Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет *Інфокомунікацій*
(повна назва)

Кафедра *Інформаційно-мережної інженерії*
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіті *перший (бакалаврський)*

Налаштування сервера системи автоматичної перевірки практичних завдань

(тема)

Виконав:
здобувач *4* року навчання,
групи *ТРІМІ-21-2*
Євгеній Цемма
(власне ім'я, прізвище)
Спеціальність *172 Телекомунікації та радіотехніка*
(код і повна назва спеціальності)
Тип програми *освітньо-професійна*
Освітня програма *Інформаційно-мережна інженерія*
(повна назва освітньої програми)
Керівник: *доцент Андрій Костромицький*
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ІМІ _____ *Валерій Безрук*
(підпис) (власне ім'я, прізвище)

2025 р.

Не містить відомостей, заборонених до відкритого публікування

Студент _____ / *Цемма Є.О.* /
(підпис) (прізвище та ініціали)

Керівник _____ / *Костромицький А.І.*/
(підпис) (прізвище та ініціали)

Харківський національний університет радіоелектроніки

Факультет_____інфокомунікацій_____

Кафедра_____інформаційно-мережної інженерії_____

Рівень вищої освіти _____перший (бакалаврський)_____

Спеціальність _____172 Телекомунікації та радіотехніка_____
(код і повна назва)

Тип програми _____освітньо-професійна_____

Освітня програма _____інформаційно-мережна інженерія_____
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«____»_____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві_____Цеммі Євгенію Олександровичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи __Налаштування сервера системи автоматичної перевірки практичних завдань_____
_____

затверджена наказом університету від _«23»___ ___05_____20_25_ р. № _410СТ_____

2. Термін подання здобувачем роботи до екзаменаційної комісії _23_ _06___ 20_25_ р.

3. Вихідні дані до роботи __ Вихідний код проекту системи автоматизації перевірки практичних завдань. Необхідно розгорнути та налаштувати сервер в AWS та налаштувати інтеграцію з тестовим курсом в системі дистанційного навчання Moodle (https://dl.nure.ua/course/view.php?id=20324).

4. Перелік питань, що потрібно опрацювати в роботі _____
Вступ_____
1. Система керування навчання_____
2. Архітектура додатку_____
3. Налаштування серверного середовища_____
4. Налаштування завдання у Moodle_____
5. Усунення помилок після розгортання проекту_____
Висновок_____

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Слайди у форматі PowerPoint(мета, актуальність, архітектура, технології, результати, висновки тощо)

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Основна частина	*доц. Костромицький А.І.*		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	*Ознайомлення із завданням. Уточнення ТЗ.*	23.05.25	
2	*Підбір літератури за темою роботи.*	23.05-24.05.25	
3	*Виконання розділу 1*	24.05-27.05.25	
4	*Виконання розділу 2*	29.05-1.06.25	
5	*Виконання розділу 3*	1.06-4.06.25	
6	*Виконання розділу 4*	4.06-8.06.25	
7	*Виконання розділу 5*	8.06.-11.06.25	
8	*Оформлення презентаційного матеріалу,*	11.06.-24.06.25	
	підготовка до захисту у ЕК		

Дата видачі завдання 23 05 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ *доцент Андрій Костромицький*
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 48 с., 30 рис., 5 джерел

Об'єкт роботи – система автоматичної перевірки практичних завдань.

Мета роботи – налаштування сервера системи автоматичної перевірки практичних завдань

Розглянуто систему автоматичної перевірки практичних завдань у середовищі Moodle з використанням протоколу LTI. Реалізовано серверну частину інструмента на Flask за підтримкою WebSocket та базою даних SQLite. Розгорнуто хмарну інфраструктуру за допомогою Terraform та Ansible у середовищі AWS.

AWS, TERRAFORM, ANSIBLE, PYTHON, NJINX, LMS, LTI, СИСТЕМА АВТОМАТИЧНОЇ ПЕРЕВІРКИ ПРАКТИЧНИХ ЗАВДАНЬ

ABSTRACT

Explanatory note: 48 p., 30 fig., 6 sources.

The object of study is an automated assignment checking system.

The purpose of this work is to configures the server for an automated system of checking practical tasks.

The work examines an automated system for evaluating practical tasks within the Moodle environment using the LTI protocol. The server-side of the tool was implemented using Flask, supported by WebSocket communication and SQLite database. A cloud infrastructure was deployed using Terraform and Ansible in the AWS environment.

AWS, TERRAFORM, ANSIBLE, PYTHON, NJINX, LMS, LTI, AUTOMATED ASSIGMENT CHEKING SYSTEM

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

IaC (Infrastructure as code) – інфраструктура як код;

AWS (Amazon Web Services) – хмарні сервіси від компанії Amazon;

EC2 (Elastic Compute Cloud) – сервіс віртуальних машин на AWS;

LMS (Learning Management System) – система управління навчанням;

HTML (HyperText Markup Language) – мова розмітки гіпертексту;

DDoS (Destributed Denial of Service) – розподілена атака на відмову в обслуговуванні;

JSON (JavaScript Object Notation) – формат обміну даних;

IP (Internet Protocol) – інтернет-протокол;

HTTP (HyperText Transfer Protocol) – протокол передачі, який є основним для комунікації між клієнтом і сервером;

HTTPS (HyperText Transfer Protocol Secure) – Зашифрована версія HTTP;

PAM (Pluggable Authentication Modules) – модульна система автентифікації в Linux, яка забезпечує гнучкість налаштувань входу користувачів;

LTI (Learning Tools Interoperability) – стандарт, який дозволяє інтегрувати зовнішні освітні сервіси в Moodle.

ВСТУП

Сучасний світ постійно змінюється під впливом технологій, які дедалі глибше проникають у всі сфери життя, зокрема і в освіту. Протягом останніх років, значно виріс попит на цифрові інструменти, а також платформи, які дозволяють оптимізувати навчальний процес. Особливо це стало помітно під час дистанційного навчання, коли все навчання перейшло в онлайн формат і потрібно було адаптувати всі наявні технології так, щоб ефективність комунікації між викладачем та студентом, а також швидкість зворотного зв’язку були максимально оптимізованими.

Метою кваліфікаційного проекту, було розгорнути, актуалізувати та налаштувати систему, яка дозволить автоматизувати перевірку практичних завдань студентів, зокрема тих, що виконуються у вигляді програмного коду. Під час розгортання проекту, були використані сучасні технології такі як Python-фреймворк Flask для серверної логіки, WebSocket для забезпечення живого зв’язку між користувачем та системою перевірки, Docker для контейнеризації перевірочних середовищ, а також Amazon Web Services (AWS), що дозволяє масштабувати систему за потреби. Для розгортання проекту, згідно найкращих практик, має використовуватися підхід інфраструктура як код (IaC), та Terraform, Ansible зокрема. Важливою частиною проекту також є впровадження інтеграціє з Moodle, через протокол LTI.

1 СИСТЕМА КЕРУВАННЯ НАВЧАННЯМ

1.1 Огляд систем управління навчанням (LMS)

Системи управління навчанням (LMS, Learning Management System) – це платформи, які допомагають організовувати навчальний процес онлайн. Вони стали незамінним інструментом як для викладача і студентів, так і для компаній, які хочуть навчати своїх співробітників. Основне завдання LMS – зробити навчання доступним, зручним та ефективним.

Фактично, LMS – це такий собі цифровий клас, де можна створювати та проходити курси, здавати завдання, отримувати оцінки та спілкуватися з іншими учасниками. Викладачі можуть завантажувати матеріали, перевіряти тести та відстежувати прогрес студентів. А студенти – навчатися в зручний для них час і темп, не прив’язуючись до фізичних аудиторій.

Більшість сучасних LMS працюють у хмарі, це означає, що доступ до них можливий з будь-якого пристрою, підключеного до інтернету. Це дуже зручно, адже навчання стає гнучкішим, а організаційні витрати – меншими. Також такі системи підтримують інтеграцію з іншими інструментами, наприклад , сервісами для відео-зв’язку, автоматичними перевірками завдань і навіть штучним інтелектом. LMS – це не просто програма, а повноцінне навчальне середовище, яке адаптується під потреби користувачів і значно полегшує процес отримання знань. У сучасному світі без такого інструменту вже складно уявити дистанційну освіту, чи професійне навчання. На рисунку 1.1 зображено схему, яка показує складові системи LMS. [1]

Рисунок 1.1 – Графічна схема LMS

1.2 Основні функції LMS

Системи управління навчанням (LMS) пропонують широкий набір інструментів, які допомагають зробити навчальний процес ефективнішим та зручнішим як для викладачів так і для студентів. Основні можливості таких платформ включають :

- Доступність та зручність. LMS дозволяють навчатися з будь-якого місця та пристрою, це особливо важливо для дистанційного навчання. Студенти можуть переглядати матеріали, виконувати завдання та проходити тести в зручний для них час.
- Адміністрування курсів. Викладачі можуть створювати курси, додавати навчальні матеріали(відео, презентації, тести), а також налаштовувати доступ та терміни виконання завдань.
- Автоматизація оцінювання. Система дозволяє проводити тестування та автоматично виставляти оцінки, а це значно економить час, витрачений на перевірку студентів викладачем і забезпечує об’єктивність перевірки.
- Аналітика та звітність. LMS відстежує успішність студентів, також генерує звіти про проходження курсів та дозволяє викладачам аналізувати, які теми викликають найбільші труднощі у студентів.

- Комунікація та співпраця. У більшості платформ є чати, форуми та можливість інтеграції з відеоконференціями, це дозволяє студентам та викладачам спілкуватися та обговорювати навчальні питання.
- Підтримка мобільних пристроїв. Багато сучасних LMS мають мобільні додатки або адаптивні версії, що дозволяє навчатися навіть зі смартфона, або планшета.
- Безпека даних. Оскільки LMS містять персональні дані студентів та викладачів, вони мають захищений доступ, шифрування даних та налаштування прав користувачів. [2, 4]

1.3 Інтеграція LMS та LTI

Щоб зробити навчальний процес ще зручнішим та ефективнішим, LMS мають підтримувати інтеграцію з іншими сервісами та інструментами. Один із ключових стандартів, який забезпечує таку інтеграцію – Learning Tools Interoperability (LTI).

LTI – це спеціальний стандарт, розроблений IMS Global Learning Consortium, він дозволяє різним освітнім платформам легко взаємодіяти між собою. Іншими словами, він працює як міст між LMS та зовнішніми навчальними інструментами.

Завдяки LTI можна, наприклад:

- Підключати зовнішні сервіси без потреби додаткової авторизації (єдиний вхід через LMS).
- Передавати дані між системами в захищеному та стандартизованому форматі.
- Інтегрувати відео лекції, віртуальні лабораторії, інтерактивні вправи та інші навчальні ресурси прямо в LMS.

Як це працює на практиці? Уявімо, що викладач хоче використовувати платформу для тестування, яка знаходиться за межами LMS. Завдяки LTI студенти зможуть проходити тести безпосередньо в LMS, а результати

автоматично з’являться у їхніх профілях. Це спрощує роботу як для студентів так і для викладачів, усуваючи зайві кроки на кшталт ручного перенесення даних.

LTI – це про зручність та гнучкість. Завдяки цьому стандарту, навчальні платформи перестають бути закритими системами, а стають частиною єдиної екосистеми електронного навчання. Це особливо важливо в умовах стрімкого розвитку EdTech, коли кожен новий інструмент має безперешкодно працювати разом з іншими.[2, 4]

2 АРХІТЕКТУРА ДОДАТКУ

2.1 Використання гібридної архітектури у проекті

Під час розгортання проекту для автоматичної перевірки завдань, було прийнято рішення базувати систему на вже існуючій платформі управління навчанням – Moodle. Використання готового рішення має низку переваг: по перше, це дозволяє значно скоротити терміни розгортання, адже велика частина базового функціоналу вже реалізована, по друге, це дає змогу зекономити ресурси, спрямовуючи їх не на створення фундаменту, а на розширення можливостей та вдосконалення користувацького досвіду. Moodle – це перевірене часом рішення, яке активно використовується навчальними закладами у всьому світі і ХНУРЕ не є виключенням.

Одним з важливих елементів системи, є платформа Moodle, яка відповідає за організацію основних процесів, таких як: створення навчальних курсів, формуванням та управлінням завдань для студентів. У межах цієї системи реалізовано широкий спектр функцій, необхідних для повноцінної організації навчального процесу онлайн.

Оскільки стандартний функціонал Moodle передбачає автоматизовану перевірку практичних завдань, це стало основним завданням проекту. Саме тому було створено окремий сервіс, який виконує функцію автоматичного перевіряльника. Цей модуль розроблявся як самостійний сервіс, який функціонує незалежно від пов’язаних з інтеграцією нового функціоналу безпосередньо у великий, щільно зав’язаний код Moodle.

Наступним важливим компонентом, є середовище виконання – це спеціально налаштована віртуальна машина, яка розгорнута на хмарному сервісі AWS. Призначення цієї віртуальної машини – виконання завдань, пов’язаних з із перевіркою виконаних завдань, що надсилаються студентами, вона приймає дані з основного серверу, обробляє їх за допомогою необхідних скриптів і повертає результати перевірки. Таким чином, уся логіка перевірки

ізольована в окремому середовищі, що гарантує більшу безпеку та контроль над процесом.

Для автоматизації налаштування сервера перевірки завдань, використав Terraform та Ansible. Terraform відповідає за створення інфраструктури: мережеві налаштування, групи безпеки тощо. Ansible, дозволяє автоматично налаштовувати сервер після створення цієї інфраструктури, від встановлення залежностей до запуску служб і налаштування середовища виконання. Це дозволило уникнути ручного налаштування на кожному етапі.

Також, в проекті використовується механізм синхронізації даних між основних сервером та worker-машиною, для цього використано WebSocket-з'єднання, яке забезпечує постійну двосторонню комунікацію, між клієнтом і сервером. Це дозволяє системі, миттєво реагувати на запити й не створює зайвого навантаження на систему, як це могло би бути у випадку з REST API.

2.2 Опис серверної частини додатку

Серверна частина системи реалізована на базі хмарної платформи Amazon Web Services (AWS), це дозволило зручно розгортати проект, оскільки таким способом, можна задіяти Terraform та багато інших іструментів, щоб досягти високої доступності та гнучкості в процесі розгортання й подальшої підтримки проекту. Використовувався інстанс, типу t2.micro, що входить до безкоштовного рівня Free Tier. Він забезпечує достатній мінімум для запуску MVP-версії системи, включаючи 1 vCPU та 1 ГБ оперативної пам'яті, чого достатньо для обробки прописаних сценаріїв проекту.

В якості операційної системи, обрав Debian Linux, яка є однією з найбільш стабільних і безпечних у світі Linux-дистрибутивів. Її мінімалістична архітектура та зручне налаштування, робить її гарним середовищем для розгортання сервера. Також, додатковою перевагою є низьке споживання ресурсів, це також є важливим чинником, оскільки я працював з обмеженим інстансом AWS.

На сервері розгорнуто основний застосунок, який обробляє всі запити від користувачів. Тобто з одного боку, Moodle, через який надсилається завдання, а з іншого – віртуальна машина на якій був розгорнутий проект, який перевіряє завдання. Moodle підключається до сервера, після підключення, сервер отримує завдання на перевірку й повертає результат.

Для забезпечення зручної та швидкої взаємодії між клієнтом та сервером, використовувався протокол WebSocket. Його головна перевага – це постійне з’єднання, яке дозволяє обмінюватися даними в режимі реального часу. Тобто не потрібно кожного разу відкривати нове з’єднання, як у випадку з REST API. Такий підхід, зменшує затримки й навантаження на сервер, що особливо корисно під час перевірки коду або передачі статусів у реальному часі. [5]

На початку розробки, застосунок просто приймав HTTP-запити через відкритий порт – цього було достатньо для тестування, але для та безпечної роботи в інтернеті, потрібно було зашифрувати трафік. Тому був налаштований Nginx як зворотній проксі-сервер, який передає запити до основного застосунку. Через нього реалізовано HTTPS – тобто усі дані між клієнтом і сервером передаються в зашифрованому вигляді. Це не тільки підвищує безпеку, а й дозволяє у майбутньому масштабувати систему, додаючи балансування навантаження або нові сервіси, на рисунку 2.1 зображена загальна схема взаємодії клієнта з сервером за допомогою WebSocket та Nginx. [5]

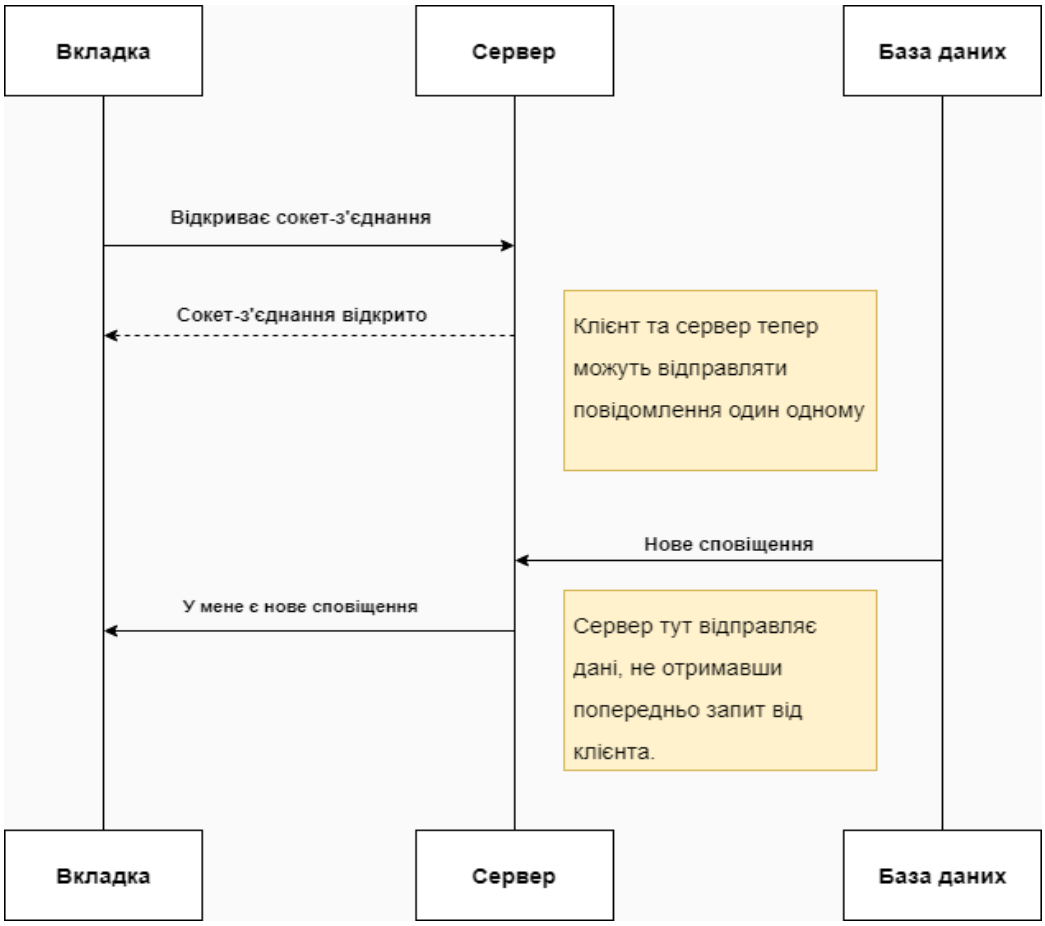

Рисунок 2.1 – Загальна схема взаємодії клієнта з сервером через WebSocket та Nginx

2.3 Клієнтське середовище для студента

Після аналізу різних операційних систем, які можна було б використовувати на комп’ютерах студентів, було вирішено зупинитися на Debian 11. Цей дистрибутив Linux вважається стабільним і не надто вимогливим до ресурсів. Однією з причин вибору саме Debian стало те, що він не оновлюється занадто часто і, відповідно не ламає щось зненацька. Порівняно з тим же Arch Linux або Fedora, у Debian зміни з’являються поступово, що дозволяє уникнути непередбачуваних багів. Також він має широку спільноту, багато інструкцій і рішень для типових проблем, що досить зручно.

Оскільки вручну налаштовувати середовище на кожному комп’ютері займе багато часу, було прийнято рішення, створювати віртуальні образи автоматично. Для цього використано Packer – утиліту, яка дозволяє описати, як має виглядати майбутня система, а потім швидко зібрати готовий образ для віртуальної машини.

Додатково виникло питання автентифікації студентів, які будуть працювати з системою. Щоб робити зайвих кроків, було вирішено використати вже вбудовану в Linux систему PAM (Pluggable Authentication Modules). Вона дозволяє налаштувати, як саме користувач буде входити в систему.

3 НАЛАШТУВАННЯ СЕРВЕРНОГО СЕРЕДОВИЩА

Для того, щоб система автоматизації перевірки практичних завдань могла працювати у хмарному середовищі, потрібно розгорнути сервер, встановити всі необхідні пакети та сервіси, а також налаштувати середовище для запуску веб-додатку. У цій частині описано процес налаштування серверної інфраструктури, який складається з двох основних етапів: створення інфраструктури за допомогою Terraform та налаштування самого сервера, за допомогою Ansible.[3]

3.1 Розгортання інфраструктури на AWS за допомогою Terraform

Необхідні умови: Перед початком роботи, необхідно переконатися, що:

1. На локальній машині встановлено Terraform;
2. Встановлено Ansible;
3. Є доступ до облікового запису AWS з правами на створення EC2, VPS, Security Group тощо;
4. Є базове розуміння роботи Terraform та Ansible;

3.1.1 Клонування репозиторію

Спочатку потрібно скопіювати репозиторій з конфігураціями для Terraform та Ansible рис.3.1, після цього перейти в робочу директорію рис. 3.2. [3]

```
ubuntu@ip-172-31-81-79:~$ git clone git@github.com:AndriiKostromytskyi/V-Mentor.git
Cloning into 'V-Mentor'...
remote: Enumerating objects: 92, done.
remote: Counting objects: 100% (92/92), done.
remote: Compressing objects: 100% (76/76), done.
remote: Total 92 (delta 5), reused 89 (delta 4), pack-reused 0 (from 0)
Receiving objects: 100% (92/92), 93.82 KiB | 9.38 MiB/s, done.
Resolving deltas: 100% (5/5), done.
```

Рисунок 3.1 – Клонування репозиторію

```
ubuntu@ip-172-31-81-79:~$ cd V-Mentor/
ubuntu@ip-172-31-81-79:~/V-Mentor$
```

Рисунок 3.2 – Робоча директорія

3.1.2 Ініціалізація Terraform

Тепер переходимо до каталогу з Terraform-конфігураціями для AWS і виконуємо ініціалізацію, за допомогою команди terraform init рис. 3.3. [3]

```
ubuntu@ip-172-31-81-79:~/V-Mentor/terraform_aws$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/aws versions matching "~> 4.0"...
- Installing hashicorp/aws v4.67.0...
- Installed hashicorp/aws v4.67.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

Рисунок 3.3 – Результат успішного виконання команди

3.1.3 Планування та створення інфраструктури

Перед створенням варто перевірити, що саме буде створено, наступною командою terraform plan, ця команда виведе нам те, що буде розгорнуто за допомогою Terraform рис. 3.4. [3]

```
ubuntu@ip-172-31-81-79:~/V-Mentor/terraform_aws$ terraform plan
data.aws_ami.ubuntu: Reading...
data.aws_ami.ubuntu: Read complete after 1s [id=ami-07182692443fccde1]

Terraform used the selected providers to generate the following execution
  + create

Terraform will perform the following actions:

  # aws_elb.ltiserv_lb will be created
  + resource "aws_elb" "ltiserv_lb" {
      + arn                         = (known after apply)
      + availability_zones          = (known after apply)
      + connection_draining         = false
      + connection_draining_timeout = 300
      + cross_zone_load_balancing   = true
      + desync_mitigation_mode      = "defensive"
      + dns_name                    = (known after apply)
      + id                          = (known after apply)
      + idle_timeout                = 60
      + instances                   = (known after apply)
      + internal                    = (known after apply)
      + name                        = "lti-lb"
      + security_groups             = (known after apply)
      + source_security_group       = (known after apply)
      + source_security_group_id    = (known after apply)
      + subnets                     = (known after apply)
```

Рисунок 3.4 – Результат виконання команди

Тепер можна запускати процес створення серверної інфраструктури за допомогою наступної команди: terraform apply (рис. 3.5) [3]

```
random_id.suffix: Creating...
random_id.suffix: Creation complete after 0s [id=yBA]
aws_elb.ltiserv_lb: Creating...
aws_elb.ltiserv_lb: Creation complete after 3s [id=lti-lb-c810]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

Outputs:

elb_dns_name = "lti-lb-c810-124350481.us-west-2.elb.amazonaws.com"
server_ip = "54.149.117.24"
```

Рисунок 3.5 – Успішне створення інфраструктури

3.2 Оновлення файлу inventory.ini для запуску Ansible

Після створення сервера, необхідно відредагувати файл inventory.ini, вказавши в ньому IP-адресу мого сервера(рис. 3.6)

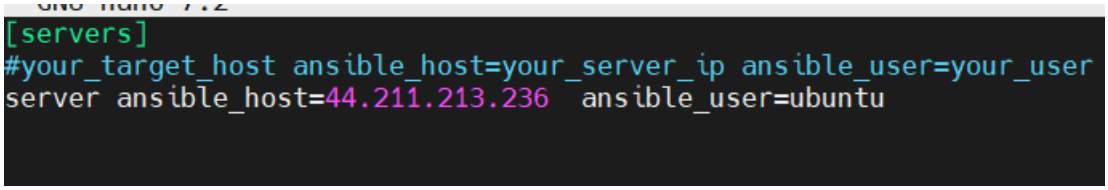

Рисунок 3.6 – Оновлений файл

3.3 Налаштування сервера за допомогою Ansible

Після оновлення файлу «інвенторі», можна переходити до автоматизованого налаштування сервера. Для цього слід повернутися до каталогу з Ansible та виконати наступну команду: ansible-playbook -I inventory.ini playbook.yaml (рис 3.7). [3]

Ця команда виконує playbook, який виконує наступні дії:

- встановлює системні залежності(Docker, Python, Nginx);
- копіює всі необхідні конфігурації;
- створює структуру каталогів для додатку;
- Розгортає проект у Docker-контейнері;
- Налаштовує веб-сервер для взаємодії з LTI-платформами.

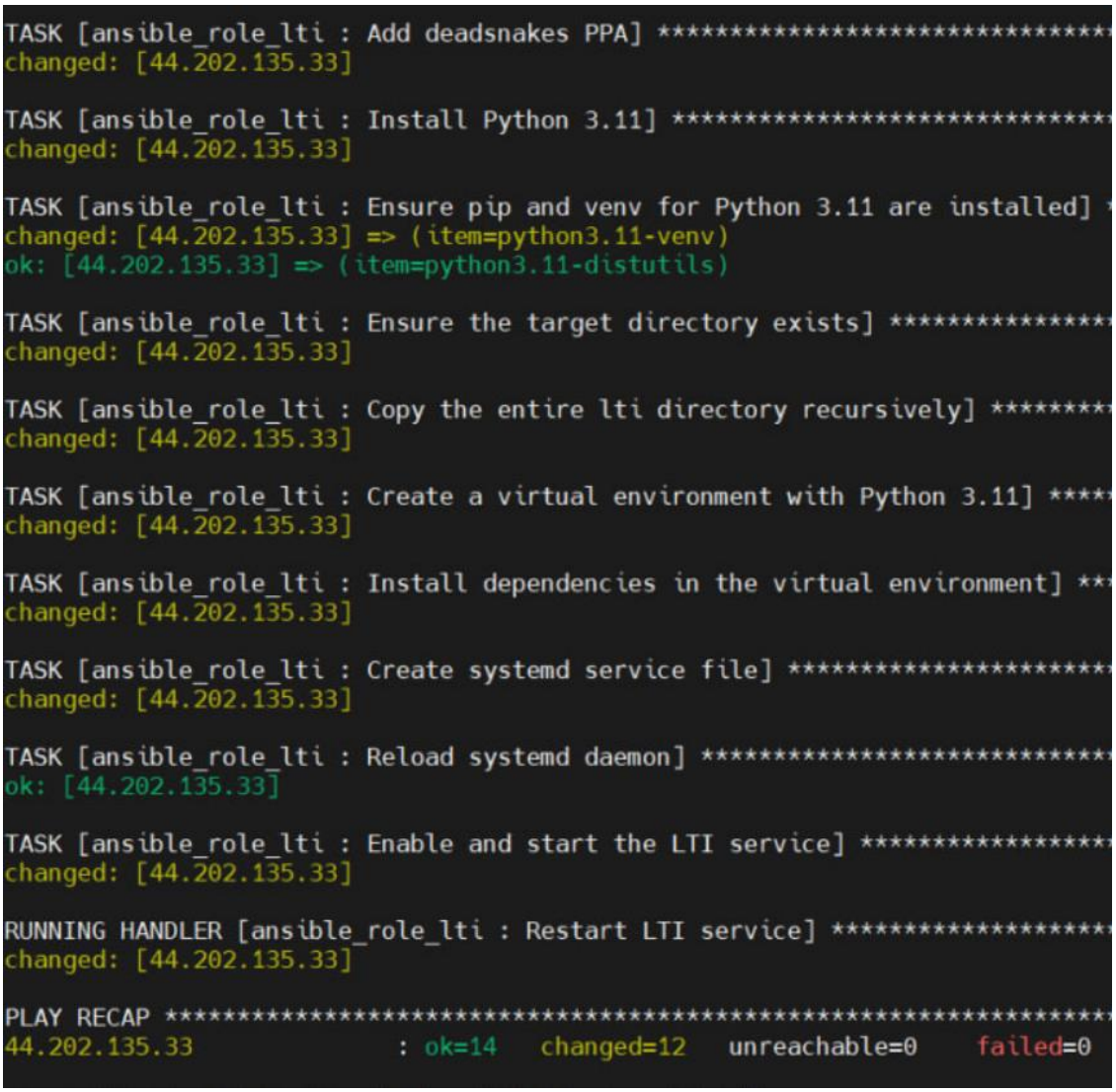

Рисунок 3.7 – Результат виконання команди ansible-playbook -I inventory.ini playbook.yaml

3.4 Обмеження доступу до серверу через відкриті порти для забезпечення безпеки.

Першим кроком для захисту серверу стало налаштування правил брандмауера (Security Group) на рівні AWS EC2. Відкрив тільки мінімально необхідні порти (рис. 3.8):

- порт 22 (SSH) – використовується лише для адміністративного доступу до сервера. Він відкритий лише для обмежених IP-адрес, а не для всього інтернету, що значно знижує ризик брутфорс-атак.

- порт 80 (HTTP) – відкритий для обробки веб-запитів користувачів.

- порт 443 (HTTPS) – потрібен для підключення SSL -сертифікатів.

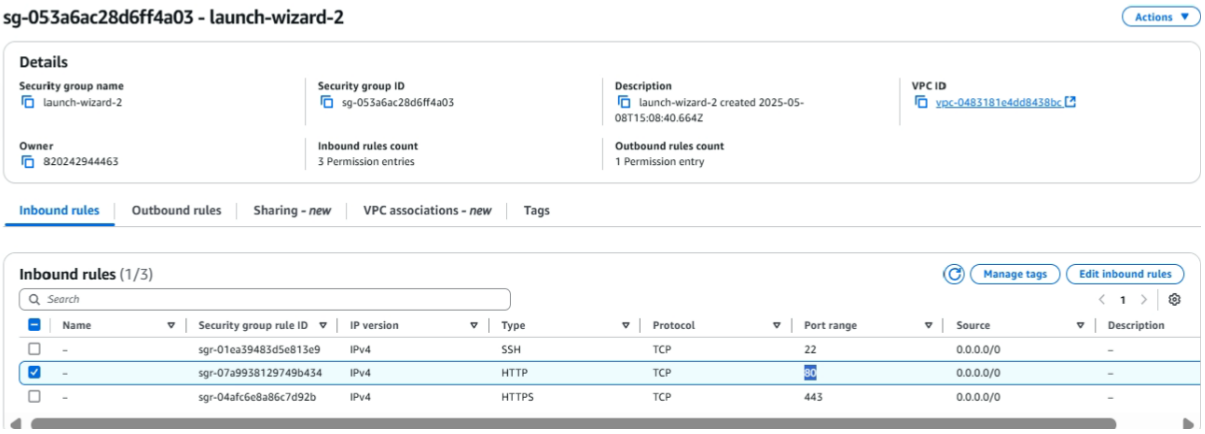

Рисунок 3.8 – Налаштовані порти для сервера

Усі інші порти, включно зі внутрішніми службовими, залишаються закритими. Таким чином, будь-який сторонній трафік, що не відповідає зазначеним портам, автоматично блокується на рівні мережевої інфраструктури.

Для доступу до сервера через SSH, використовується спеціальний приватний ключ у форматі .pem(рис. 3.9), який було згенеровано при створенні EC2-інстансу. Щоб уникнути несанкціонованого доступу, були дотримані наступні правила безпеки:

1) PEM-файл зберігається локально, лише на захищеній машині.

2) Було виставлено права доступу 400(chmod 400 diplom.pem), що дозволяє читати файл лише власнику і повністю виключає можливість редагування, або перегляду іншими користувачами.

3) Сам файл ніколи не передається через інтернет, або месенджери, навіть через зашифровані канали

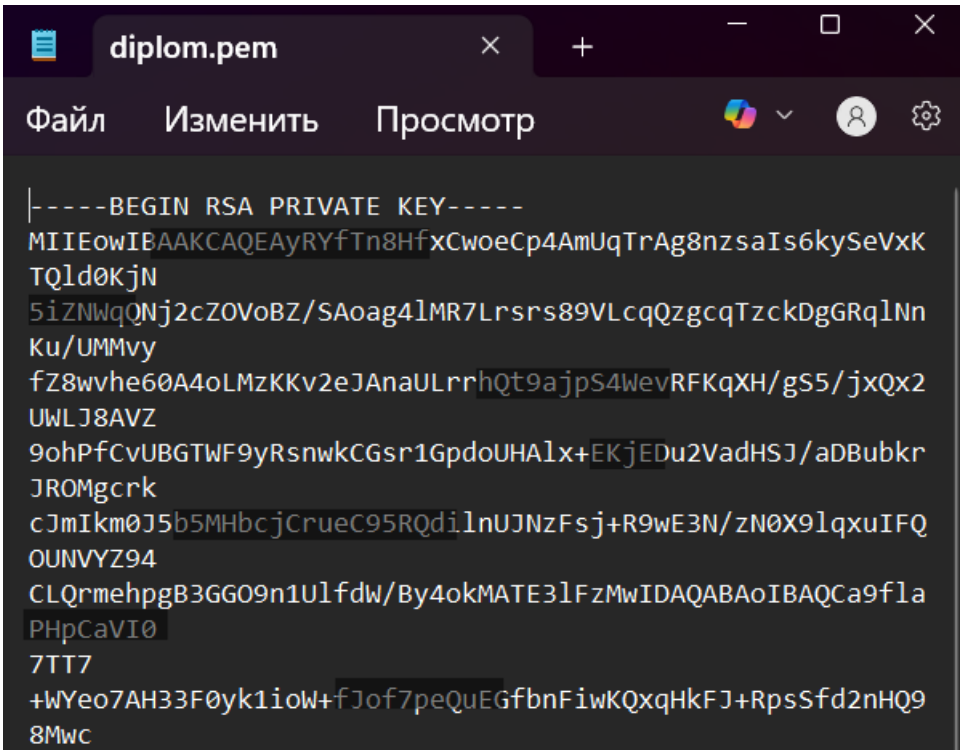

Рисунок 3.9 – Вигляд ключа PEM

4 НАЛАШТУВАННЯ ЗАВДАННЯ У MOODLE

4.1 Загальна характеристика інтеграції LTI 1.3

У процесі розробки системи для автоматизації перевірки завдань, одним із пріоритетних завдань, стало забезпечення її інтеграції з платформою Moodle. Для цього було використано стандарт LTI (Learning Tools Interoperability), який дозволяє безпечно поєднувати зовнішні сервіси з навчальними платформами. Завдяки цій технології стало можливим наступне:

1) Виконувати автоматичний вхід користувачів, без потреби додаткової реєстрації;
2) Передавати інформацію про роль користувача(тобто студент, або викладач);
3) Здійснювати зворотну передачу оцінок з зовнішньої системи без участі викладача.

4.2 Загальні відомості про інструмент

Було створено новий зовнішній інструмент, який підключається до сервісу LTI, розгорнутого на EC2-сервері AWS. Основні параметри інструменту, мають наступний вигляд рис. 4.1.

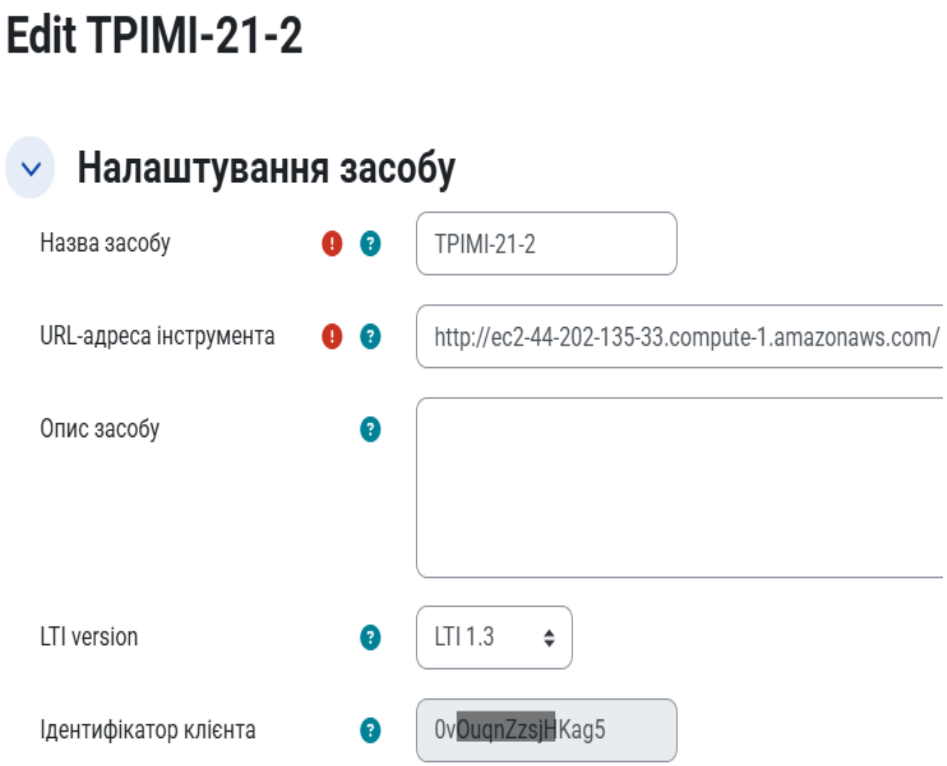

Рисунок 4.1 – Основні параметри

4.3 Верифікація через відкритий ключ

Для встановлення довірчих відносин між Moodle та сервером де розгорнутий проект, використовуються відкриті ключі. Система Moodle звертається до URL-адреси, де зберігається відкритий ключ у форматі JSON Web Key Set(JWKS) рис. 4.2.

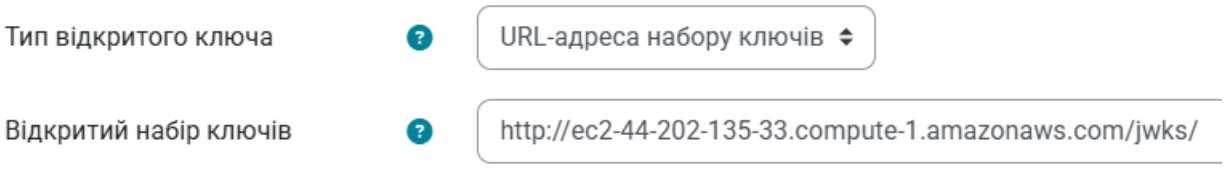

Рисунок 4.2 – Налаштування відкритих ключів

4.4 Авторизація та запуск інструмента

Ініціалізація входу виконується через спеціальну URL-адресу, яка запускає процес аутентифікації та передає токен з інформацією про користувача рис.4.3.

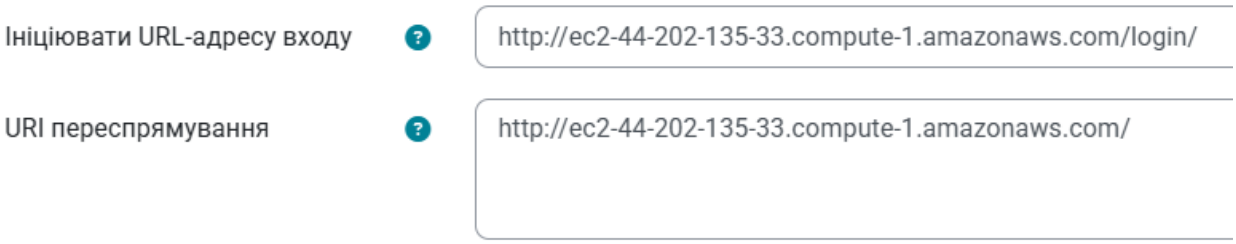

Рисунок 4.3 – Налаштування авторизації

4.5 Відображення інтерфейсу інструмента

Обрав наступний режим відкриття інструменту рис. 4.4.

Рисунок 4.4 – Обраний контейнер запуску

Це дозволяє відкривати систему у новій вкладці браузера, що зручно для користувача, оскільки це зберігає інтерфейс Moodle відкритим у попередній вкладці.

4.6 Додаткові сервіси LTI

Для повноцінної інтеграції було увімкнено підтримку кількох додаткових сервісів, що надаються в рамках LTI 1.3 рис. 4.5.

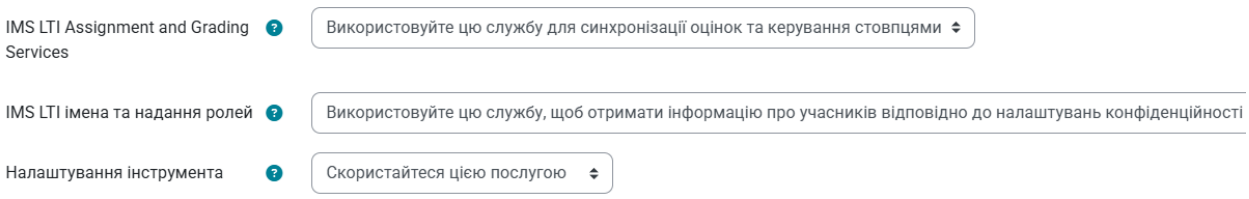

Рисунок 4.5 – Підключення додаткових сервісів

IMS LTI Assignment and Grading Services – дозволяє синхронізувати оцінки з зовнішнього сервісу безпосередньо в журналі оцінок Moodle.

IMS LTI імена та надання ролей – надає інформацію про зареєстрованих користувачів та їх ролі (студент, викладач), що важливо для правильного розмежування доступу в інструменті.

Tool Settings Services – використовується для передачі налаштувань між Moodle та інструментом.

4.7 Налаштування конфіденційності

Для коректної персоналізації інтерфейсу та ведення журналу результатів, увімкнув наступні дані рис. 4.6.

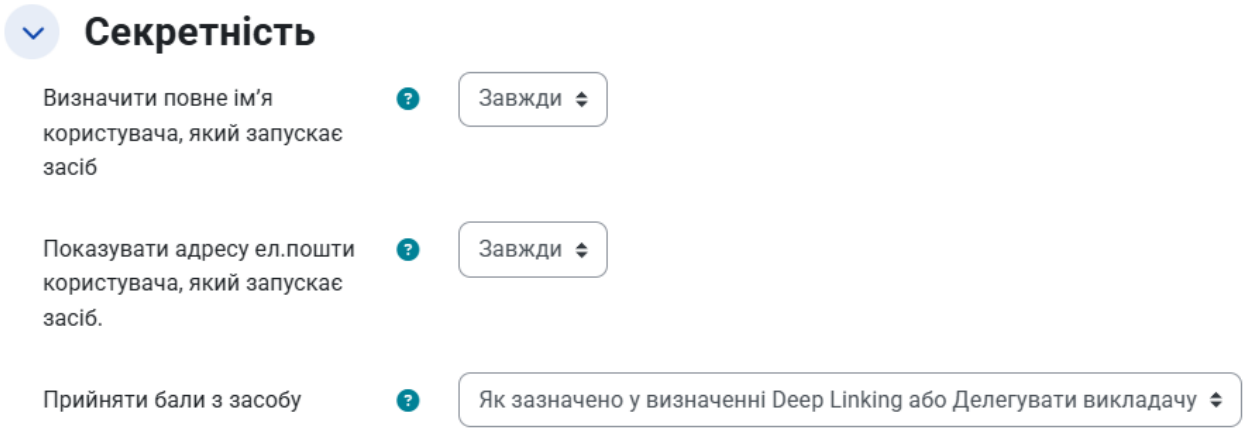

Рисунок 4.6 – Налаштування секретності

5 УСУНЕННЯ ПОМИЛОК ПІСЛЯ РОЗГОРТАННЯ ПРОЕКТУ

5.1 Помилка через відсутність доступу до сервісів LTI

На самому початку інтеграції LTI-інструменту з Moodle виникла наступна проблема – при спробі запуску, з’явилося повідомлення про «Недійсний запит» рис. 5.1. Це свідчило про те, що Moodle не зміг коректно ініціалізувати авторизацію з інструментом.

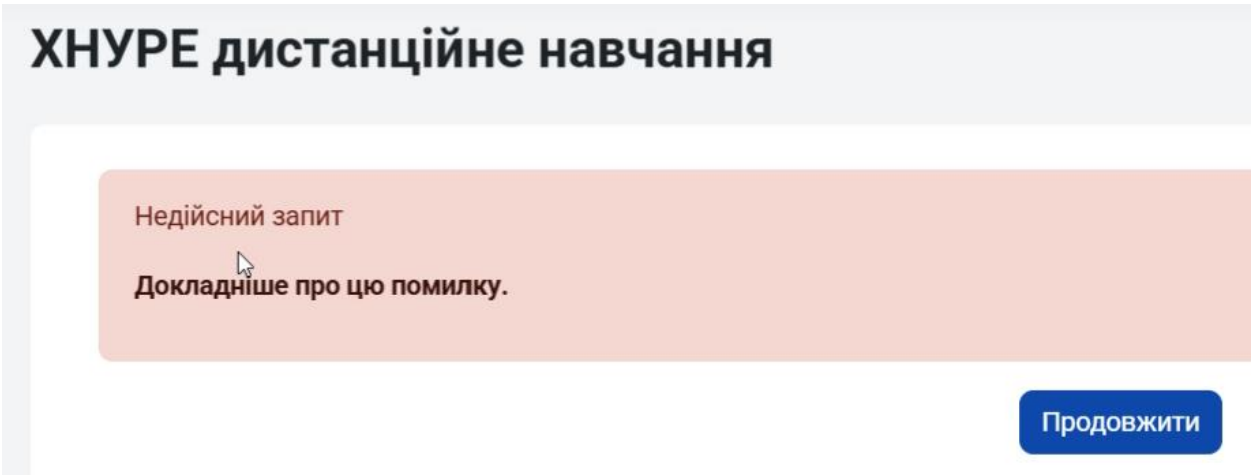

Рисунок 5.1 – Помилка запиту

Після аналізу конфігурації, з’ясував, що причиною є неактивовані сервіси LTI, які Moodle має використовувати для повноцінної взаємодії з зовнішнім інструментом рис. 5.2. На момент виникнення проблеми в налаштуваннях усі служби LTI були вимкнені.

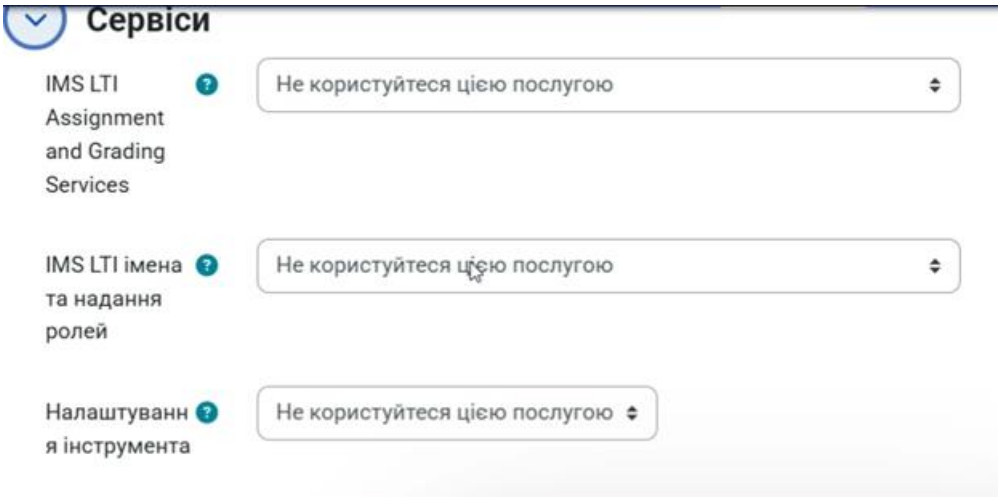

Рисунок 5.2 – Налаштування за замовчуванням

- IMS LTI Assignment and Grading Services – встановлено наступне: «використовуйте цю службу для синхронізації оцінок»
- IMS LTI імена та надання ролей – встановлено: «використовуйте цю службу, щоб отримати інформацію»
- Налаштування інструмента – встановлено: «Скористайтеся цією послугою» рис. 5.3.

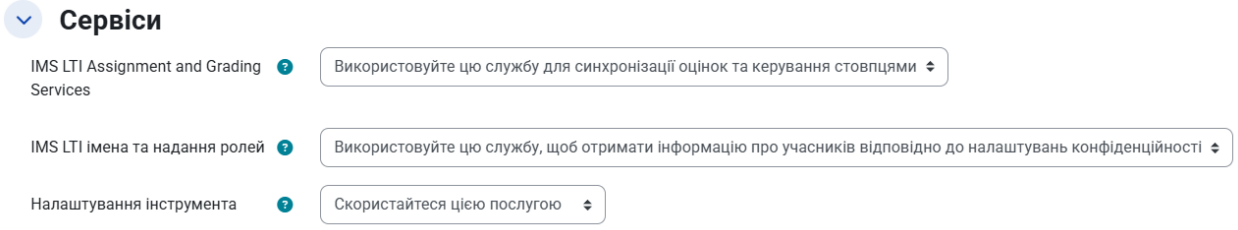

Рисунок 5.3 – Зміни в налаштуванні

5.2 Помилка запуску LTI-інструменту в Moodle

Тепер виникла помилка (рис. 5.4) запуску LTI-інструменту в Moodle, це вказує на те, що сервер інструменту не знайшов конфігурації для client_id, які Moodle надсилає при ініціалізації запуску. Щоб усунути цю проблему, я зробив наступне налаштування конфігураційного JSON-файлу на сервері інструменту.

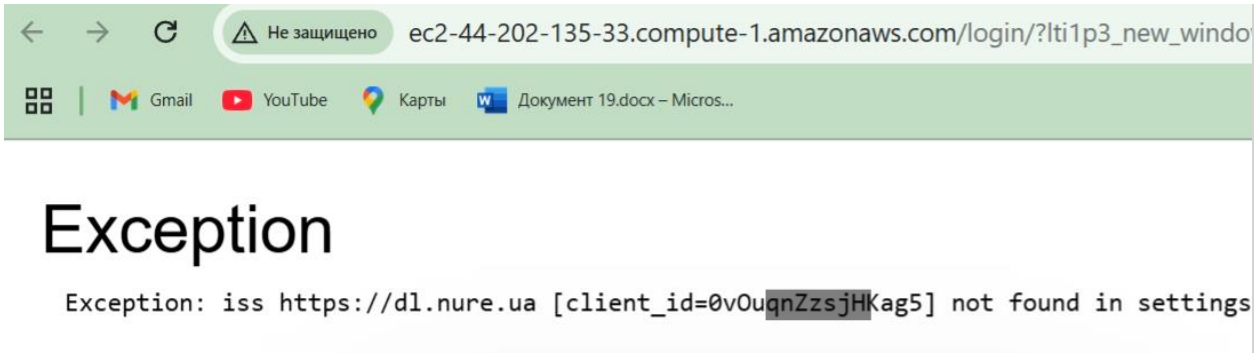

Рисунок 5.4 – Помилка, яка виникає після запуску LTI

1) Перейшов до конфігураційного файлу рис. 5.5.

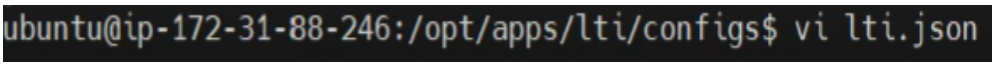

Рисунок 5.5 – Перехід в конфігурацію lti.json

2) Зміна поточної конфігурації

Проблема полягала в тому, що активною (“default”:true) була неправильна секція, де client_id не відповідав тому, що Moodle намагався використати.

Помилку виправив наступним чином:

Знайшов секцію, в якій client_id відповідає тому на який йде запит, встановив “default”: true для цієї секції. Для інших секцій “default” змінив на

false. Також відредагував поле “deployments_ids”, а саме, замінів ID з [“7”] на [“8”] оскільки Moodle надсилає саме його. рис. 5.6. [6]

Рисунок 5.6 – Фінальний вигляд конфігурації

3) Перезапуск сервісу

Після внесення змін, конфігураційний файл було збережено і виконано перезапуск сервісу.

```
ubuntu@ip-172-31-88-246:/opt/apps/lti/configs$ sudo systemctl restart
```

Рисунок 5.6 – Перезапуск сервісу

5.3 Помилка верифікації SSL-сертифікатів при звернені до LTI-сервісу

Після виправлення попередньої помилки, з’явилася помилка, яка вказує на те, що не вдалося отримати відкриті ключі з Moodle. Також помилка вказує, що верифікувати сертифікат сервера не вдалося рис. 5.7.

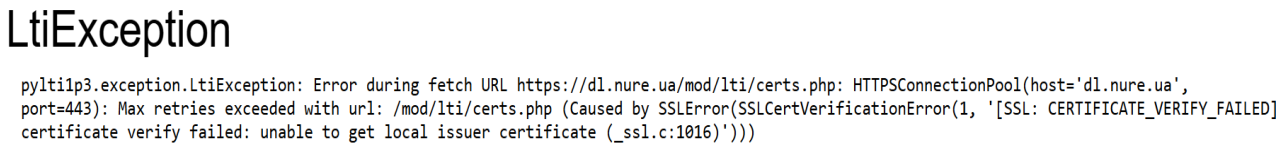

Рисунок 5.7 – Помилка підключення до LTI-сервісу

Причина полягала в тому, що серверна частина інструменту не мала доступу до повного ланцюга довіри (CA). Детально проаналізувавши помилку, зрозумів, що проблема полягала не в сертифікатах, а в тому, що сервер Moodle, повертав HTML-сторінку із капчею, замість очікуваного JSON з ключами, тобто причиною стало те, що на сервері де був розгорнутий LTI-інструмент, використовувалася динамічна адреса, оскільки сервер безкоштовний, через це сервер нашого Moodle сприймав кожне нове підключення як підозріле, що активувало Captcha.oj w Цей механізм був реалізований адміністраторами для захисту від можливих DDoS-атак. Щоб вирішити цю проблему, створив еластичну IP-адресу(постійну зовнішню IP) в інфраструктурі AWS. Далі звернувся до адміністрації Moodle з проханням додати цю адресу до переліку виключень.

5.4 Помилка підключення до бази даних

Після усунення проблеми з капчею, виникла наступна помилка(рис. 5.8)

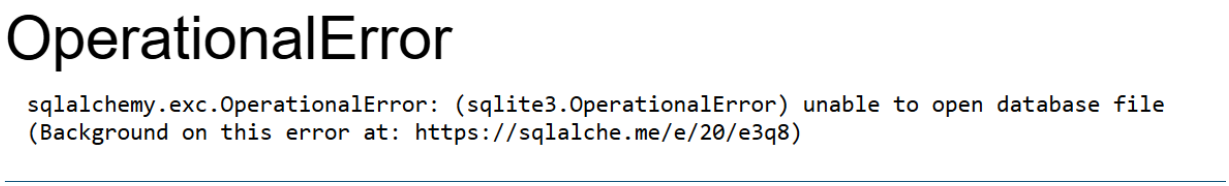

Рисунок 5.7 – Помилка при підключенні до бази даних

Ця помилка вказувала на неможливість відкриття файлу бази даних SQLite, яка в проекті використовується для зберігання структури та даних про тестові завдання. Нажаль, файл після розгортання інфраструктури був не доступний, оскільки не актуалізували зміни до Ansible. Для вирішення цієї проблеми, спочатку перевірив структуру проекту, а також доступність файлу бази даних у зазначеному шляху, в результаті виявив, що база даних, яка мала б знаходитися у директорії /var/lti/, фізично існує, але доступ до неї був не можливий. Для виправлення помилки, вирішив повністю перевстановити базу, використовуючи систему керування міграціями Flask-Migrate. Попередньо зупинив роботу застосунку, після чого вручну видалив наявний файл бази даних. Далі у середовищі app.py, виконав повторне застосування міграцій за допомогою команди flask db upgrade. Це дозволило створити нову структуру бази даних, відповідно до актуальної версії схеми, яка зберігається у каталозі міграцій рис. 5.8-5.9. [6]

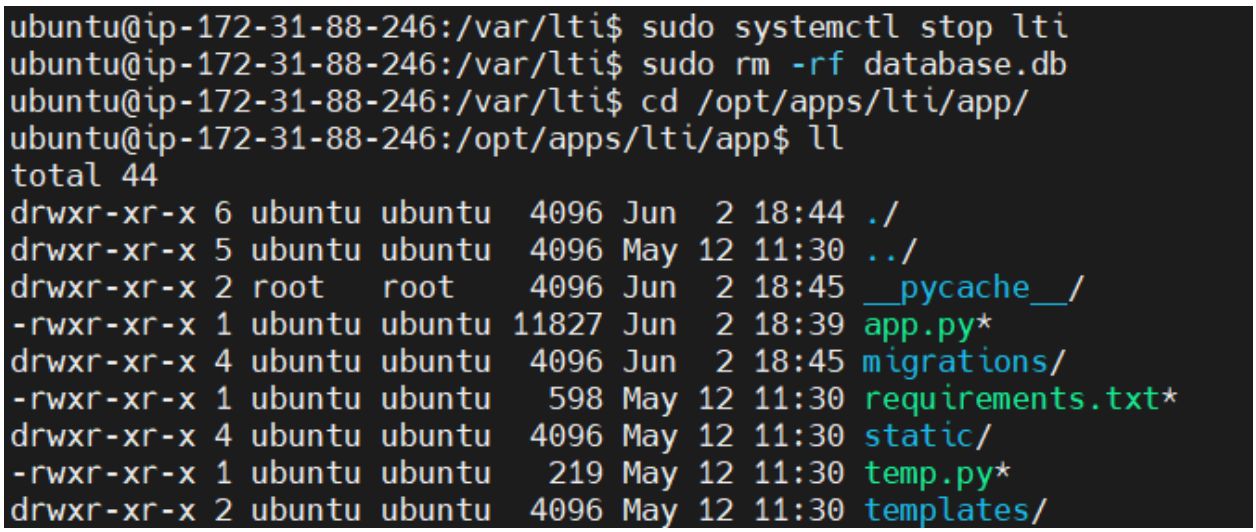

Рисунок 5.8 – Процес виправлення помилки

```
ubuntu@ip-172-31-88-246:/opt/apps/lti/app$ sudo ../env/bin/python -m flask db upgrade
INFO  [alembic.runtime.migration] Context impl SQLiteImpl.
INFO  [alembic.runtime.migration] Will assume non-transactional DDL.
INFO  [alembic.runtime.migration] Running upgrade  -> e44f4344f381, Initial migration
ubuntu@ip-172-31-88-246:/opt/apps/lti/app$ sudo ../env/bin/python -m flask db current
INFO  [alembic.runtime.migration] Context impl SQLiteImpl.
INFO  [alembic.runtime.migration] Will assume non-transactional DDL.
e44f4344f381 (head)
ubuntu@ip-172-31-88-246:/opt/apps/lti/app$ sqlite3 /var/lti/database.db
SQLite version 3.45.1 2024-01-30 16:01:20
Enter ".help" for usage hints.
sqlite> .tables
alembic_version  task             test
sqlite> select * from alembic_version;
e44f4344f381
sqlite>
```

Рисунок 5.9 – Процес виправлення помилки

Після цього сервіс було перезапущено і проект став повноцінно працювати.

5.5 Інтерфейс після успішного запуску системи

Після успішного усунення помилок і налаштування всіх компонентів, система була успішно розгорнута і запущена. Після авторизації під обліковим записом студента, користувач потрапляє на головну сторінку інтерфейсу, де відображається список доступних завдань рис. 5.10.

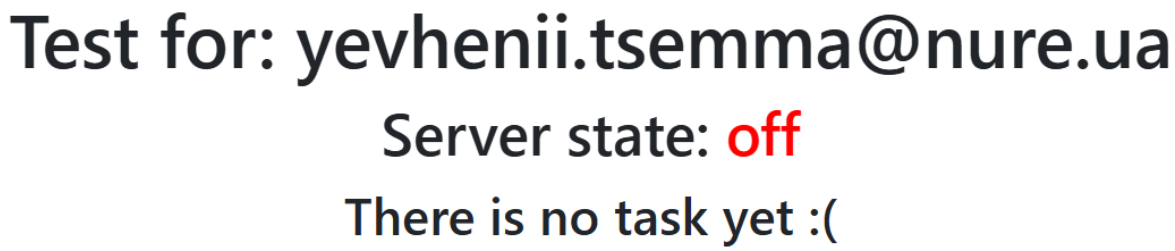

Рисунок 5.10 – Інтерфейс після авторизації студента

Якщо увійти під роллю викладача, система надає додаткові можливості, тобто надає доступ до додаткових функцій де можна завантажити тест, написати йому назву, а також можливості перегляду результатів тестів рис.

5.11. Завдяки цьому, забезпечується зручне управління процесом оцінювання без необхідності виходити за межі LMS.

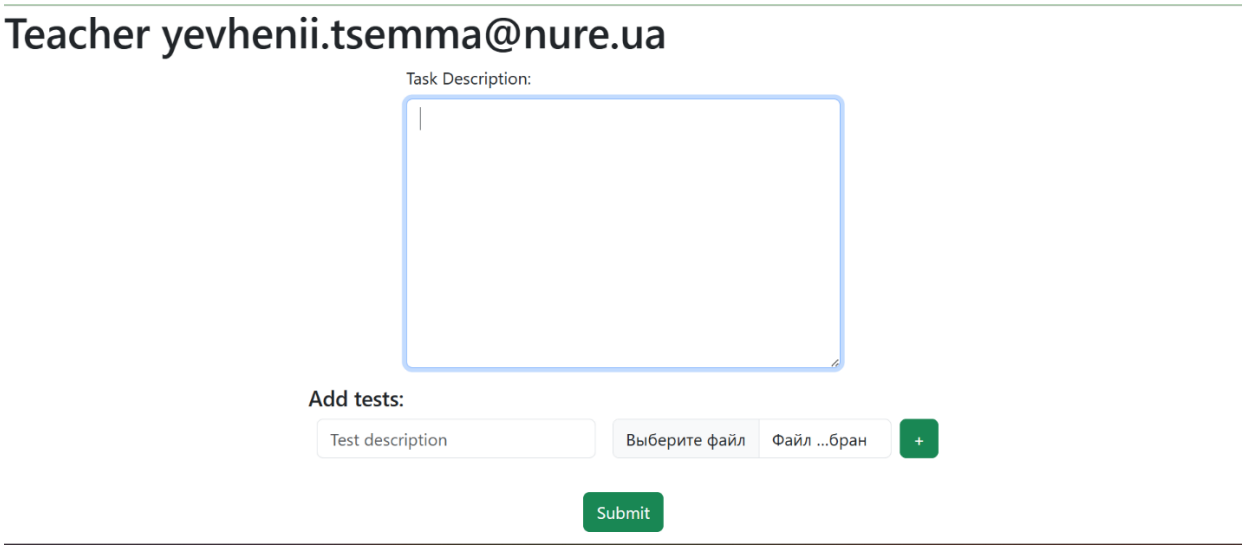

Рисунок 5.11 – Інтерфейс після авторизації викладача

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи, розгорнув та актуалізував повноцінну систему автоматизації перевірки практичних завдань студентів, яка дозволяє ефективно організувати навчальний процес у програмуванні та адмініструванні. Актуальність цього проекту обумовлена потребою в оптимізації навчального процесу, особливо умовах дистанційної освіти, оскільки важливо забезпечити швидкий зворотній зв'язок між студентом і викладачем.

В цій роботі було поєднано сучасні інструменти, наприклад серверна частина, створена на базі Flask, використано WebSocket для інтерактивного обміну повідомленнями, Docker для створення ізольованих середовищ перевірки коду, а також Terraform і Ansible для автоматизації розгортання інфраструктури в хмарі AWS. Крім того, було реалізовано підтримку LTI 1.3, яка дає можливість інтегрувати систему з Moodle нашого університету. В результаті, отримав стабільно працюючу систему, яка спрощує онлайн-навчання.

З переваг реалізованого рішення, можна виділити повну автоматизацію перевірки, це дозволяє викладачу зосередитися на якості викладання, а не на рутинній перевірці коду. Також, інструмент легко масштабується завдяки хмарній інфраструктурі, а використання таких стандартів, як LTI, WebSocket робить його універсальним і придатним до подальшого розвитку.

Для вдосконалення проекту, варто реалізувати веб-інтерфейс для керування тестами та перегляду логів, розширити типи завдань, які можна перевіряти. Також можливо адаптувати інструмент з іншими платформами навчання, окрім Moodle, оскільки протокол LTI це дозволяє.

Усі завдання, поставлені в технічному завданні, виконав в повному обсязі. Отриманий результат може застосовуватися у реальному навчальному процесі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Документація до LMS Moodle [Електронний ресурс] Режим доступу до ресурсу: https://docs.moodle.org/403/en/Main_page

2. Специфікація LTI [Електронний ресурс] Режим доступу до ресурсу: https://www.imsglobal.org/spec/lti/v1p3/impl/

3. DigitalOcean. How To Use Ansible with Terraform for Configuration Management[Електронний ресурс] // DigitalOcean Community. – 2022. – Режим доступу: https://www.digitalocean.com/community/tutorials/how-to-use-ansible-with-terraform-for-configuration-management

4. Що таке LTI: https://www.overnitecbt.com/2019/02/15/what-is-an-lms-and-how-can-it-help-your-business/

5.DevZone. Масштабування WebSocket-з'єднань за допомогою роздільних воркерів [Електронний ресурс] // Devzone.– 2023. – Режим доступу: https://devzone.org.ua/post/masshtabuvannia-websocket-zyednan-za-dopomohoiu-rozdiliuvanykh-vorkeriv

6. Ubuntu Server Documentation[Електронний ресурс]. – Canonical Ltd. – Режим доступу: https://ubuntu.com/server/docs