Харківський національний університет радіоелектроніки
(повне найменування вищого навчального закладу)

Факультет *інфокомунікацій*
(повна назва)

Кафедра *інформаційно-мережної інженерії*
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

на тему *Дослідження шляхів автоматизації перевірки виконання практичних завдань в локальному віртуальному оточенні користувача. Розробка API*

Виконав: студент *2* курсу, групи *ІМІм-22-1*
напряму підготовки
172 "Телекомунікації і радіотехніка"
(шифр і назва напряму підготовки)
Таран М.В.
(прізвище та ініціали)
Керівник *Костромицький А.І.*
(прізвище та ініціали)

Допускається до захисту

Зав. кафедри _____ *Безрук В.М.*
(підпис) (прізвище, ініціали)

Харків - 2024 рік

Не містить відомостей, заборонених до відкритого публікування

Студент _____

Керівник_____

Харківський національний університет радіоелектроніки
(повне найменування вищого навчального закладу)

Факультет *інфокомунікацій*

Кафедра *інформаційно-мережної інженерії*

Рівень вищої освіти *другий (магістерський)*

Напрям підготовки *172 «Телекомунікації і радіотехніка»*
(шифр і назва)

ЗАТВЕРДЖУЮ

Зав.кафедри _____
(Підпис)

“____” _____ 2023 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Тарану Максиму Вадимовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Дослідження шляхів автоматизації перевірки виконання практичних завдань в локальному віртуальному оточенні користувача. Розробка API*

затверджені наказом ВНЗ від “ *23* ” *жовтня* 2023 року № *1233 Ст.*

2. Строк подання студентом роботи *10 січня 2024 року*

3. Вихідні дані до роботи *Провести дослідження існуючих систем для автоматизованої перевірки завдань та проаналізувати їх сильні та слабкі сторони. Розробити прототип системи для автоматизованої перевірки завдань у віртуальному середовищі учня. Розробити API для комунікації різних модулів системи.*

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Вступ

1. Огляд систем керування навчанням

2. Проєктування архітектури та API

3. Розробка системи автоматизованої перевірки завдань

Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Слайди у форматі Power Point (назва та мета роботи, огляд, переваги та недоліки LMS, актуальність, проєктування, вибір архітектури, середовище учня, можливості розвитку проєкту, висновки)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Основна частина	*доц. Костромицький А.І.*	23.10.23	

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	*Ознайомлення із завданням. Уточнення ТЗ.*	*23.10.23*	
2	*Підбір літератури за темою роботи.*	*24.10-26.10.23*	
3	*Виконання розділу 1*	*27.10-05.11.23*	
4	*Виконання розділу 2*	*06.11-20.11.23*	
5	*Виконання розділу 3*	*21.11-25.12.23*	
6	*Оформлення презентаційного матеріалу,*	*26.12.23-10.01.24*	
	підготовка до захисту у ЕК		

Дата видачі завдання *23* *жовтня* *2023 р.*

Студент _____ *Таран М.В.*
(підпис) (прізвище та ініціали)

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

РЕФЕРАТ

Пояснювальна записка: 55 с., 11 рис., 7 джерел

Мета роботи – автоматизація перевірки виконання практичних завдань, у локальному середовищі учня.

Сучасний світ стикається з численними викликами, одним із найбільших з яких є необхідність впровадження ефективних форм та засобів віддаленого навчання. Але це питання тягне за собою численні проблеми, такі як низька співпраця між студентами і викладачами, важкість відслідковування академічного прогресу і необхідність забезпечити доступність матеріалів. Для вирішення проблем віддаленого навчання і забезпечення ефективної освіти, використання систем управління навчанням стає критично важливим питанням.

LMS, LTI, LEARNING AUTOMATION, СИСТЕМА УПРАВЛІННЯ НАВЧАННЯМ

ABSTRACT

Explanatory note: 55 p., 11 fig., 7 sources.

The object of study is an automation of learning process with custom virtual machine image.

The modern world faces many challenges related to the transformation of the education system at all levels. One of the most serious and urgent issues is the provision of comprehensive, accessible and quality tools for distance learning. While various learning management systems (LMS) have long been successfully used to solve the issues of conducting classes with an instructor, delivering various content, conducting tests, and controlling the learning process, to test the practical skills of students often requires the development and use of specialized "sandboxes". Unfortunately, the solutions available on the market have a number of limitations that do not allow their effective integration into existing training programs.

LMS, LTI, LEARNING AUTOMATION, СИСТЕМА УПРАВЛІННЯ НАВЧАННЯМ

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ 6
ВСТУП 7
1 ОГЛЯД СИСТЕМ КЕРУВАННЯ НАВЧАННЯМ 8
1.1 Що таке LMS? 8
1.2 Переваги та недоліки LMS 9
1.3 Актуальність 9
2. ПРОЄКТУВАННЯ АРХІТЕКТУРИ ТА API 11
2.1 Архітектура проєкту 11
2.2 Канал зв’язку між клієнтом та сервером 15
2.3 Будова серверної частини системи 17
2.4 Інструмент для перевірки завдань 17
2.5 Локальне середовище учня 20
3 РОЗРОБКА СИСТЕМИ АВТОМАТИЗОВАНОЇ ПЕРЕВІРКИ ЗАВДАНЬ 22
3.1 Модуль автентифікації користувача 22
3.2 Клієнтський рушій 23
3.3 Збірка образу віртуальної машини 24
3.4 Робота з віртуальним середовищем 25
3.5 Можливості вдосконалення 30
ВИСНОВКИ 31
ДОДАТОК А СЛАЙДИ ПРЕЗЕНТАЦІЇ 33
ДОДАТОК Б ТЕЗИ ДОПОВІДІ НА КОНФЕРЕНЦІЇ 42
ДОДАТОК В ЛІСТИНГ ПРОГРАМНОГО КОДУ 44

ПЕРЕЛІК СКОРОЧЕНЬ

API – Application Programming Interface

IAM – Identity and Access Management

LDAP – Lightweight Directory Access Protocol

LMS – Learning Management System

LTI – Learning Tools Interoperability

OIDC – OpenID Connect

PAM – Privileged Access Management

ВСТУП

Learning Management System (LMS) – це спеціалізовані платформи, які надають освітнім установам і організаціям засоби для організації, ведення і відстеження навчального процесу та ресурсів. Вони з'явилися як результат поєднання технологічних можливостей з потребами сучасної освіти. Вони пройшли довгий шлях від перших експериментів до потужних та інтегрованих платформ, що забезпечують доступ до якісної освіти у будь-якому місці і в будь-який час.

Потреба у LMS у сучасному світі визначається необхідністю забезпечити доступ до якісної освіти незалежно від географічного положення та обставин. Вони сприяють зростанню доступності освіти, покращенню якості навчання і створюють механізми для ефективної взаємодії між учасниками освітнього процесу, що є ключовими компонентами сучасної освіти.

1 ОГЛЯД СИСТЕМ КЕРУВАННЯ НАВЧАННЯМ

1.1 Що таке LMS?

Learning Management System (LMS) - це програмний продукт для адміністрування навчальних курсів в рамках дистанційного навчання. Подібні системи з'явилися як результат поєднання технологічних можливостей з потребами сучасної освіти. Вони пройшли довгий шлях, від перших експериментів до потужних та інтегрованих платформ, що забезпечують доступ до якісної освіти у будь-якому місці і в будь-який час.

Рисунок 1.1 – Графічне представлення поняття LMS

На даний момент LMS – це невід’ємна частину нашого життя, допомагаючи постійно дізнаватись щось нове, та опановувати все нові навички, які допомагають нам як в роботі, так і в повсякденному житті.

1.2 Переваги та недоліки LMS

Переваги:

– легко зберігати, оновлювати та розповсюджувати навчальні матеріали, зменшуючи витрати на друк і фізичне зберігання;

– легко відстежувати прогрес студентів, автоматизувати оцінювання і забезпечувати ведення обліку навчальних результатів;

– дозволяють студентам і викладачам отримувати доступ до навчальних матеріалів і завдань з будь-якого місця та у будь-який час, що особливо важливо в умовах віддаленого навчання;

– дозволяють налаштовувати навчальний процес для кожного студента окремо, створюючи індивідуальні плани навчання та надаючи рекомендації на основі академічного прогресу;

Недоліки:

– віддалене навчання побудоване навколо LMS може призвести до відсутності особистого контакту між студентами і викладачами;

– вимагає від студентів більшої самостійності та самодисципліни, що не завжди може бути досягнутим.

1.3 Актуальність

Розробка системи на основі LMS з автоматичною перевіркою якості виконання завдань є актуальною і доцільною задачею в сучасному освітньому середовищі. Сучасні технології та зростаюча конкуренція вимагають від університетів та освітніх закладів ефективних інструментів для підвищення якості навчання. Системи LMS, поєднані з автоматизованими засобами перевірки завдань, дозволяють створити інтерактивне середовище для студентів, де вони можуть не лише здобувати знання, а й отримувати негайний

зворотний зв'язок щодо свого навчального прогресу. Це сприяє покращенню академічних досягнень студентів і розвиває навички самоорганізації та відповідальності. Крім того, автоматична перевірка завдань зменшує навантаження на викладачів і дозволяє їм більше уваги приділяти індивідуальному супроводу студентів. Така система підтримує створення ефективного і забезпеченого якістю навчання, що є критичним у сучасному освітньому ландшафті. Додатково, система на основі LMS з автоматичною перевіркою завдань спростить процес оцінювання та аналізу навчальних результатів. Викладачі зможуть швидко отримувати доступ до статистичних даних про виконання завдань всіма студентами і вчасно реагувати на індивідуальні потреби студентів або виявляти загальні тенденції навчання. Ця система також сприяє створенню об'єктивних критеріїв оцінювання та зменшує можливість суб'єктивних оцінок. Усе це робить розробку системи LMS з автоматичною перевіркою завдань не лише актуальною, але і важливою задачею для вдосконалення процесу навчання та підвищення якості освіти.

2. ПРОЄКТУВАННЯ АРХІТЕКТУРИ ТА API

2.1 Архітектура проєкту

Існує два основних підходи до створення архітектури програмного забезпечення - це монолітний та мікросервісний, наведені на рис. 2.1.

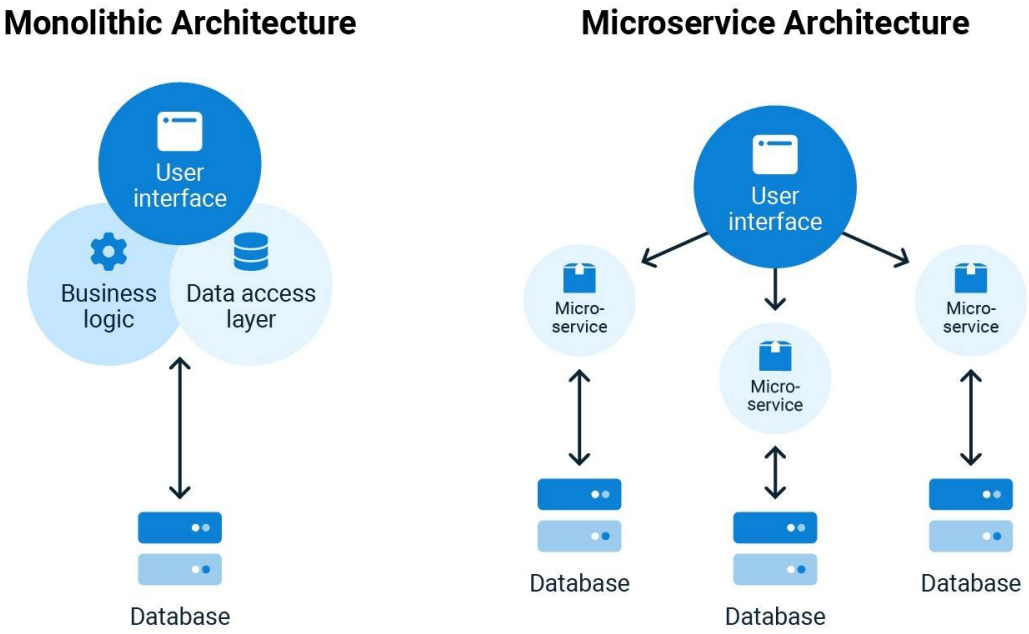

Рисунок 2.1 – Візуальне зображення монолітної та мікросервісної архітектур

Монолітна архітектура – це підхід до розробки програмного забезпечення, в якому вся програма побудована як єдиний, компактний блок. У такій архітектурі весь функціонал, та інтерфейс користувача зазвичай вбудовані в єдиний застосунок.

Переваги:

– відносна простота розробки, оскільки всі компоненти знаходяться в одному місці;

– тестування монолітів зазвичай легше і дешевше, оскільки весь код належить до одного застосунку;

– відносно просто “розгорнути” застосунок за рахунок того що не треба думати про комунікацію між окремими модулями програми.

Недоліки:

– з ростом проєкту стає все складніше підтримувати застосунок, оскільки всі компоненти залежні один від одного і невеличкі зміни у програмі можуть потягнути за собою проблеми одразу у багатьох місцях;

– дуже складно масштабувати монолітну програму оскільки з ростом попиту треба збільшувати потужності сервера на якому застосунок працює.

Мікросервісна архітектура – це підхід до розробки програмного забезпечення, в якому додаток розділений на невеликі незалежні мікросервіси, які працюють як окремі компоненти та взаємодіють через мережу. Кожен мікросервіс виконує обмежену функцію і може бути розроблений, встановлений, тестований і масштабований окремо.

Переваги:

– легке масштабування окремих компонентів що дозволяє швидко та зручно підлаштовуватися під зміни у попиті до ресурсів;

– кожен мікросервіс може розроблюватися окремо один від одного та навіть різними мовами програмування що додає гнучкості у питанні підтримки коду;

Недоліки:

– комунікація між мікросервісами може бути досить реалізувати у окремих випадках.

Поєднання цих двох фундаментально різних підходів до проєктування архітектури призвело до створення гібридної архітектури, що дозволило поєднати переваги обидвох підходів.

Основою всього проєкту є LMS. Вона дає основний функціонал, довкола якого буде базуватись система автоматизації перевірки практичних завдань. LMS надає функціонал створення курсів, завдань, ресурсів для навчання, а також гнучке та комфортне керування ними.

Вже існує достатньо таких систем у відкритому доступі, найпопулярніші з них це Moodle та Open edX. Вибір припав саме на Moodle, і на це є декілька причин:

1. У зв’язку з високою популярністю є багато різних матеріалів з описом процесу налаштування та розгортання системи.
2. При виникненні проблем або запитань можна звернутися на тематичні форуми.
3. Багатий функціонал забезпечує гнучкість та зручність у використанні.
4. Можливість налаштування авторизації за допомогою пошти. Це дозволить створити наскрізний ключ, який дозволить ідентифікувати людину всередині нашої системи та забезпечить безпеку для даних.

Гібридна архітектура поєднує у собі переваги монолітної та мікросервісної архітектури. Це дозволяє розробникам використовувати монолітну архітектуру для стабільних і основних компонентів застосунку, а мікросервіси для більш гнучких та швидкозмінних функціональних можливостей. Використання гібридної архітектури дозволяє балансувати незалежність і легкість оновлення або розвитку програмного коду. Саме тому нами було обрано гібридну архітектуру для нашої роботи.

За основу для проєкту було вирішено використати готову LMS. Таке рішення має багато переваг, таких як, збереження часу відведеного на

розробку та можливість додати підтримку інших LMS у майбутньому. Після порівняння декількох готових систем ми зупинилися на популярній системі керування навчанням – Moodle.

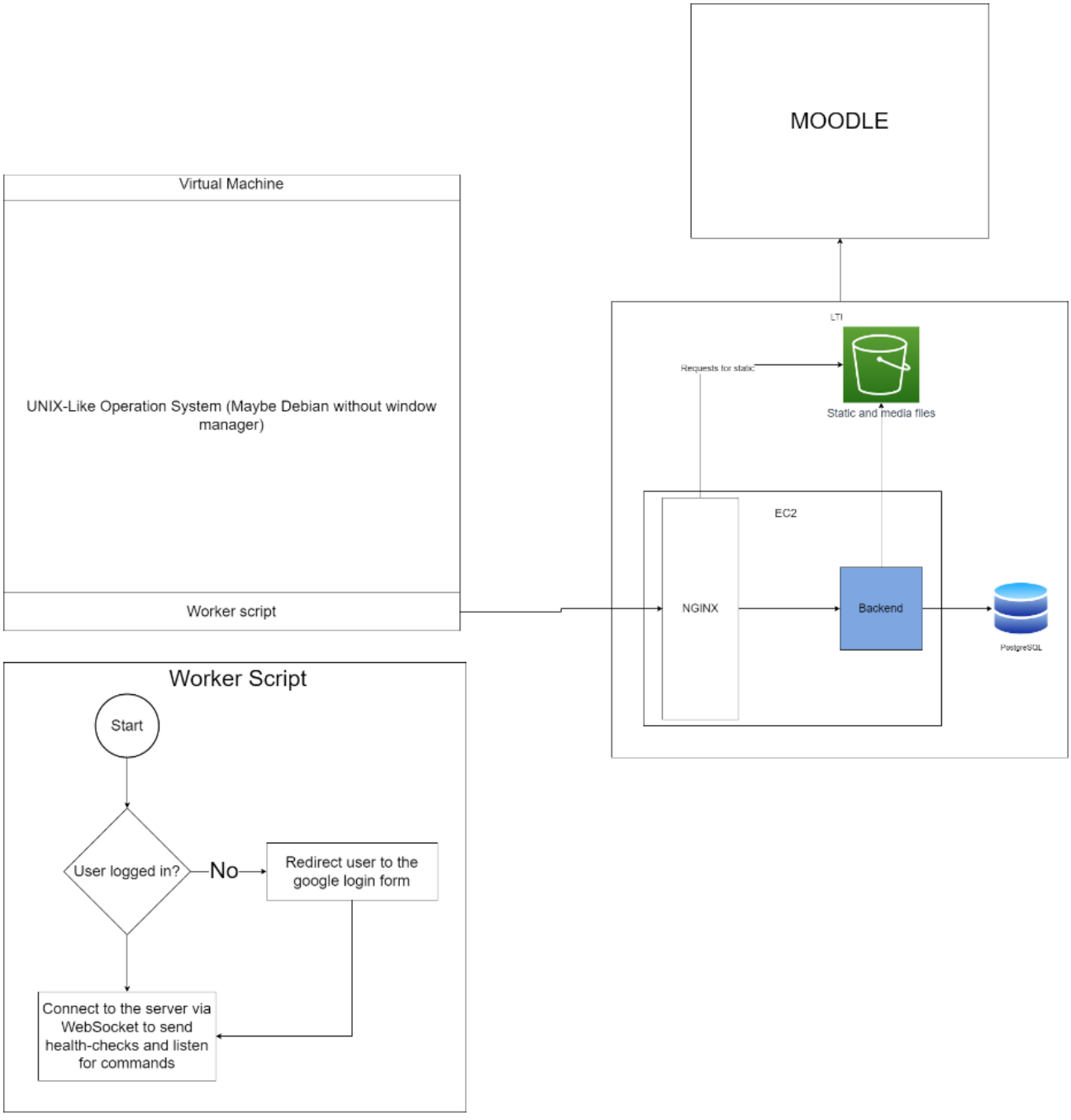

Рисунок 2.2 – Обрана архітектура проєкту

2.2 Канал зв’язку між клієнтом та сервером

Основним каналом зв’язку між клієнтом та сервером був обраний протокол WebSocket виконаний на базі протоколу TCP. З переваг WebSocket можна виділити простоту роботи з ним, велику популярність та наявність незліченної кількості бібліотек для роботи з ним для різних мов програмування. Також одним з головних критеріїв для вибору протоколу було можливість підтримки зв'язку між клієнтом та сервером через надійний швидкий канал що може використовуватися у обидві сторони. Також варто зазначити що саме клієнтська машина повинна здійснювати підключення до сервера з публічною адресою і не навпаки через неможливість визначити адресу клієнтської системи та побудувати до неї маршрут із зовні.

На ранніх етапах проєктування також розглядалась можливість використання звичайного REST API на базі HTTP (рис. 2.4) але у нашому випадку використання протоколу WebSocket дає нам змогу відкрити канал та відправляти системні сповіщення у форматі JSON по відкритому каналу та дозволяє виконати авторизацію клієнта одноразово під час відкриття з’єднання (на відміну від REST). Порівняння протоколу WebSocket та HTTP наведено на рис. 2.3.

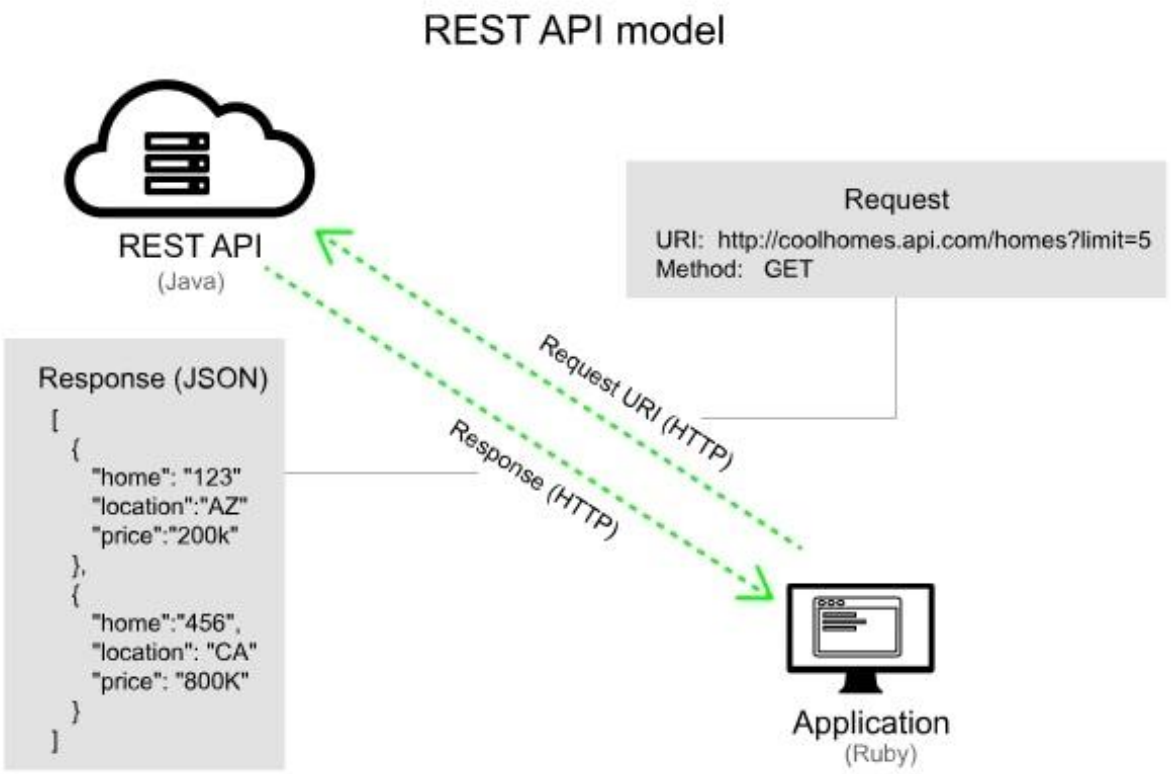

Рисунок 2.3 – Приклад запиту за допомогою REST API

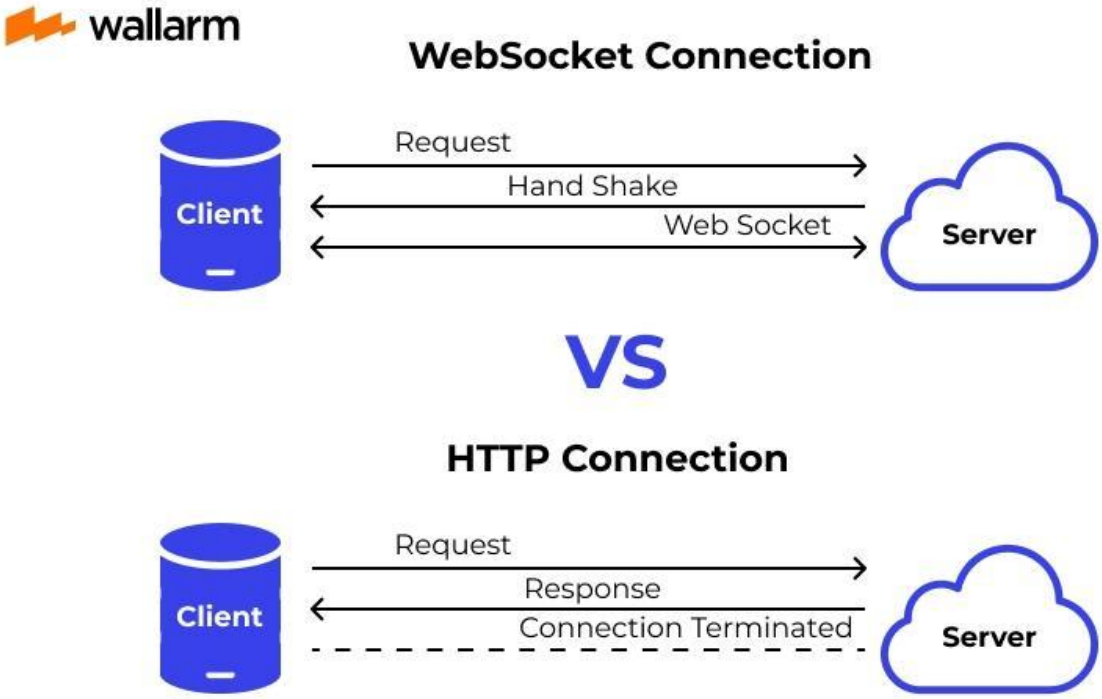

Рисунок 2.4 – Порівняння з’єднань через WebSocket та HTTP

2.3 Будова серверної частини системи

Для сервера було обрано систему Debian Linux, стабільну та надійну. На сервері запущено програму, що обробляє усі запити від користувачів. У ролі користувачів в нас виступають веб-браузер учня та його віртуальна машина (worker). Клієнти спілкуються з сервером використовуючи вебсокети (WebSocket), що дуже зручно для підтримки відкритого двостороннього з’єднання. Для взаємодії клієнтів та сервера потрібно розробити прикладний програмний інтерфейс (API) – набір чітко визначених методів взаємодії між сервісом та програмою, що його використовує.

Наш сервер створює у системі активний порт для вхідних підключень за допомогою HTTP, що припустимо при ведені розробки у локальному середовищі, але небезпечно при взаємодії з користувачем через глобальну мережу, бо може привести до витоку даних. Тобто потрібно захистити з’єднання за допомогою HTTPS. Для цього було налаштовано окремий публічний сервер з Nginx у ролі зворотнього проксі сервера. Це дозволило нам швидко налаштувати HTTPS для нашого сервера, а також перенести сервер у захищену мережу VPN. Також, у майбутньому це дасть нам змогу збільшити надійність системи, за рахунок налаштування лімітів для відкритих з’єднань з однієї IP адреси та налаштувати LoadBalancer для розподілення навантаження.

2.4 Інструмент для перевірки завдань

Для виконання головної мети – автоматизації перевірки домашніх завдань, потрібен якийсь інструмент, котрий буде вміти це робити, і тут є два підходи, для вирішення цієї задачі.

Перший підхід – це розширення функціоналу самої LMS, але щоб це зробити доведеться залізти в величезну кодову базу, при цьому нічого не зламавши.

Другий підхід займе більше часу для реалізації та в деяких моментах може видатися складнішим, але при цьому він більш доцільний, з точки зору реалізації. Підхід полягає у створенні окремого додатку котрий буде перевіряти завдання, підтримувати зв'язок з машиною учня, та ініціювати перевірки виконання завдань. Цей підхід є більш складним, тому що, треба додатково дослідити роботу такого стандарту як LTI, котрий був розроблений саме для об'єднання різних елементів систем навчання.

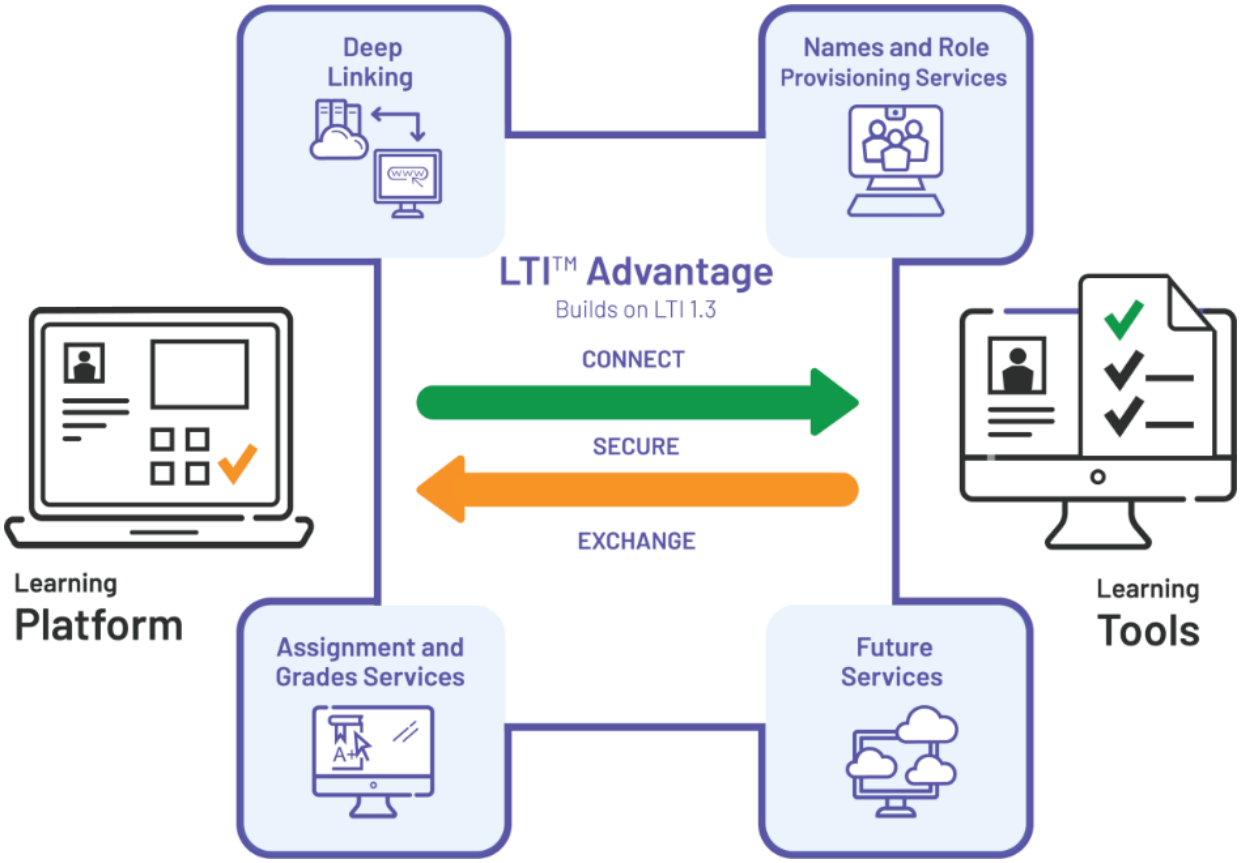

Рисунок 2.5 – Візуалізація стандарту LTI

LTI (Learning Tools Interoperability) - це стандарт, який дозволяє інтегрувати зовнішні навчальні засоби та додатки (наприклад, відеоуроки, тестові системи або інші інтерактивні ресурси) в систему керування навчанням безперешкодно та безпечно.

Переваги:

– LTI спрощує процес інтеграції зовнішніх навчальних ресурсів в LMS, що дозволяє вчителям та студентам швидко отримувати доступ до цих ресурсів;

– Стандартизує взаємодію між різним LMS і зовнішніми застосунками;

– Забезпечує механізми авторизації, тим самим дозволяє захистити зовнішні ресурси і забезпечити конфіденційність даних.

Недоліки:

– Обмежує функціонал зовнішніх додатків до стандартного набору можливостей, що може бути недостатнім для деяких освітніх завдань.

– Інтерфейси LTI можуть відрізнятися між різними LMS, що може призвести до несумісності деяких додатків.

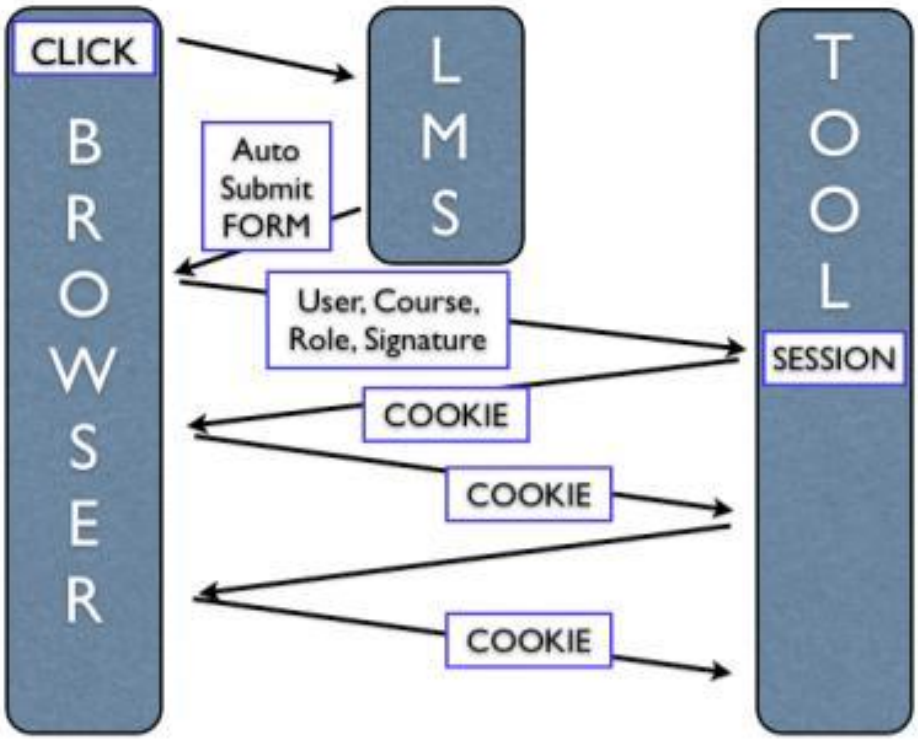

Рисунок 2.6 – Приклад початку сесії клієнтом у системі побудованій з використанням стандарту LTI

Для написання серверної сторони додатку, котрий автоматизує перевірку виконання завдань учнем, було обрано мову програмування Python.

2.5 Локальне середовище учня

До вибору операційної системи для клієнтської машини також підійшли з відповідальністю та провели порівняння переваг та недоліків основних дистрибутивів Linux на даний момент. Найкращим варіантом став Debian 11, саме його було обрано у якості основної операційної системи для середовища учня.

З переваг Debian 11 можна виділити:

– Стабільні релізи, що зменшує ймовірність випадково зламати систему (у порівняння наприклад з Arch);

– Висока популярність та розвинута спільнота;

– Стандартизована взаємодія з системою, бо саме ядро Debian лежить у основі найбільш популярних дистрибутивів, таких як Ubuntu або Kali;

– Відносна простота у налаштуванні безпеки та конфіденційності даних.

Після обрання операційної системи постало питання автоматизованої генерації образа віртуальної машини для значної економії часу на тестування та створення нових версій середовища (для зберігання у системі контролю версій). У якості інструменту для виконання роботи з генерації образів було обрано Packer від компанії HashiCorp. Він дозволяє описати бажану конфігурацію у вигляді коду та генерувати образи віртуальних машин одразу для декількох найпопулярніших гіпервізорів (VirtualBox, VMware ESXi/Workstation, Proxmox VE).

Наступним етапом було вирішення проблеми з доступом студента для ідентифікації його у системі контролю навчання. На щастя операційні системи на базі Linux мають інтегрований компонент для авторизації користувачів – PAM (Privileged Access Management). Використовуючи PAM, розробник майже необмежений у рішенні авторизації користувачів у системі і має можливість реалізувати свою логіку авторизації за потреби. У нашому випадку для авторизації студента за допомогою OAuth нам потрібно пов'язати клієнтську машину з OIDC провайдером, і для цього є два варіанти:

1. Реалізувати OIDC PAM модуль, що буде використовувати OAuth Device Flow або grant_type=password;
2. Використати додатковий IAM (Identity and Access Management) сервіс з підтримкою LDAP, який буде слугувати прошарком між OIDC провайдером та клієнтською системою.

Перший варіант потребує розробки власного PAM модуля а також має більш складну взаємодію з користувачем та потребує реалізації логіки оновлення токенів закладених у основу технології OpenID, що може погіршити взаємодію користувача з системою та привести до непередбачуваних проблем з синхронізацією прав доступу. Другий варіант потребує налаштування додаткових сервісів IAM та LDAP, але з іншого боку дозволяє нам використати вбудовану бібліотеку pam_ldap.so і не перейматися за синхронізацію даних з OIDC провайдером. Було вирішено обрати перший варіант, бо налаштовувати IAM та LDAP для використання однієї функції здалося недоцільним.

3 РОЗРОБКА СИСТЕМИ АВТОМАТИЗОВАНОЇ ПЕРЕВІРКИ ЗАВДАНЬ

3.1 Модуль автентифікації користувача

Для визначення приналежності віртуальної машини певному студенту потрібен спосіб безпечно верефікувати користувача в системі. Як результат було розроблено простий модуль PAM, що працює з системоюOpenID OIDC. Конфігурація PAM для віртуальної машини учня наведена на рис. 3.1.

```
root@debian:[~]: cat /etc/pam.d/common-auth
auth    [success=1 default=ignore]      pam_succeed_if.so user in student
auth    [success=2 default=ignore]      pam_succeed_if.so user in root:admin
auth    [success=1 default=ignore]      pam_oidc.so client-id=
      .apps.googleusercontent.com client-secret=
auth    [default=die]                   pam_deny.so
auth    optional                        pam_unix.so nullok
```

Рисунок 3.1 – Конфігурація PAM з підтримкою автентифікації за допомогою OIDC

Дана конфігурація поєднує як звичайну автентифікацію, за допомогою pam_unix.so, так і автентифікацію із використанням pam_oidc.so. Це дозволяє використовувати обліковий запис “student” для доступу до робочого середовища учня та обліковий запис “root” для обслуговування віртуальних машин адміністратором.

3.2 Клієнтський рушій

Клієнтський рушій (рис.3.2) виконано на базі мови програмування Golang, що дозволяє нам скомпілювати код та запакувати усі залежності у один бінарний файл. Такий підхід збільшує надійність з боку безпеки в порівнянні наприклад зі скриптовими мовами програмування та у порівнянні з ними ж дозволяє збільшити швидкість роботи програми. Додатково таку програму набагато простіше налаштувати під середовище, де її планується запускати. Також це підвищує надійність та безпеку системи за рахунок відсутності потреби додавати сторонні пакети програм, такі як інтерпретатори коду, що у свою чергу зменшує кількість вразливих точок системи.

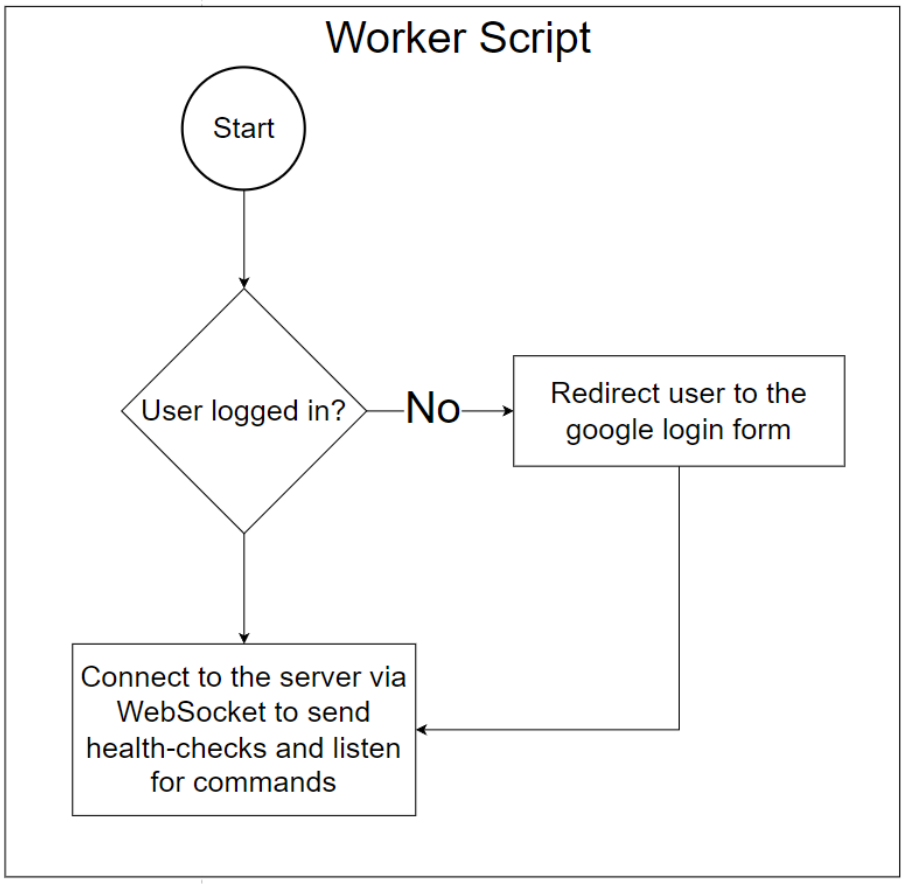

Рисунок 3.2 – Абстрактна діаграми логіки роботи клієнтського рушія

3.3 Збірка образу віртуальної машини

Пакування монолітної програми за допомого Packer не видається складним рішенням та може бути реалізовано завдяки спеціальній інструкції для копіювання локальних файлів у середину згенерованого образу або за допомогою копіювання файлу через протокол SSH (для цього підходять інструменти SCP та SFTP). Packer підтримує постпроцесори для виконання потрібних нам задач після збірки образу.

```
"provisioners": [
  {
    "type": "shell",
    "execute_command": "echo 'packer'|{{.Vars}} sudo -S -E bash '{{.Path}}'",
    "inline": [
      "apt -y update && apt -y upgrade",
      "apt -y install python3-pip",
      "pip3 --no-cache-dir install ansible",
      "mkdir -p /opt/lms && chmod 777 /opt/lms"
    ]
  },
  {
    "type": "file",
    "source": "files/worker",
    "destination": "/opt/lms/worker"
  },
  {
    "type": "file",
    "source": "files/service.sh",
    "destination": "/opt/lms/service.sh"
  },
  {
    "type": "file",
    "source": "files/pam_oidc.so",
    "destination": "/tmp/"
  },
  {
    "type": "ansible-local",
    "playbook_file": "files/setup.yml"
  },
  {
    "type": "shell",
    "execute_command": "echo 'packer'|{{.Vars}} sudo -S -E bash '{{.Path}}'",
    "scripts": ["files/cleanup.sh"]
  }
]
```

Рисунок 3.3 – Блок “provisioners” конфігурації Packer

На рисунку 3.3 зображено блок “provisioners” конфігурації Packer, де встановлено кроки та порядок налаштування віртуальної машини. Завдяки цьому після встановлення операційної системи Packer виконує наступне:

1. Оновлює базу даних у якій зазначено доступні для встановлення пакети та встановлює необхідні нам пакети.
2. Копіює до диску віртуальної машини клієнтський рушій
3. Копіює сервіс написаний на мові Bash, який створено для контролю стану клієнтського рушія та його автоматичного перезапуску при виникненні збоїв.

3.4 Робота з віртуальним середовищем

Коли студент отримує завдання вперше, він починає роботу із "чистої" віртуальної машини, тобто такої, до якої не було внесено змін, після того, як учень отримав архів із віртуальною машиною від викладача. Така машина не містить інформації, щодо належності тому чи іншому учню, та потребує первинного налаштування. Для початку потрібно відкрити сторінку із завданням (рисунок 3.4).

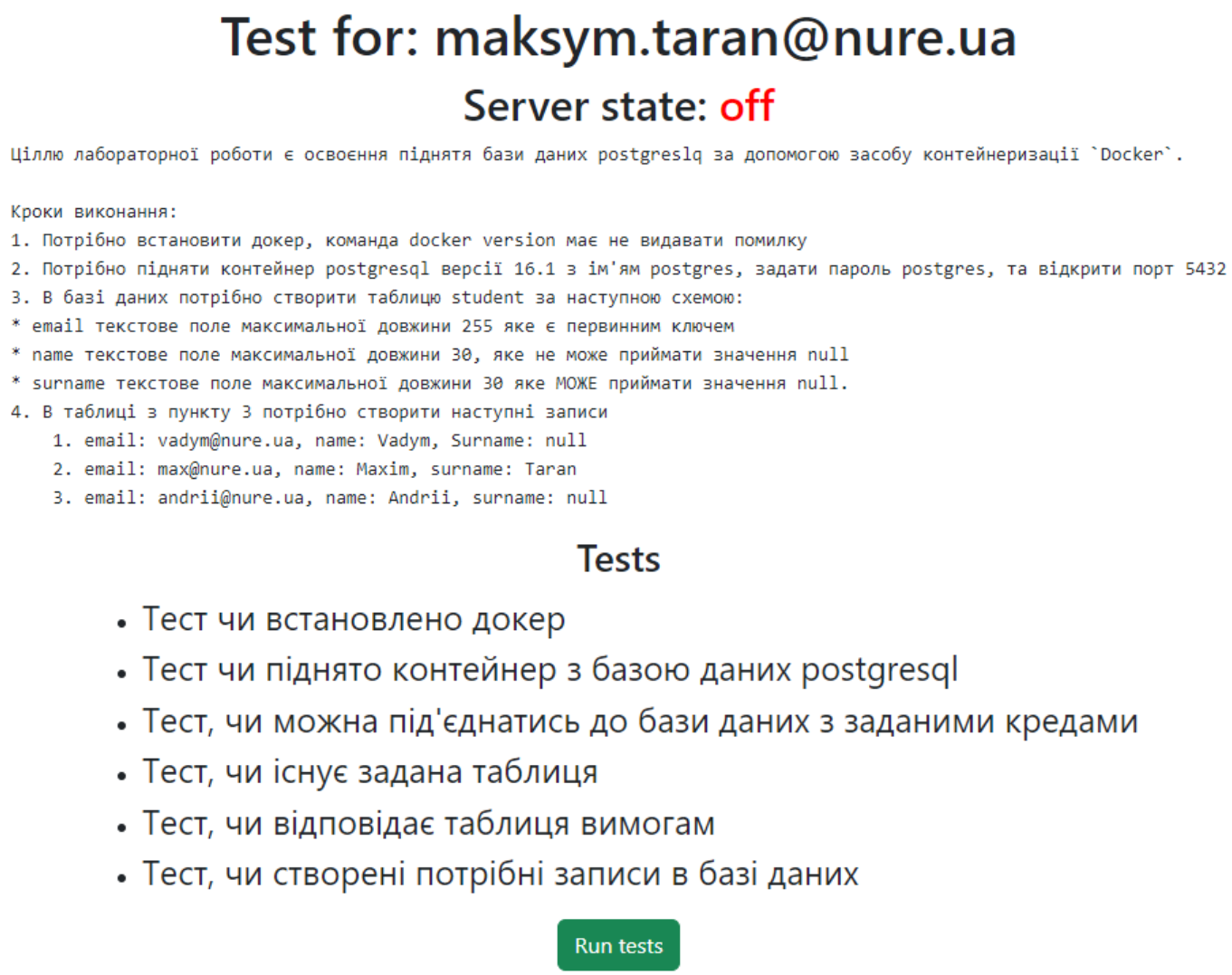

Рисунок 3.4 – Сторінка із завданням

Тут можна побачити вказівки до виконання роботи, статус робочого середовища учня (server state) та звідси ж можна запустити перевірку виконання завдань.

Далі студент повинен запустити своє робоче середовище (Віртуальну машину видану викладачем). Після запуску системи та появи діалогу про вхід у систему учень повинен ввести логін "student" (надалі цей логін буде використовуватися учнем для входу та роботи з системою). Після цього система покаже сповіщення (рисунок 3.5), викликане нашим модулем pam_oidc.so. Щоб увійти до системи потрібно прослідувати інструкції на екрані, а саме, перейти за посиланням google.com/devices у браузері на іншому

пристрої, та авторизуватися за допомогою акаунту Google, тобто пошти домену nure.ua.

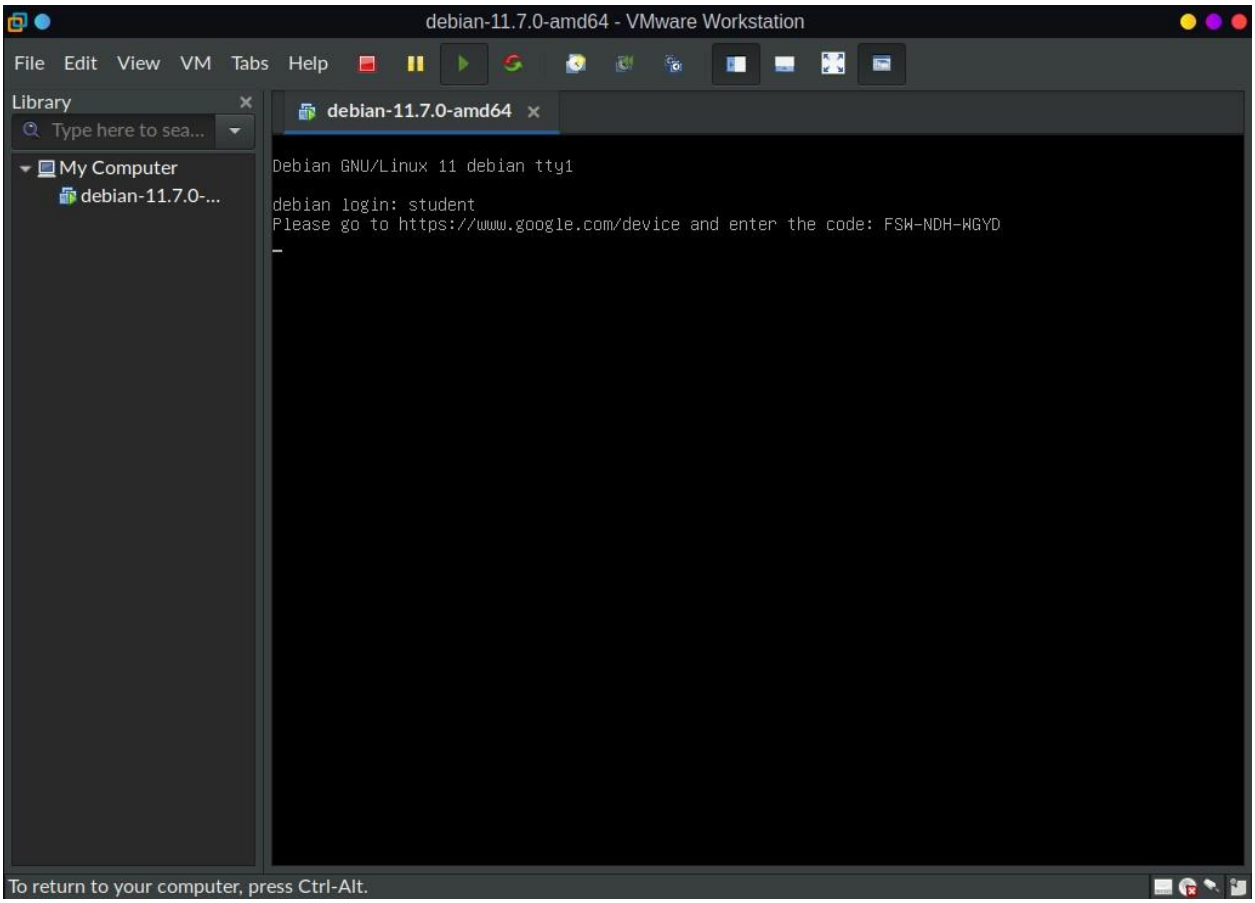

Рисунок 3.5 – Процес авторизації у середовищі учня

Після успішної авторизації у системі студент має змогу працювати у своєму віртуальному оточенні, виконуючи завдання видані викладачем на сайті dl.nure.ua.

Завдяки використанню постпроцесорів система повністю готова до експлуатації одразу після виконання команди packer build. Вигляд віртуального робочого середовища учня наведено на рис. 3.6.

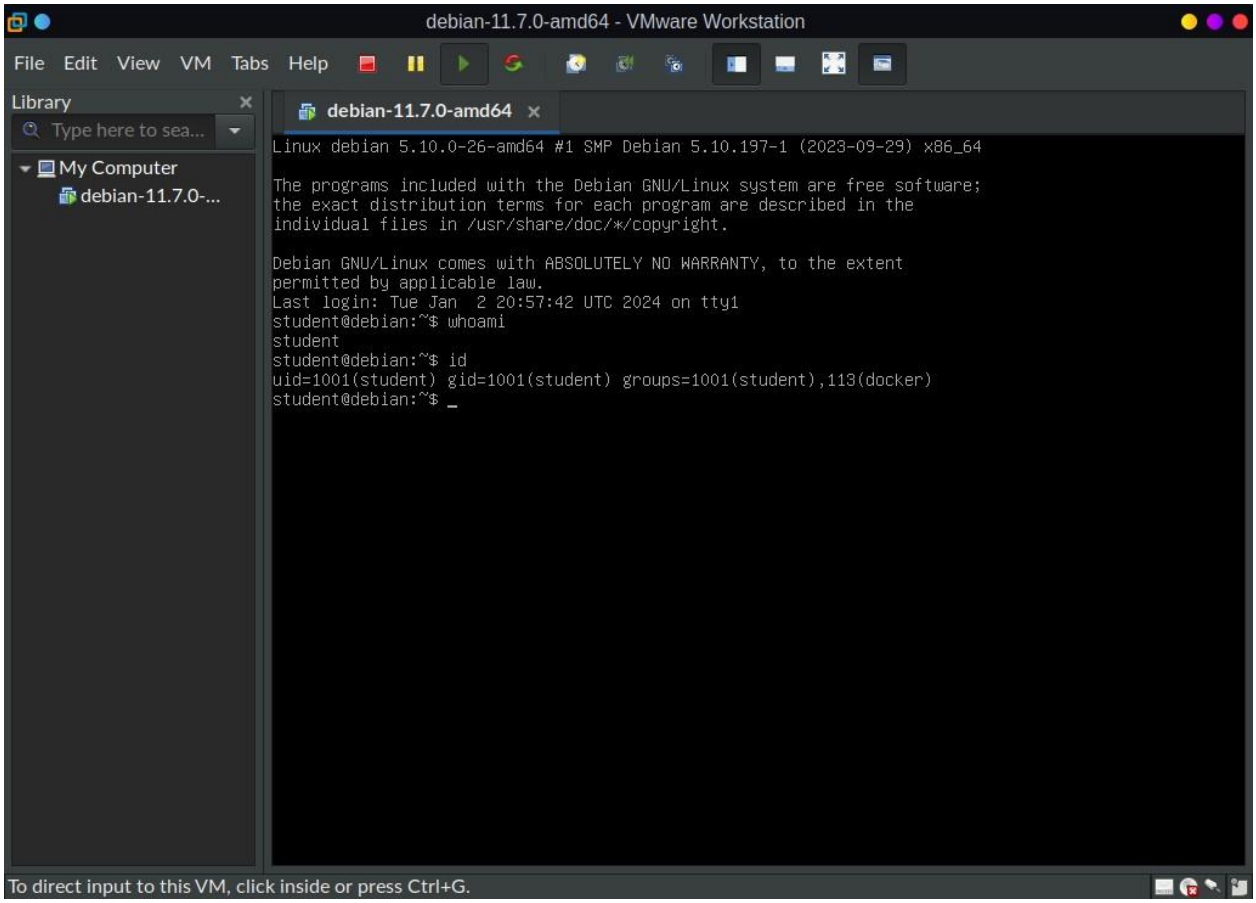

Рисунок 3.6 – Авторизована робота з середовищем учня

На цьому етапі учень повинен побачити, що статус його машини (server state) змінився з “off” на “on”. Переконатися в цьому можна на сторінці із поставленим заданням, як показано на рисунку 3.7.

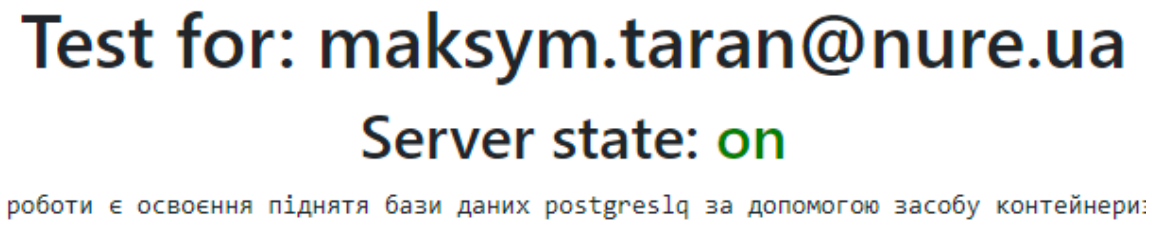

Рисунок 3.7 – Зміна статусу при підключеній віртуальній машині

Після виконання завдань учень повинен запустити тести на сторінці із завданням, натисканням кнопки "Run tests". Усі результати перевірки відобразяться на цій сторінці (рисунок 3.8). Якщо тест не завершився успіхом, то студент може проаналізувати опис помилки та спробувати виправити проблему власноруч.

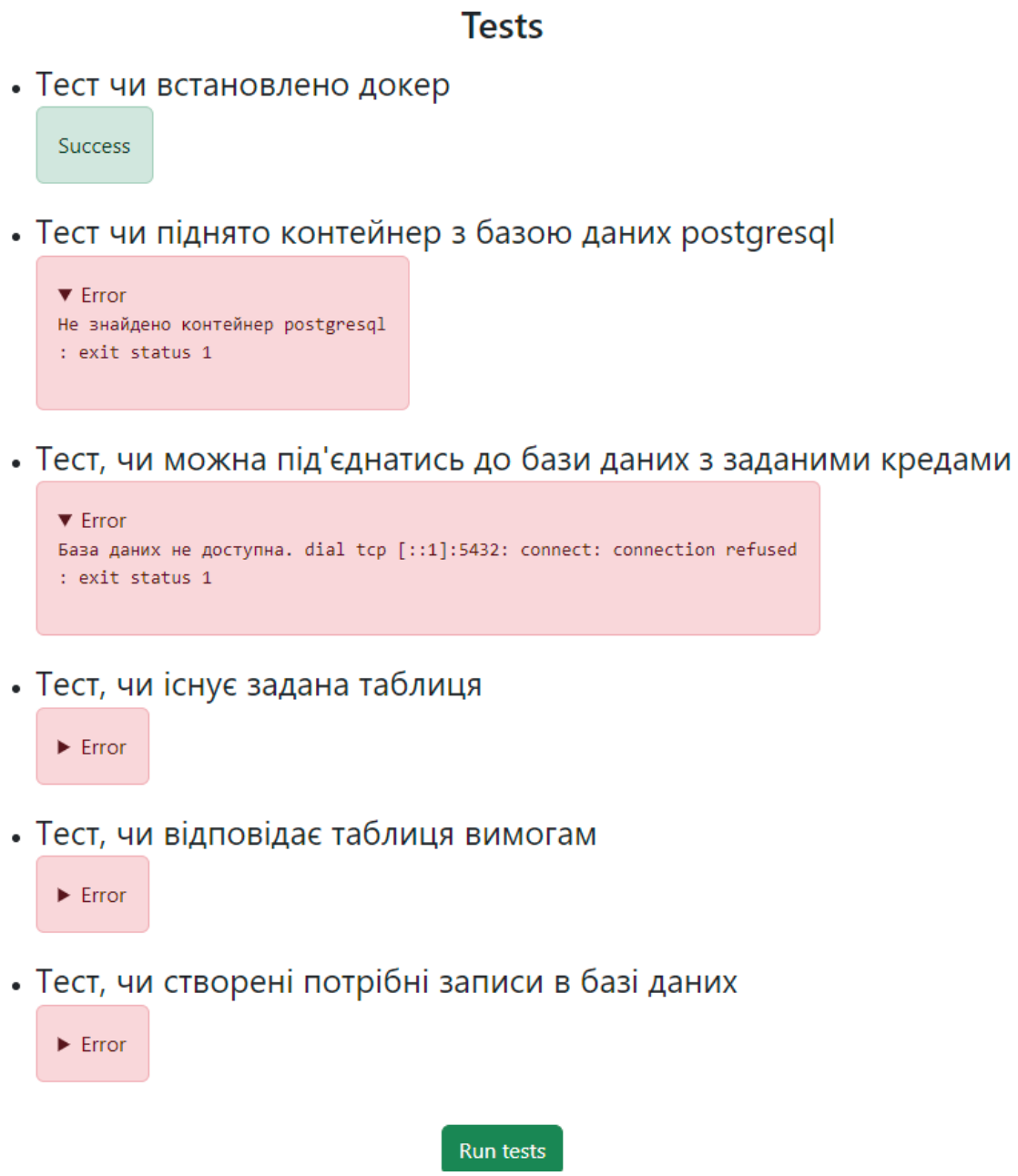

Рисунок 3.8 – Результат перевірки виконання завдань

Коли користувач запускає перевірку завдання система автоматично виставляє оцінку зважаючи на кількість правильно та не правильно виконаних задач завдання, тобто передбачається що усі проміжні задачі оцінюються однаково.

3.5 Можливості вдосконалення

Створена нами система має великий потенціал щодо вдосконалення. Наприклад безпека може бути вдосконалена додаванням модуля, який буде перевіряти UUID системи, який можна зчитати з фай лу /sys/class/dmi/id/product_uuid. Також має сенс додати можливість змінювати оцінку яку учень отримує за ту чи іншу задачу завдання, або ввести поняття коефіцієнта вагомості задачі.

Обрана архітектура дозволяє абстрагуватися від таких понять, як устаткування або операційна система система, що в свою чергу дозволяє гнучко заміняти елементи системи без великих зусиль та витрати часу. Наприклад, за потребою можна перенести частину або навіть усі віртуальні середовища учнів у хмару, а термінал, що дозволяє працювати з середовищем інтегрувати у сторінку із завданням. Для цього можна використовувати як AWS EC2, так і кластер Kubernetes, що дозволить дуже швидко створювати нові середовища для роботи.

ВИСНОВКИ

Дослідження існуючих на ринку рішень показало, що наявність подібних систем до описаної у поставленої задачі але також і можливість до вдосконалення та створення унікальної LMS, що може використовуватися для оцінювання якості виконання роботи у локальному середовищі студента.

Вирішення поставленої задачі це комплексне рішення з проєктування оптимальної архітектури та API для комунікації між окремими модулями усієї системи. Продуманий підхід до проєктування, налаштування та автоматизації дозволяє зменшити час потрібний на розробку системи допомагає зменшити кількість помилок та підвищує надійність застосунку.

Система, що сприяє підвищенню ефективності та доступності освіти, спрощує оцінювання та аналіз результатів, а також допомагає викладачам більш ефективно взаємодіяти зі студентами це рішення що знайде своє місце у сучасному світі що має активну тенденцію до підвищення якості освіти та адаптивність до швидкоплинних змін у інформації та технологіях.

Всі пункти технічного завдання виконано в повному обсязі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

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

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

3. Специфікація LTI: https://www.imsglobal.org/spec/lti/v1p3/impl/

4. Що таке Packer: https://developer.hashicorp.com/packer/docs/intro

5. Порівняння монолітної та мікросервісної архітектур: https://aws.amazon.com/compare/the-difference-between-monolithic-and-microservices-architecture/

6. Що таке WebSocket: https://www.wallarm.com/what/a-simple-explanation-of-what-a-websocket-is

7. Таран М. В. Дослідження шляхів автоматизації перевірки виконання практичних завдань в локальному віртуальному оточенні користувача / М. В. Таран, В. В. Тіщенко, А. І. Костромицький. // Міжнародна науково-технічна конференція «Інформаційно-комунікаційні технології та кібербезпека (ІКТК-2023)» Харків, 7 – 8 грудня 2023.