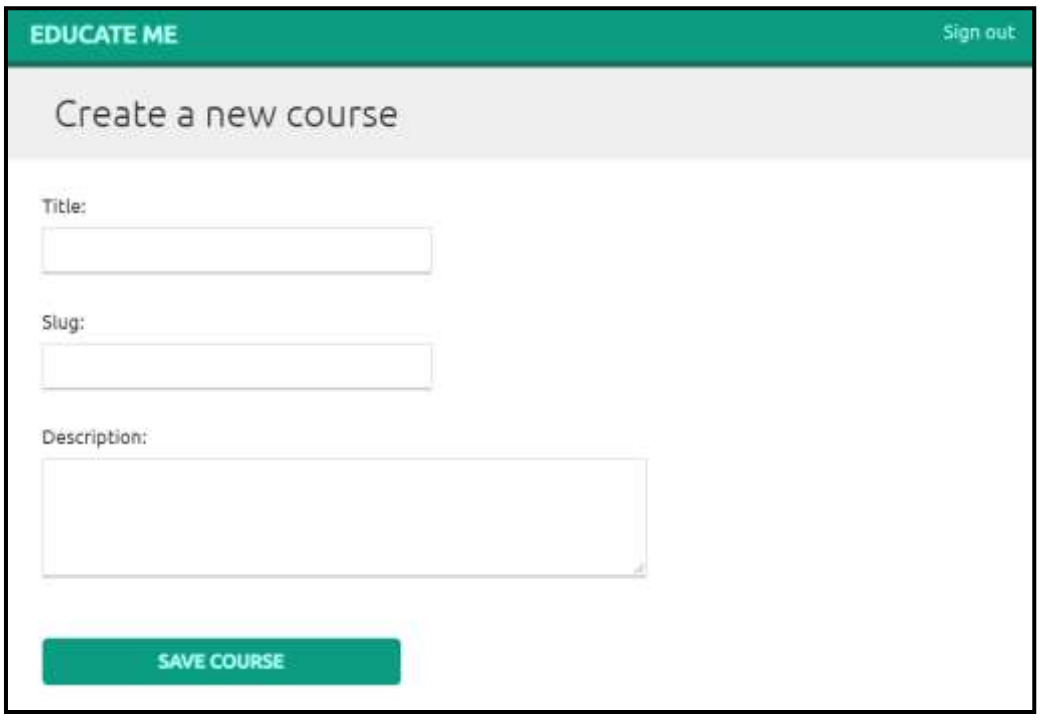

Рисунок 5.1.3 – Сторінка створення нового курсу

У вже створених курсах можна створювати завдання та редагувати їх, в тому числі, наповнюючи контентом. Сторінка управління завданнями зображена на рисунку 5.1.4.

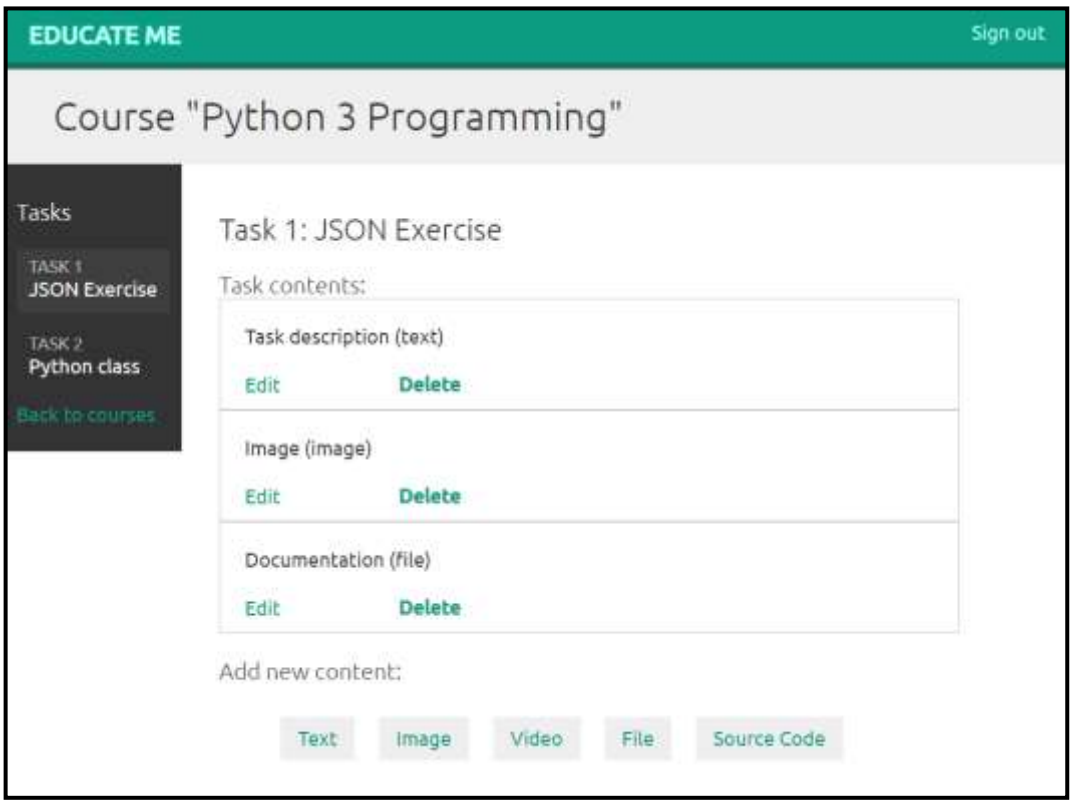

Рисунок 5.1.4 – Сторінка управління завданнями

Крім того, у вже створених курсах можна переглядати відправлені студентами рішення по кожному завданню. На рисунку 5.1.5 зображено список рішень студентів, а на рисунку 5.1.6 – рішення конкретного студента.

Рисунок 5.1.5 – Сторінка перегляду списку рішень студентів

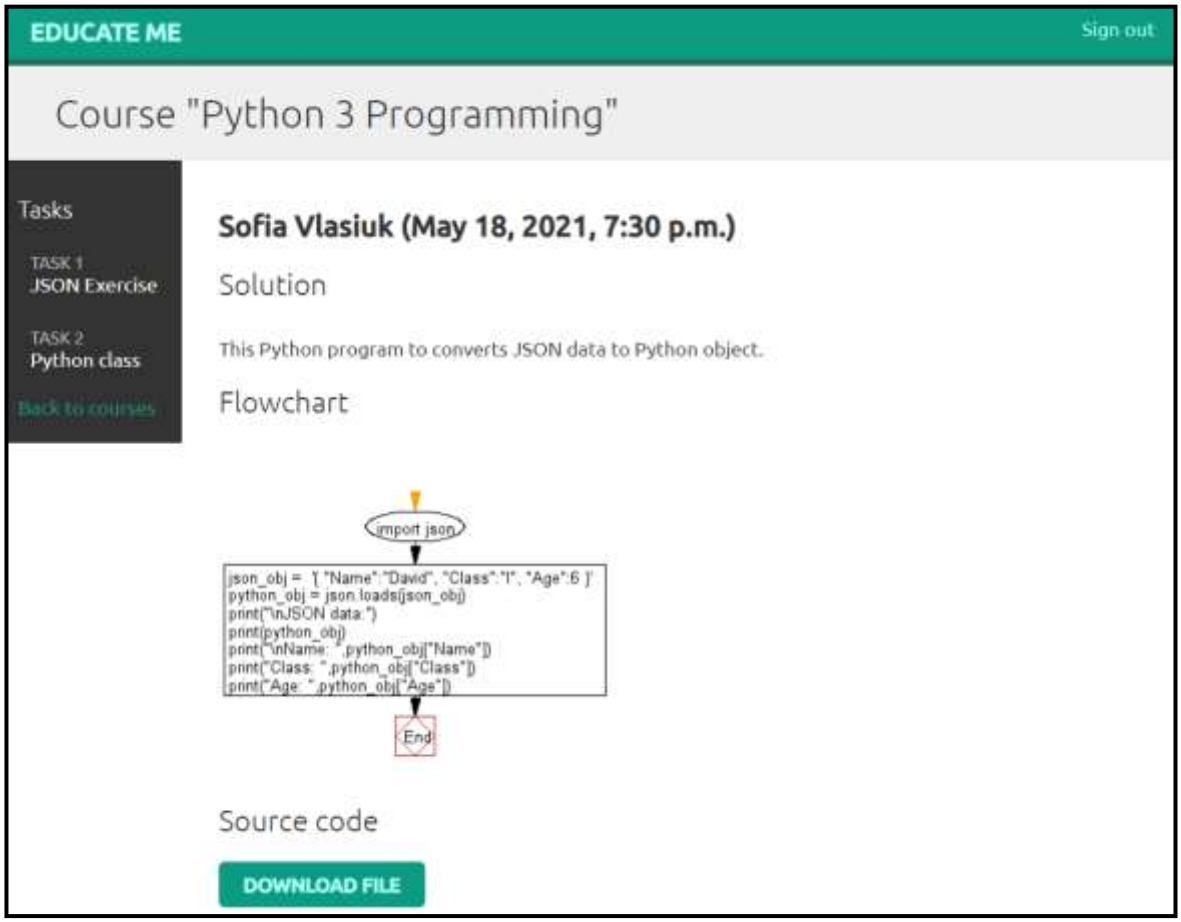

Рисунок 5.1.6 – Сторінка перегляду рішення конкретного студента

Крім того, у модулі викладача передбачені відповідні кнопки видалення курсів та виходу із аккаунту. Але ці функції не мають супроводжуючих форм, тому не мають необхідності в демонстрації.

5.3. Робота із модулем студента

Для роботи у модулі студента, необхідно авторизуватись у системі. Сторінка авторизації зображена на рисунку 5.1.1. Якщо студент ще не зареєстрований, він може скористатись сторінкою реєстрації. Сторінка реєстрації користувача зображена на рисунку 5.2.1.

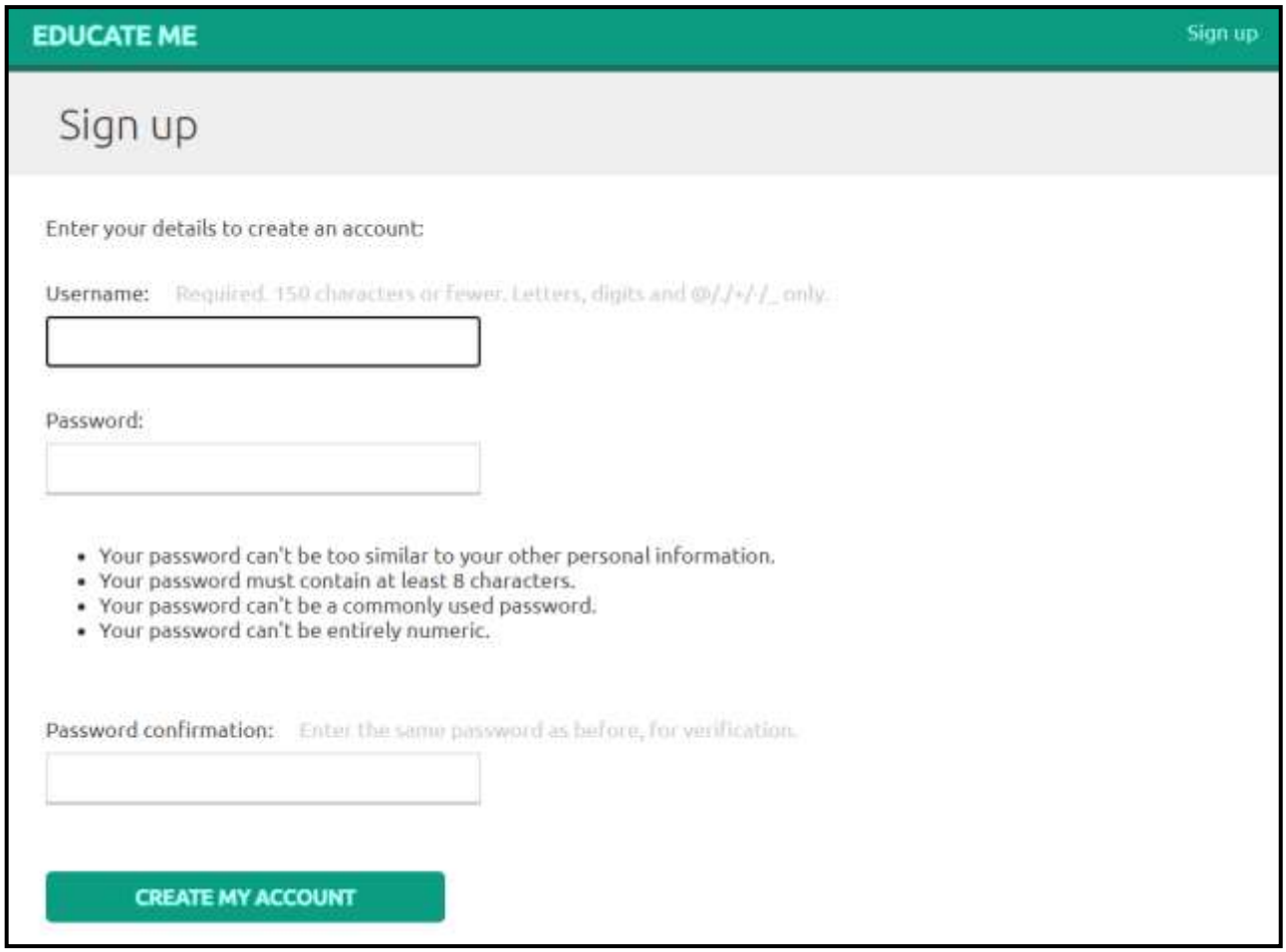

Рисунок 5.2.1 – Сторінка реєстрації користувача

Після авторизації у системі, студент потрапляє на сторінку зі списком всіх своїх курсів (предметів). Відповідна сторінка зображена на рисунку 5.2.2.

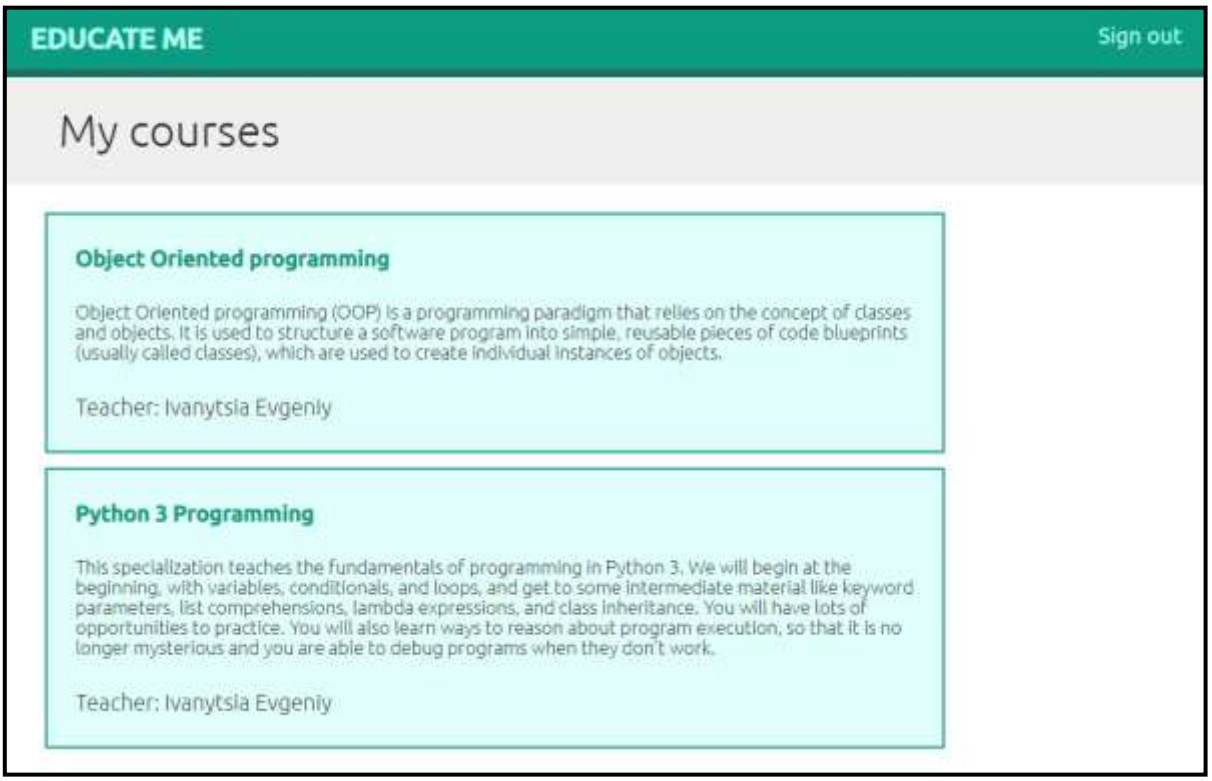

Рисунок 5.2.2 – Сторінка зі списком всіх своїх курсів модуля студента

Вибравши та натиснувши на конкретний курс, відбувається перенаправлення на сторінку перегляду завдань по обраному курсу. Відповідна сторінка зображена на рисунку 5.2.3.

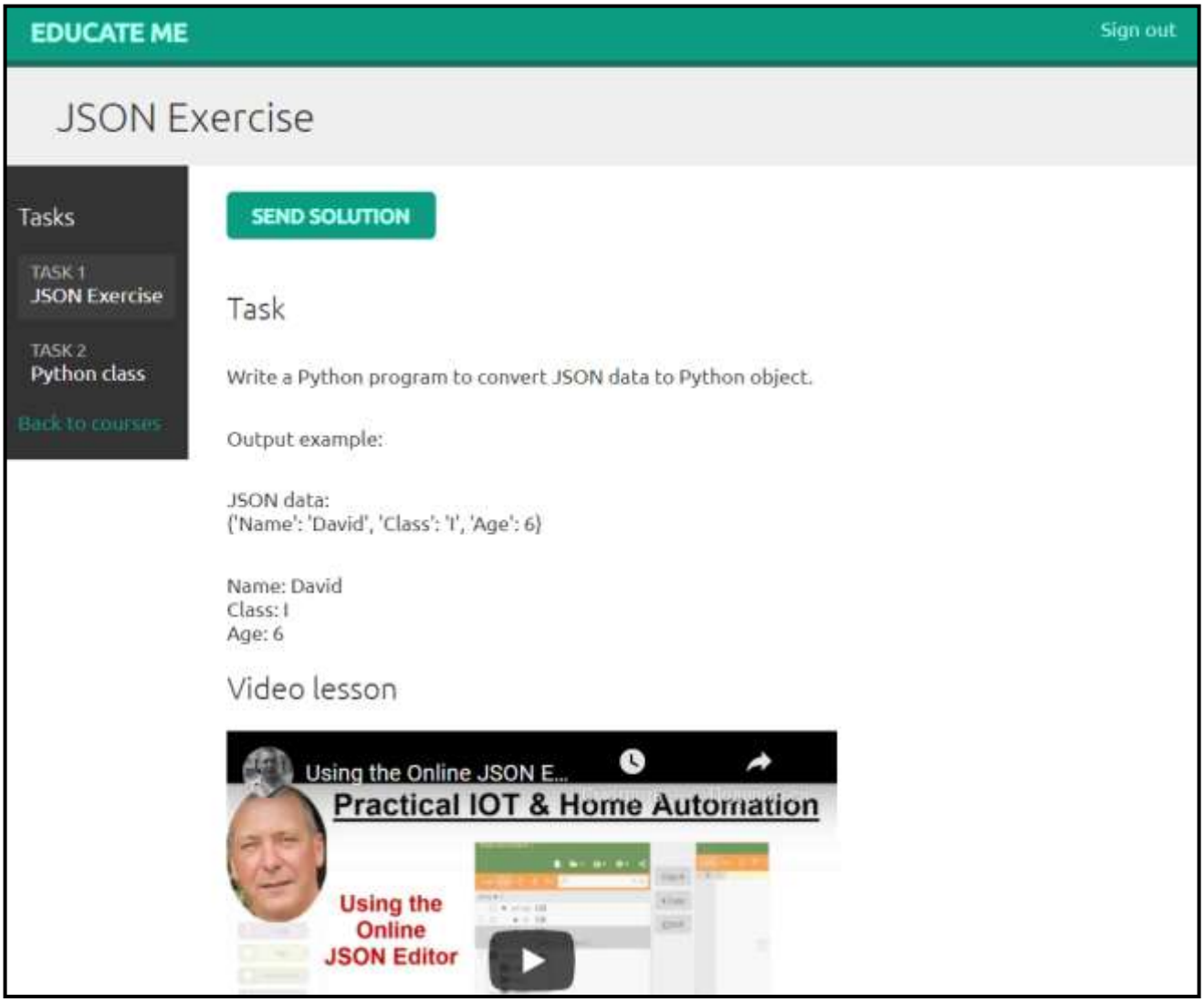

Рисунок 5.2.3 – Сторінка перегляду завдання

Ознайомившись із завданням, студент має можливість надіслати його рішення, натиснувши на кнопку “Send solution” та наповнивши своє рішення контентом. Сторінка створення рішення зображена на рисунку 5.2.4.

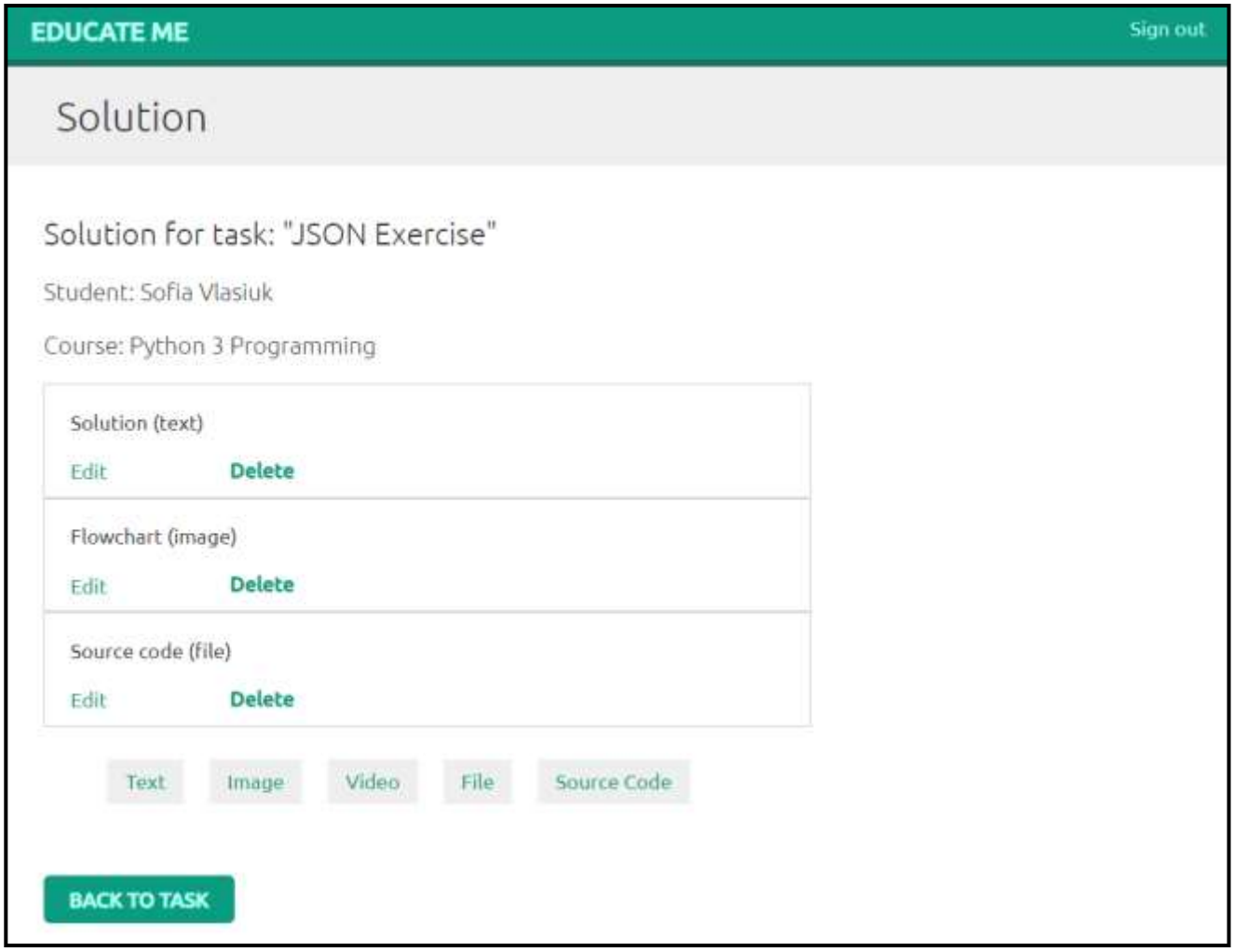

Рисунок 5.2.4 – Сторінка створення рішення

Крім того, у модулі студента передбачені інші допоміжні можливості, так як, наприклад, вихід із аккаунта. Але ці функції не мають супроводжуючих форм, тому не мають необхідності в демонстрації.

5.4. Робота із модулем адміністрування

Для роботи у модулі адміністрування, необхідно авторизуватись у системі. Сторінка авторизації зображена на рисунку 5.1.1. Після авторизації, якщо поточний користувач має роль адміністратора, відбудеться автоматичне перенаправлення на головну сторінку модуля адміністрування (рисунок 5.3.1).

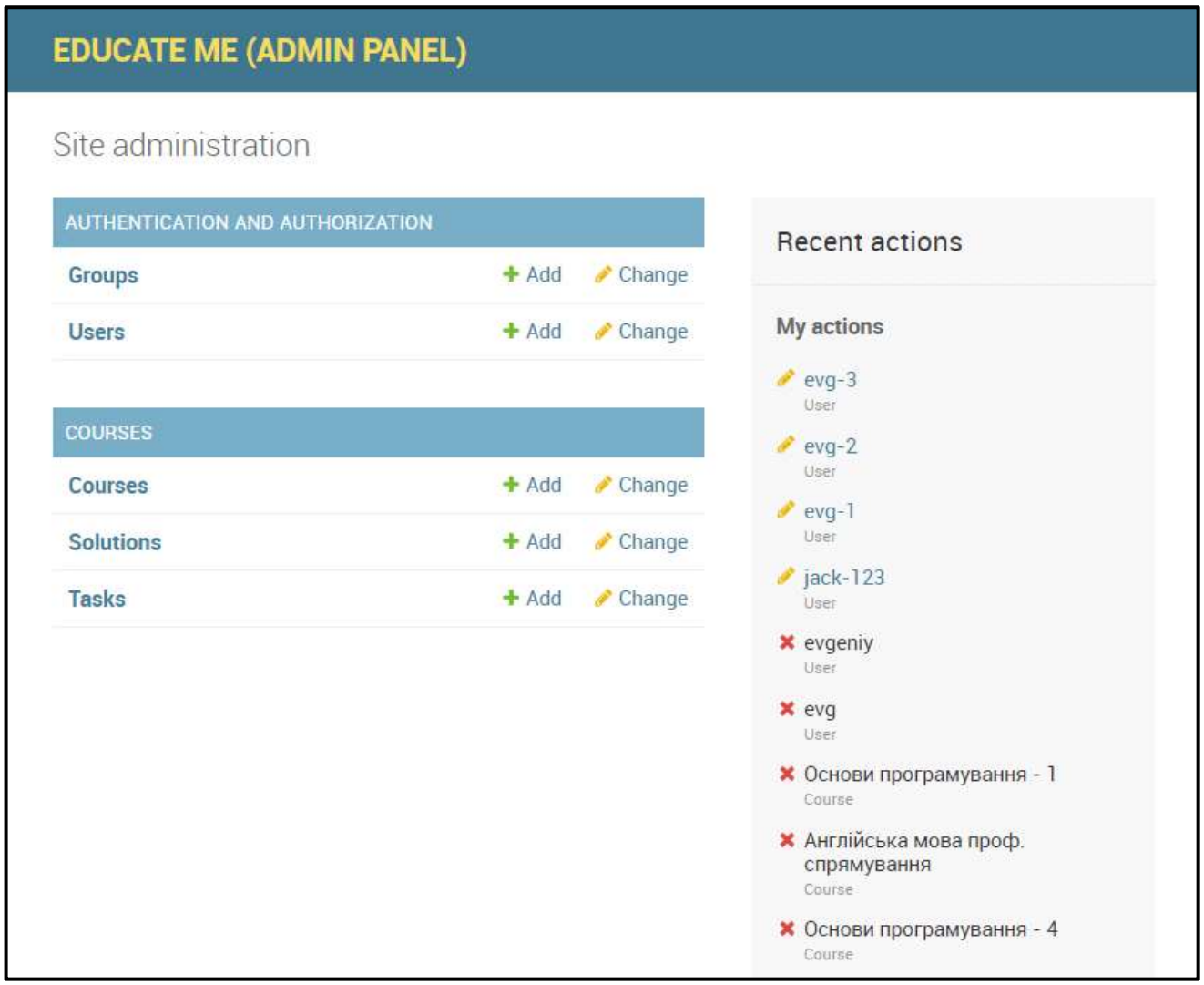

Рисунок 5.3.1 – Модуль адміністрування

На рисунку 5.3.1 зображено головну сторінку модуля адміністрування, що складається із трьох основних блоків: панель управління користувачами, панель управління даними системи (курси, завдання та рішення), а також із панелі, на якій відображені історія змін у модулі адміністрування веб-платформи.

За допомогою панелі адміністрування, адміністратор може змінювати, додавати та видаляти усі поля в базі даних. В тому числі, у модулі адміністратора користувачам можна присвоювати ролі викладача чи студента.

Як приклад, на рисунку 5.3.2 продемонстровано сторінку редагування користувачів.

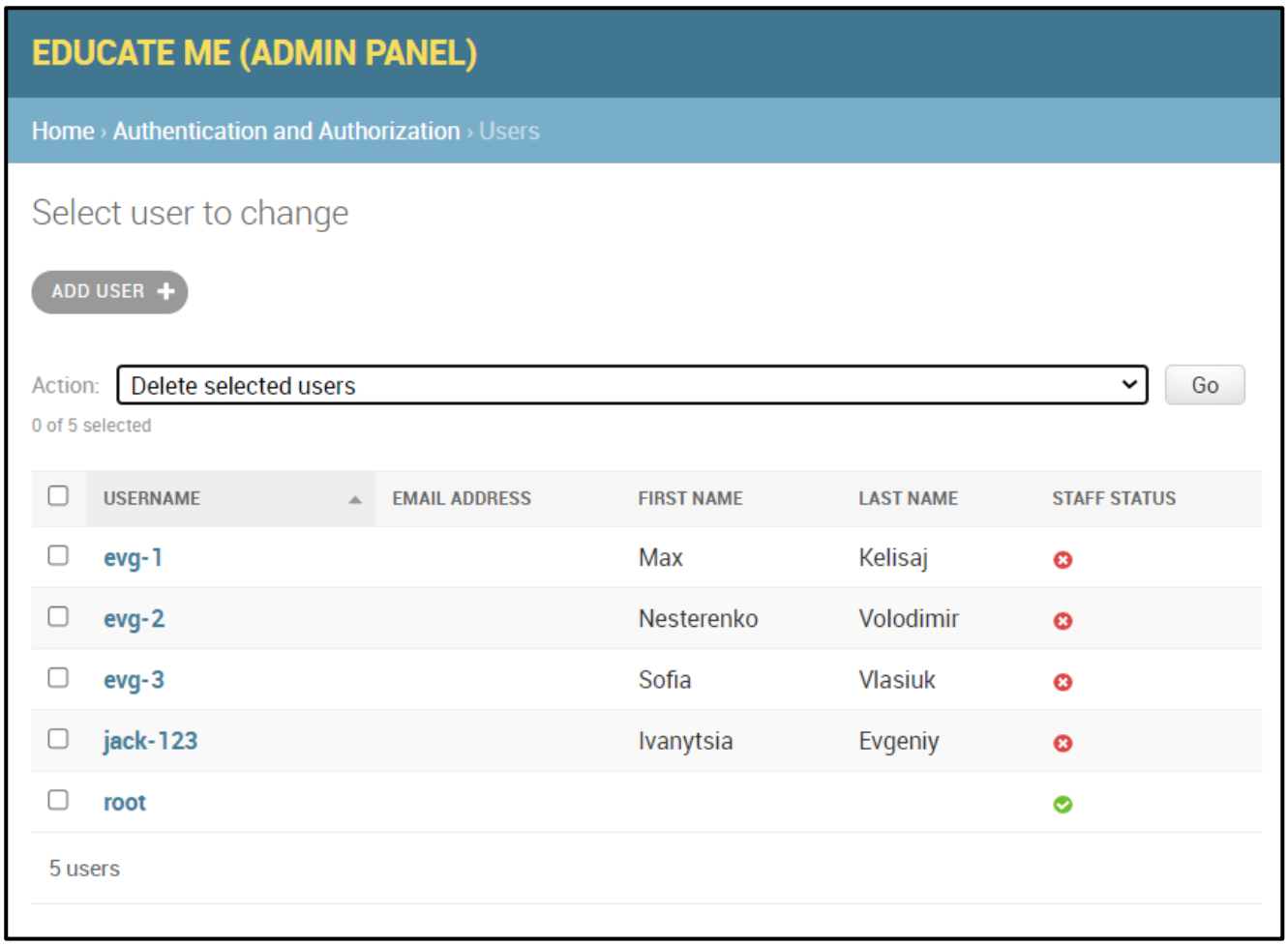

Рисунок 5.3.2 – Сторінка редагування користувачів

Сторінка редагування користувачів складається зі списку із інформацією про всіх користувачів системи. За допомогою випадаючого меню можна застосувати масові дії до обраних користувачів, або відредагувати конкретного користувача, натиснувши на його нікнейм.

5.5. Робота із модулем онлайн-компілятора

При переході на сторінку онлайн-компілятора, відразу відкривається сторінка із текстовим редактором та усіма доступними можливостями. Сторінка зображена на рисунку 5.4.1.

Рисунок 5.4.1 – Сторінка онлайн-компілятора

На сторінці онлайн-компілятора можна змінити поточну мову програмування у випадаючому вікні (рисунок 5.4.2), змінити тему на світлу, або ввести вихідні дані до програми (рисунок 5.4.3), оскільки за замовчуванням тема редактора темна.

Рисунок 5.4.2 – Зміна мови програмування

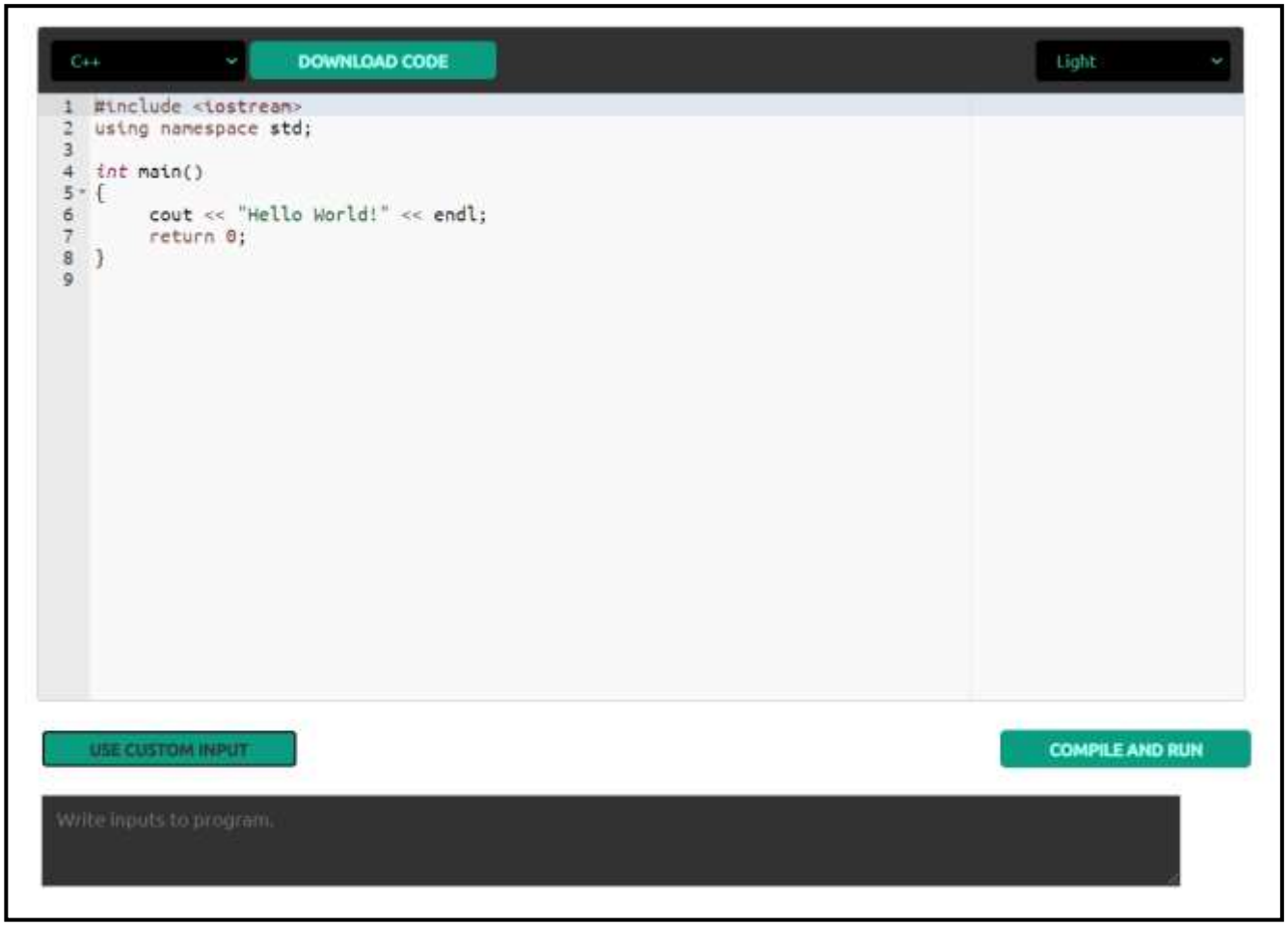

Рисунок 5.4.3 – Онлайн-компілятор у світлій темі із відкритим полем вводу вихідних даних

Для виконання введеного вихідного коду необхідно натиснути кнопку “Compile and run”, після чого програма зкомпілюється, виконається, а результат виконання з’явиться у новому полі (рисунок 5.4.4).

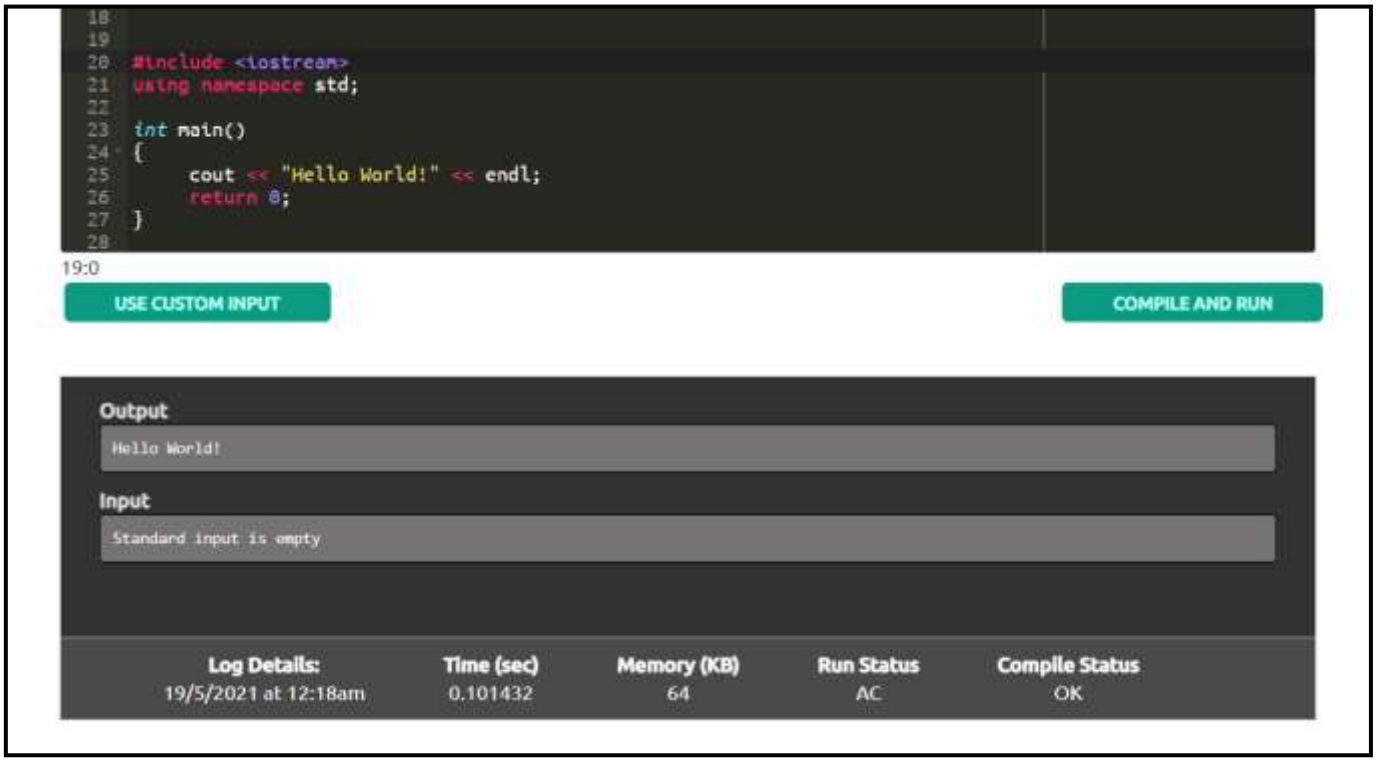

Рисунок 5.4.4 – Результат виконання програми

Крім того, модуль дозволяє завантажувати написаний у редакторі код за допомогою кнопки “Download code”.

5.4. Висновки до розділу

У даному розділі описані системні вимоги до розробленого додатку. Крім того, у розділі наведено керівництво користувача, в якому описані всі варіанти взаємодії користувача із програмною системою у модулі студента, викладача, адміністрування на онлайн-компілятора.

ВИСНОВКИ

В ході У ході виконання даної роботи було створено повноцінну систему обліку робіт студентів із можливістю віддаленої компіляції вихідного коду. Додаток було розроблено мовою програмування Python на основі фреймворку Django. Система складається із клієнтського модуля викладача, клієнтського модуля студента, системи адміністрування, модуля онлайн-компілятора, веб-сервера, серверного додатку та бази даних.

У модулі викладача реалізовано наступні можливості:

- Створення, перегляд, редагування та видалення курсів;
- Створення, перегляд, редагування та видалення завдань, які викладач створює для окремого курсу;
- Наповнення завдань контентом;
- Перегляд рішень студентів по кожному завданню.

Модуль студента реалізує наступні можливості:

- Перегляд інформації про курс;
- Перегляд інформації та контенту по завданням для кожного курсу;
- Створення, перегляд, редагування та видалення рішень по кожному завданню;
- Додавання контенту до кожного рішення.

У системі адміністрування реалізовано наступні можливості:

- Можливість перегляду, редагування та видалення будь-яких полів бази даних;
- Додавання, редагування та видалення користувачів системи;
- Встановлення та вилучення прав доступу (студент, викладач, адміністратор) для зареєстрованих користувачів системи;
- Перегляд історії змін у модулі адміністрування.

Онлайн-компілятор працює окремим відокремленим модулем у системі і реалізовує наступний функціонал:

- Компіляція та виконання вихідних кодів програм, написаних такими мовами програмування як: C, C++, C#, Go, Java, JavaScript, Pascal, PHP, Python;
- Користувацький ввід даних перед виконанням програми;
- Вивід результатів компіляції;
- Вбудований текстовий редактор для написання вихідних кодів програм безпосередньо у модулі онлайн-компілятора;
- Зміна теми текстового редактора (світла і темна);
- Завантаження написаних вихідних кодів на пристрій користувача;

Вищеописані користувацькі модулі реалізовані окремими додатками фреймворку Django. Інтерфейс системи реалізовано засобами HTML, CSS, JavaScript. Серверна частина – мовою програмування Python на основі фреймворку Django та із використанням СУБД SQLite.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Kats Y. Learning Management System Technologies and Software Solutions for Online Teaching: Tools and Applications: Tools and Applications / Yefim Kats. – Нью-Йорк: IGI Global, 2010. – 486 с.

2. Konstantina O. Perceived Usability Evaluation of Learning Management Systems: Empirical Evaluation of the System Usability Scale [Електронний ресурс] / O. Konstantina, T. Nikolaos, C. Katsanos // International Journal of Advanced Computer Science. – 2014. – Режим доступу до ресурсу: https://www.researchgate.net/profile/Nikolaos_Tselios/publication/268388033_Perceived_Usability_Evaluation_of_Learning_Management_Systems_Empirical_Evaluation_of_the_System_Usability_Scale/links/546a0daa0cf2397f78300f9b.

3. Weaver D. Academic and student use of a learning management system: Implications for quality [Електронний ресурс] / Weaver D., Spratt C., Nair C. S. // Australasian Journal of Educational Technology. – 2008. – Режим доступу до ресурсу: https://ajet.org.au/index.php/AJET/article/view/1228.

4. Moodle documentation [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://docs.moodle.org/311/en/Main_page.

5. Google Classroom Help [Електронний ресурс] – Режим доступу до ресурсу: https://support.google.com/edu/classroom/?hl=en&authuser=0#topic=10298088.

6. Docebo documentation [Електронний ресурс] – Режим доступу до ресурсу: https://www.docebo.com/why-docebo/.

7. Python 3.9.5 Documentation [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://www.python.org/doc/.

8. Holovaty J. The Definitive Guide to Django: Web Development Done Right / J. Holovaty, A. Kaplan-Moss. – Нью-Йорк: Apress, 2009. – 536 с.

9. Django Project MVT Structure [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.geeksforgeeks.org/django-project-mvt-structure/.

10. JavaScript [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/JavaScript.

11. HTML Living Standard [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://html.spec.whatwg.org/.

12. W3C Cascading Style Sheets [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3.org/Style/CSS/Overview.en.html.

13. SQLite Documentation [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://www.sqlite.org/docs.html.

14. Rosenberg D. Use Case Driven Object Modeling with UML / D. Rosenberg, M. Stephens. – Нью-Йорк: Apress, 2007. – 440 с.

ДОДАТОК А

Система обліку робіт студентів із можливістю віддаленої компіляції вихідного коду

Специфікація

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ71161_21Б

Аркушів 2

Київ — 2021

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ71161_21Б	Записка.docx	Текстова частина дипломної роботи
Компоненти		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ71161_21Б 12-1	swms	Основний компонент роботи системи, регулює поведнку та взаэмодію всіх модулів
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ71161_21Б 12-2	courses	Компонент, що відповідає за всі маніпуляції при роботі із курсами, реалізує модуль викладача
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ71161_21Б 12-3	students	Компонент, що реалізує функціонал роботи модуля студента
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ71161_21Б 12-4	compiler	Компонент, реалізуючий роботу онлайн-компілятора
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ71161_21Б 12-5	manage.py	Утиліта для для виконання адміністративних команд системи

ДОДАТОК Б

Система обліку робіт студентів із можливістю віддаленої компіляції вихідного коду

Текст програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ71161_21Б

Аркушів 12

Київ — 2021

manage.py

```
#!/usr/bin/env python
import os
import sys

if __name__ == '__main__':
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'swms.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)
```

models.py

```
from django.db import models
from django.contrib.auth.models import User
from django.contrib.contenttypes.models import ContentType
from django.contrib.contenttypes.fields import GenericForeignKey
from .fields import OrderField
from django.template.loader import render_to_string
from django.utils.safestring import mark_safe


class Course(models.Model):
    teacher = models.ForeignKey(User,
                                related_name = 'courses_created',
                                on_delete = models.CASCADE)
    title = models.CharField(max_length = 50)
    slug = models.SlugField(max_length = 50, unique = True)
    description = models.TextField(blank = True)
    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


class Task(models.Model):
    course = models.ForeignKey(Course,
                               related_name = 'tasks',
                               on_delete = models.CASCADE)
    title = models.CharField(max_length = 50)
    description = models.TextField(blank = True)
```

```
    created = models.DateTimeField(auto_now_add = True)
    order = OrderField(blank=True, for_fields=['course'])

    class Meta:
        ordering = ['order']

    def __str__(self):
        return '{}. {}'.format(self.order, self.title)


class Solution(models.Model):
    task = models.ForeignKey(Task,
                             related_name = 'solutions',
                             on_delete = models.CASCADE)
    student = models.ForeignKey(User,
                              related_name = 'solution_performer',
                              on_delete = models.CASCADE)
    created = models.DateTimeField(auto_now_add = True)

    class Meta:
        ordering = ['-created']

    def __str__(self):
        return self.task.title


class Content(models.Model):
     task = models.ForeignKey(Task,
                              related_name='contents',
                              on_delete=models.CASCADE)
     content_type = models.ForeignKey(ContentType,
                              on_delete=models.CASCADE,
                              limit_choices_to={'model__in':(
                                                  'text',
                                                  'sourcecode',
```

```
                                        'video',
                                        'image',
                                        'file')})
    object_id = models.PositiveIntegerField()
    item = GenericForeignKey('content_type', 'object_id')
    order = OrderField(blank=True, for_fields=['task'])

    class Meta:
        ordering = ['order']


class SolutionContent(models.Model):
    solution = models.ForeignKey(Solution,
                        related_name='sol_contents',
                        on_delete=models.CASCADE)
    content_type = models.ForeignKey(ContentType,
                        on_delete=models.CASCADE,
                        limit_choices_to={'model__in':(
                                        'text',
                                        'sourcecode',
                                        'video',
                                        'image',
                                        'file')})
    object_id = models.PositiveIntegerField()
    item = GenericForeignKey('content_type', 'object_id')
    order = OrderField(blank=True, for_fields=['solution'])

    class Meta:
        ordering = ['order']


class ItemBase(models.Model):
    teacher = models.ForeignKey(User,
                        related_name='%(class)s_related',
                        on_delete=models.CASCADE)
```

```
    title = models.CharField(max_length=250)
    created = models.DateTimeField(auto_now_add=True)
    updated = models.DateTimeField(auto_now=True)

    class Meta:
        abstract = True

    def __str__(self):
        return self.title

    def render(self):
        return render_to_string('courses/content/{}.html'.format(
            self._meta.model_name), {'item': self})


class Text(ItemBase):
    content = models.TextField()


class SourceCode(ItemBase):
    code = models.TextField()
    language = models.CharField(max_length = 20)


class File(ItemBase):
    file = models.FileField(upload_to='files')


class Image(ItemBase):
    file = models.FileField(upload_to='images')


class Video(ItemBase):
    url = models.URLField()
```

models.py

```
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, Task, Content, Solution
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 django.views.generic.detail import DetailView
from .forms import TaskFormSet
from django.forms.models import modelform_factory
from django.apps import apps
from django.db.models import Count


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


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


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


class OwnerCourseEditMixin(OwnerCourseMixin, OwnerEditMixin):
    fields = ['title', 'slug', 'description']
```

```
    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 CourseTaskUpdateView(TemplateResponseMixin, View):
    template_name = 'courses/manage/task/formset.html'
    course = None

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

    def dispatch(self, request, pk):
        self.course = get_object_or_404(Course, id=pk, teacher=request.user)
        return super(CourseTaskUpdateView, 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 ContentCreateUpdateView(TemplateResponseMixin, View):
    task = None
    model = None
    obj = None
    template_name = 'courses/manage/content/form.html'

    def get_model(self, model_name):
        if model_name in ['text', 'sourcecode', '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=['teacher',
                                                 'order',
                                                 'created',
                                                 'updated'])
        return Form(*args, **kwargs)

    def dispatch(self, request, task_id, model_name, id=None):
        self.task = get_object_or_404(Task, id=task_id, course__teacher=request.user)
        self.model = self.get_model(model_name)
        if id:
            self.obj = get_object_or_404(self.model, id=id, teacher=request.user)
        return super(ContentCreateUpdateView, self).dispatch(request, task_id, model_name, id)
```

```
    def get(self, request, task_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.teacher = request.user
            obj.save()
            if not id:
                Content.objects.create(task=self.task, item=obj)
            return redirect('task_content_list', self.task.id)
        return self.render_to_response({'form': form, 'object': self.obj})


class ContentDeleteView(View):
    def post(self, request, id):
        content = get_object_or_404(Content,
                                    id=id,
                                    task__course__teacher=request.user)
        task = content.task
        content.item.delete()
        content.delete()
        return redirect('task_content_list', task.id)


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

    def get(self, request, task_id):
        task = get_object_or_404(Task,
```

```
                                id=task_id,
                                course__teacher=request.user)
        return self.render_to_response({'task': task})


class TaskListView(TemplateResponseMixin, View):
    model = Task
    template_name = 'courses/course/list.html'

    def get(self, request, course = None):
        courses = Course.objects.annotate(total_tasks = Count('tasks'))
        tasks = Task.objects.all()
        if course:
            course = get_object_or_404(Course, slug = course)
            tasks = tasks.filter(course = course)

        return self.render_to_response({'courses': courses,
                                        'course': course,
                                        'tasks': tasks})


class TaskDetailView(DetailView):
    model = Task
    template_name = 'courses/course/detail.html'

    def get(self, request, task_id):
        task = get_object_or_404(Task,
                                id=task_id)
        return self.render_to_response({'task': task})


class TaskSolutionsListView(TemplateResponseMixin, View):
    template_name = 'courses/manage/solution/list.html'

    def get(self, request, task_id):
```

```
        task = get_object_or_404(Task,
                                 id=task_id,
                                 course__teacher=request.user)
        return self.render_to_response({'task': task})


class TaskSolutionView(DetailView):
    template_name = 'courses/manage/solution/detail.html'

    def get(self, request, task_id, solution_id):
        solution = get_object_or_404(Solution,
                                 id=solution_id)
        return self.render_to_response({'solution': solution})
```

ДОДАТОК В

Система обліку робіт студентів із можливістю віддаленої компіляції вихідного коду

Опис програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ71161_21Б 13-1

Аркушів 10

Київ — 2021

АНОТАЦІЯ

Додаток описує веб-платформу для здачі та обліку робіт студентів із можливістю віддаленої компіляції вихідного коду засобами вбудованого онлайн-компілятора.

Система надає викладачам і студентам інструментарій для роботи із курсами, створення і поширення завдань, здачі рішень із можливістю наповнювнення їх контентом. Крім того, в системі окремим модулем реалізований онлайн-компілятор для компіляції та запуску вихідних кодів безпосередньо у системі.

Веб-платформа розроблема переважно засобами веб-фреймворку Django мови програмування Python у середовищі PyCharm.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ

У додатку міститься опис основних компонентів веб-платформи для здачі та обліку робіт студентів із можливістю віддаленої компіляції вихідного коду засобами вбудованого онлайн-компілятора. Додаток Б містить програмний код цих компонентів.

Для роботи веб-платформи потрібно мати будь-який браузер незалежно від платформи використання та операційної системи, а також стабільне підключення до мережі Інтернет.

Система розроблема переважно засобами веб-фреймворку Django мови програмування Python у середовищі PyCharm.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Клієнтський модуль викладача реалізовує наступні можливості в рамках системи:

- Створення, перегляд, редагування та видалення курсів (предметів);
- Створення, перегляд, редагування та видалення завдань, які викладач створює для окремого курсу;
- Наповнення завдань контентом: текстом, зображеннями, відео, або іншими файлами, які б дозволяли зручно переглянути весь матеріал по конкретній темі та виконати відповідне завдання студентам;
- Перегляд рішень студентів по кожному завданню.

Клієнтський модуль студента, дозволяє користувачам виконувати наступні дії у системі:

- Перегляд інформації про курс (предмет);
- Перегляд інформації та контенту по завданням для кожного курсу;
- Створення, перегляд, редагування та видалення рішень по кожному завданню;
- Додавання контенту із описом до кожного рішення: супроводжувальний текст, зображення, відео, файли будь-якого формату, вихідні коди програм для того, аби сформовані рішення підходили до завдань будь-якого формату.

Наступний модуль – система адміністрування. У цьому модулі реалізовано наступні можливості:

- Перегляд, редагування та видалення будь-яких даних в системі (курсів, завдань, рішень);
- Додавання, редагування та видалення користувачів системи;

- Встановлення та вилучення прав доступу (студент, викладач, адміністратор) для зареєстрованих користувачів системи;
- Перегляд історії змін у модулі адміністрування.

Онлайн-компілятор працює окремим відокремлений модулем у системі і реалізовує наступний функціонал :

- Компіляція та виконання вихідних кодів програм, написаних основними мовами програмування (C, C++, C#, Go, Java, JavaScript, Pascal, PHP, Python);
- Користувацький ввід даних перед виконанням програми;
- Вивід результатів компіляції;
- Вбудований текстовий редактор для написання вихідних кодів програм безпосередньо у модулі онлайн-компілятора;
- Зміна теми текстового редактора (світла і темна);
- Завантаження написаних вихідних кодів на пристрій користувача;

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Створена веб-платформа складається із чотирьох користувацьких модулів: модуля викладача, модуля студента, модуля адміністрування та віокремленого модуля онйлайн-компілятора.

Система використовує архітектуру MTV (Model-Template-View). Вона використовує шаблони для представлень та контролера. Шаблон відноситься до представлення у архітектурі MVC (Model-View-Controller) та, по суті, контролює вміст, який відображатиметься, та те, як він відображатиметься для користувача.

Модель – це частина програми, яка виступає посередником між інтерфейсом веб-сайту та базою даних. Модель є структурою даних, що стоїть за цілим додатком, і представлена базою даних. Модель – це компонент, який є частиною бізнес-логіки в архітектурі Django.

Представлення – це компонент, який містить логіку інтерфейсу в архітектурі Django. представлення насправді є інтерфейсом користувача програми. Вони представлені файлами HTML/ CSS/ Javascript. Як правило, вміст надходить з компонента моделей.

Контролер, як випливає з назви, є основним компонентом управління. Це означає, що контролер обробляє взаємодію користувача та вибирає представлення відповідно до моделі. Основне завдання контролера – вибрати компонент подання відповідно до взаємодії користувача, а також застосувати компонент моделі.

ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Для роботи клієнтської частини веб-платформи потрібно мати будь-який браузер, а також стабільне підключення до мережі Інтернет. Ніяких обмежень у виборі платформи та браузера не накладається.

Система проектувалась для роботи під управлінням Apache з модулем mod_python і з використанням SQLite в якості бази даних.

Система також може працювати під управлінням FastCGI, mod wsgi, або SCGI на Apache і інших серверах (lighttpd, nginx тощо), сервера uWSGI. А також працювати з іншими СУБД: MySQL, SQLite, Microsoft SQL Server, DB2, Firebird, SQL Anywhere і Oracle.