Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

WEB-ДОДАТОК ДЛЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ КОНТРОЛЮ ЯКОСТІ ВИРОБНИЦТВА

Виконав: студент 2 курсу, групи 2КІ-22м
спеціальності 123 — «Комп'ютерна інженерія»
_____ Салата О. Л.

Керівник: к.пед.н., доц.каф. ОТ
_____ Добровольська Н. В.
«11» 12 2023 р.

Опонент: д.т.н., проф. каф. ПЗ
_____ Ліщинська Л. Б.
«12» 12 2023 р.

Допущено до захисту

Завідувач кафедри ОТ
_____ д.т.н., проф. Азаров О. Д.
«15» 12 2023 р

ВНТУ 2023

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Галузь знань — Інформаційні технології
Освітній рівень — магістр
Спеціальність — 123 Комп'ютерна інженерія
Освітньо-професійна програма — Комп'ютерна інженерія

ЗАТВЕРДЖУЮ

Завідувач кафедри обчислювальної техніки

О.Д. Азаров

"26" вересня 2023 р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Салаті Олександру Леонідовичу

1 Тема роботи «Web-додаток для автоматизованої системи контролю якості виробництва» керівник роботи Добровольська Наталія Вікторівна к.пед.н., доцент, затверджено наказом вищого навчального закладу від **18.09.23** року **№ 247.**

2 Строк подання студентом роботи **14.12.23**.

3 Вихідні дані до роботи: призначення : інформування користувача про якість продукції на виробництві, засоби — середовище програмування Visual Studio Code , мови програмування JavaScript, Node.js.

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): вступ, аналіз сучасних методів та технологій контролю якості на виробництві, Аналіз та вибір інструментів для розробки веб-додатку, розробка та тестування веб-додатку.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): блок-схема алгоритму, ER-діаграма бази даних, структурна схема.

6 Консультанти розділів роботи приведені в таблиці 1.

Таблиця 1— Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Добровольська Наталія Вікторівна к.пед.н., доцент		
5	Небава Микола Іванович проф., к.е.н		

7 Дата видачі завдання **19.09.2023** .

8 Календарний план виконання МКР приведений в таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів МКР	Строк виконання	Підпис
1	Постановка задачі	**19.09.23**	
2	Огляд існуючих рішень	**21.09.23**	
3	Розробка структурної схеми	**27.09.23**	
5	Розрахунок аналогової частини	**10.10.23**	
6	Вибір ПЗ для розробки	**18.10.23**	
7	Розробка роботи веб-додатку	**25.10.23**	
8	Розрахунок економічної частини	**5.11.23**	
9	Оформлення пояснювальної записки та ілюстративного матеріалу	**18.11.23**	
10	Виконання магістерської кваліфікаційної роботи	**25.11.23**	
11	Перевірка якості виконання магістерської кваліфікаційної роботи та усунення недоліків	**4.12.23**	
12	Підписи супроводжувальних документів у керівника, опонента, нормоконтролера	**8.12.23**	
13	Перевірка «антиплагіат»	**11.12.23**	
14	Попередній захист	**15.12.23**	

Студент Салата Олександр Леонідович

Керівник к.пед.н., доц. Добровольська Наталія Вікторівна

АНОТАЦІЯ

УДК 004

Салата О. Л. Web-додаток для автоматизованої системи контролю якості виробництва. Магістерська кваліфікаційна робота зі спеціальності 123 — Комп'ютерна Інженерія, Вінниця: ВНТУ, 2023 — 120 с. На укр. мові. Бібліогр.: 25 назв; рис.: 28; табл. 10.

У роботі розглянуто сучасні методи та технології автоматизації контролю якості на виробництві, вибрано та обгрунтуванно програмне забезпечення, розроблено клієнтську та серверну частину веб-додатку, написані рекомендації для користувача.

Ключові слова: контроль, якість, веб-додаток, виробництво.

ABSTRACT

УДК 004

Salata O. L. Web application for an automated quality control system in manufacturing. Master's qualification work in the specialty 123 — Computer Engineering, Vinnytsia: VNTU, 2023 — 120 p. In Ukrainian. Bibliography: 25 titles; figures: 28; tables: 10.

The paper explores modern methods and technologies for automating quality control in manufacturing, selects and justifies the software, develops the client and server parts of the web application, and provides user recommendations.

Keywords: control, quality, web application, manufacturing.

ЗМІСТ

ВСТУП

Контроль якості на виробництві — це систематичний і комплексний процес, спрямований на забезпечення високого рівня якості продукції чи послуг, які випускає компанія. Цей процес охоплює весь життєвий цикл виробництва, від початкового етапу до завершення, і включає в себе контроль якості матеріалів, технологічних процесів та готової продукції.

Метою контролю якості є виявлення та усунення будь-яких невідповідностей стандартам якості, специфікаціям чи вимогам клієнтів. Це може включати в себе випробування продукції, вимірювання параметрів якості, визначення відповідності технічним стандартам, а також системи аудиту та внутрішнього контролю.

Успішний контроль якості допомагає підтримувати репутацію компанії, забезпечує задоволення клієнтів та сприяє уникненню негативних наслідків, пов'язаних з дефектами чи неякісною продукцією. Він також може підвищувати конкурентоспроможність компанії на ринку, створюючи підстави для довіри споживачів та партнерів.

Розробка веб-додатку, який автоматизує процес контролю якості, стає невід'ємною частиною сучасного виробництва.

Актуальність теми дослідження полягає в тому, що розроблювальний додаток забезпечує швидкий та ефективний моніторинг усіх етапів виробничого процесу та дозволяє збільшити ефективність, знизити витрати та мінімізувати ризик виробничих невдач. Такий веб-додаток є актуальним не лише для підприємств, але й для клієнтів, які отримають гарантію якості та можливість впливати на виробництво за допомогою ефективного моніторингу.

Метою роботи є розробка веб-додатку для автоматизованої системи контролю якості виробництва, з можливістю покращення інформаційне обслуговування працівників, завдяки IoT-технологіям.

Для досягнення поставленої мети у роботі розв’язуються такі **задачі**:

— аналіз сучасного стану розвитку веб-додатку для контролю за якістю на виробництві;

— вибір інструментів для розробки веб-додатку;

— розробка архітектури веб-додатку;

— розробка клієнтської та серверної частини веб-додатку.

Об'єктом дослідження є процес взаємодії веб-додатку із апаратним забезпеченням для контролю якості на виробництві.

Предметом дослідження є програмні засоби для перегляду матеріалів з використанням технологій розробки веб-додатків для використання у браузері.

Новизна роботи полягає в тому, що набула подальшого розвитку модель, яка, на відміну від існуючих враховує особливості сучасних сенсорів та IoT-з'єднань, що дозволяє переглядати інформацію у режимі онлайн, забезпечуючи користувачам миттєвий та актуальний доступ до даних, вимірювання якості в реальному часі та оперативне реагування на будь-які аномалії чи зміни у виробничому процесі.

Практичне значення роботи полягає у тому, що на основі отриманих у магістерській кваліфікаційній роботі теоретичних положень, запропоновано програмний засіб, який втілює в собі розроблені концепції та методології, надаючи ефективні інструменти для реалізації передових веб-технологій та автоматизації системи контролю якості виробництва.

Апробація результатів роботи здійснена у доповіді на LI Науково-технічній конференції підрозділів Вінницького національного технічного університету (2023) [1].

1 АНАЛІЗ СУЧАСНИХ МЕТОДІВ ТА ТЕХНОЛОГІЙ СИСТЕМИ КОНТРОЛЮ ЯКОСТІ ВИРОБНИЦТВА

1.1 Огляд сучасних інформаційних технологій для автоматизації контролю якості.

Сучасний контроль якості стало неможливо уявити без інформаційних технологій, які значно полегшують процес моніторингу та забезпечення високої якості продукції. Існує безліч інноваційних підходів і технологій, які революціонізують контроль якості в різних галузях [1].

Інтернет речей (IoT) — це концепція, яка визначає мережу підключених до Інтернету фізичних об'єктів, пристроїв і систем, які можуть обмінюватися даними та інформацією між собою та з центральними обчислювальними системами через Інтернет. Основна ідея за IoT полягає в тому, щоб робити об'єкти навколишнього світу "розумними", дозволяючи їм сприймати оточуюче середовище, збирати дані і навіть взаємодіяти з іншими об'єктами та системами (рисунок 1.1).

Рисунок 1.1 — Інтернет речей (IoT)

Основні компоненти IoT включають в себе:

— сенсори, які можуть вимірювати різні параметри, такі як температура, вологість, рух, освітленість, звук і багато інших., процесори і мікроконтролери дозволяють обробляти ці дані та комунікувати з іншими пристроями;

— дані, які зібрані сенсорами, передаються через мережу до центральних серверів або хмарних обчислювальних ресурсів;

— дані, які надходять від iot-пристроїв, зберігаються та аналізуються для виділення корисної інформації та виявлення залежностей та трендів;

— системи iot можуть надавати можливість віддаленого управління пристроями та системами через інтернет, а також надсилати сповіщення та реагувати на події.

Інтернет речей (IoT) став переліком пристроїв та сенсорів, які з'єднані в одну мережу. Вони надсилають дані про стан обладнання, виробничі параметри, а також інші важливі метрики в режимі реального часу. Це дозволяє здійснювати нагляд і керування виробничими процесами, реагуючи на будь-які аномалії або відхилення від норми [2].

Аналітика даних стала важливим інструментом для автоматичного виявлення аномалій і трендів в процесах виробництва. За допомогою різних алгоритмів і моделей машинного навчання можна передбачити можливі проблеми з якістю виробів або виробничими процесами.

Машинне навчання і штучний інтелект розвинулись як інструменти для автоматичного аналізу та розпізнавання дефектів на виробах, і їхні можливості стають надзвичайно важливими в галузі контролю якості. Вони можуть визначати навіть найдрібніші недоліки та аномалії, які можуть залишитися непоміченими за допомогою традиційних методів інспекції [2].

За допомогою навчання машин та штучного інтелекту, системи можуть аналізувати великі обсяги візуальних та сенсорних даних з виробничого процесу. Наприклад, камери і сенсори можуть зафіксувати мільйони пікселів зображення в секунду, а потужні алгоритми обробки даних можуть оцінювати кожен з них на предмет можливих дефектів.

Ця точність у виявленні дефектів не тільки полегшує роботу інспекторів та контролерів, але й дозволяє підприємствам виявляти і усувати проблеми на ранніх стадіях виробництва, що значно зменшує витрати та відходи. Крім того, ця технологія сприяє постійному вдосконаленню виробничих процесів і якості

продукції завдяки аналізу великої кількості даних та ідентифікації чинників, що впливають на якість [2].

Комп'ютерне зорове спостереження використовує камери і системи обробки зображень для виявлення дефектів і аномалій. Це особливо корисно в автоматизованих системах сортування виробів за якістю [2].

QR-коди і RFID-мітки надають можливість ідентифікувати та відстежувати вироби в реальному часі, збираючи дані про їхню історію та якість на різних етапах виробництва [2].

Цифрові двійники, або цифрові моделі фізичних об'єктів, дозволяють створити віртуальну копію продукту або процесу для моніторингу і аналізу в реальному часі (рисунок 1.2).

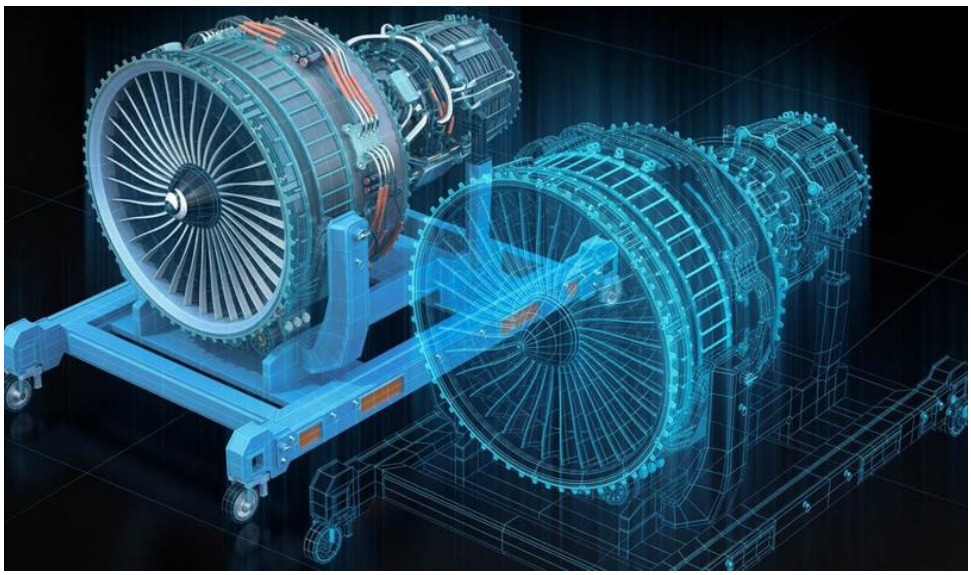

Рисунок 1.2 — Цифрові двійники

Впровадження машинного зору суттево спростить процес контролю якості, та дозволить поліпшити ключові показники виробництва.

Візуальна інспекція є невід’ємною складовою контролю якості виробництва. Впровадження автоматизованих систем візуальної інспекції забезпечує високий рівень контролю якості виготовленої продукції. Такі системи можуть використовуватись як частина контролю якості виробничої або пакувального лінії та проводити комплексну перевірку якості виготовленої продукції на кожному з етапів виробництва (рисунок 1.3) [2].

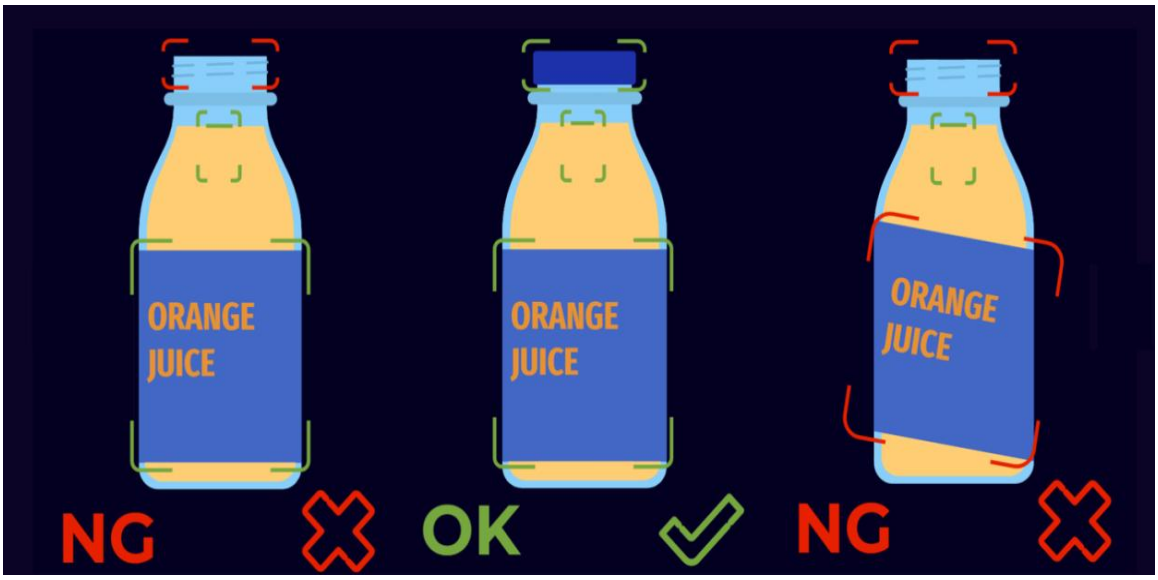

Рисунок 1.3 — Приклад візуальної інспекції

Системи забезпечують повну автоматизацію процесу візуальної інспекції виготовленої продукції та проводити наступні перевірки: наявність / відсутність елементів, перевірка розмірів та форми елементів, перевірка кількості та позиції елементів [3].

Системи машинного зору також використовуються для перевірки комплектності вузлів, агрегатів і компонентів та їх відповідності специфікаціям замовника, а також для автоматичного виявлення дефектів до того, як продукція зійде з виробничої лінії, що дозволяє підвищити продуктивність і знизити кількість бракованої продукції [3].

На рисунку 1.4 наведена система машинного зору, яка була розроблена для перевірки якості пакування шоколадної плитки. Інтуїтивно зрозумілий інтерфейс користувача дозволяє легко провести налаштування, а також забезпечує доступ до статистичних даних та трендів виробництва. Веде архівування даних та зображень. Автоматизована система візуальної інспекції контролює наступні параметри:

— відсутність механічних пошкоджень пакування;
— дотримання геометричних розмірів упаковки;
— здійснює контроль форми упаковки;
— здійснює контроль бренду продукції.

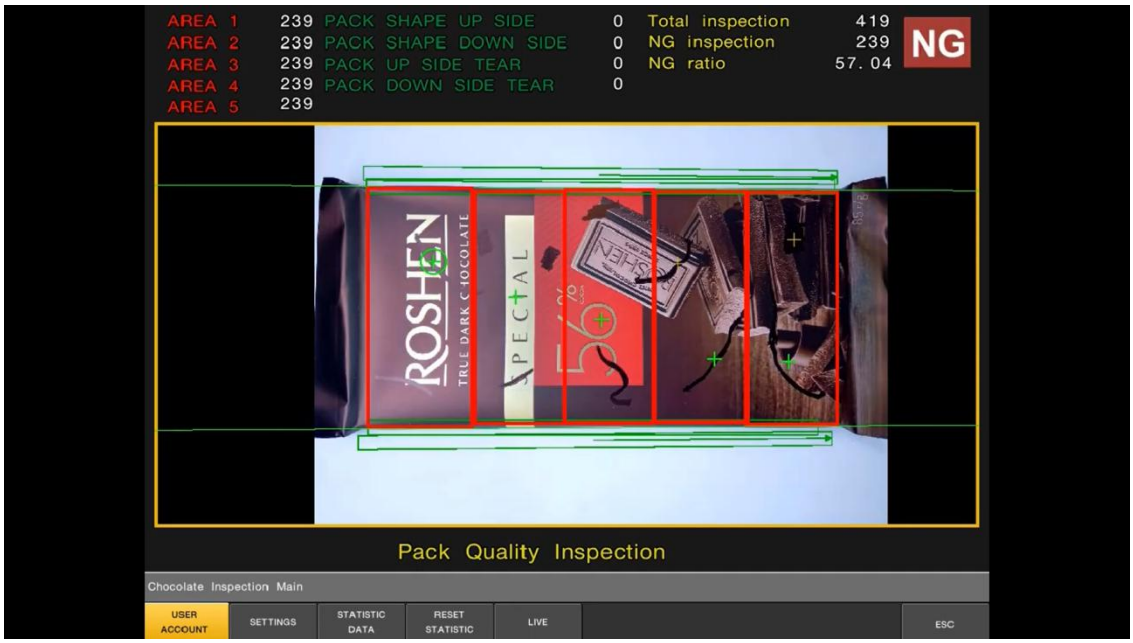

Рисунок 1.4 — Приклад роботи системи машинного зору

Окрім того, розглядається використання блокчейн-технологій для підвищення прозорості та відстеження якості виробничих ланцюгів. Децентралізована природа блокчейну дозволяє впроваджувати системи, в яких кожен етап виробництва та кожен учасник мають можливість перевірити якість продукції.

Зокрема, також досліджуються можливості використання великих даних (big data) у контексті контролю якості. Аналіз великих обсягів даних дозволяє виявляти тренди, вдосконалювати виробничі процеси та адаптувати стратегії контролю якості до змінюючих умов.

Усі ці технології можуть працювати разом або окремо, утворюючи потужні системи контролю якості, які сприяють виробникам підтримувати високий рівень якості продукції та оптимізувати виробничі процеси.

1.2 Вплив інноваційних технологій на контроль якості.

Контроль якості — це система заходів і процедур, спрямованих на визначення та забезпечення високої якості продукції або послуг. Основною метою контролю якості є гарантування того, що вироби чи послуги

відповідають встановленим стандартам та вимогам, задовольняючи очікування споживачів та забезпечуючи їх безпеку та ефективність [4].

Контроль якості включає в себе різні етапи та дії, серед яких можуть бути:

— розробка конкретних критеріїв та стандартів, які повинні бути виконані виробником для виробництва продукції або надання послуги;

— визначення стратегії та методів контролю якості, включаючи вибір методів вимірювань та інспекції;

— вимірювання та оцінка параметрів, що впливають на якість продукції або послуги. це може включати в себе лабораторні тести, інспекцію на виробничих лініях тощо;

— обробка та аналіз результатів контролю для визначення, чи відповідає продукція стандартам якості;

— при необхідності внесення змін у виробничий процес для покращення якості продукції;

— постійний контроль якості протягом всього виробничого процесу, включаючи контроль вхідних матеріалів та компонентів;

— зберігання записів і документів, які підтверджують якість продукції або послуги;

— виявлення потенційних ризиків для якості та прийняття заходів для їх уникнення або зменшення.

Контроль якості важливий для підприємств у всіх галузях, оскільки він допомагає зберегти довіру споживачів, покращити конкурентоспроможність та знизити витрати, пов'язані з виробництвом низькоякісної продукції. Він грає ключову роль у забезпеченні якості товарів та послуг, яку споживачі очікують і на яку вони мають право [4].

Оскільки це дозволяє організаціям постійно вдосконалювати свої процеси та продукти, залишаючись конкурентоспроможними, інновації є важливим компонентом практик управління якістю. Компанії, які віддають перевагу

інноваціям, з більшою ймовірністю будуть прибутковими вище середнього, з вищим ростом і прибутком.

Автоматизація, аналітика даних, штучний інтелект, машинне навчання та Інтернет речей (IoT) є прикладами передових технологій, які можуть допомогти підприємствам краще відстежувати та контролювати їх стандарти якості. Крім того, лінійне управління та інновації в покращенні якості можуть допомогти в оптимізації та вдосконаленні процесів і визначенні нових підходів до контролю та забезпечення якості [4].

Ефективне керівництво також потрібне для сприяння інноваціям у підприємстві. Майбутнє інновацій у управлінні якістю світле, із організаціями, які використовують передові технології та постійне вдосконалення для покращення якості продукції та послуг, зниження витрат і підвищення задоволення клієнтів.

Згідно з опитуванням ASQ, компанії, які надають перевагу інноваціям у своїх програмах управління якістю, відзначаються зростанням на 60% та прибутком вищим на 55%, ніж компанії, які не віддають перевагу інноваціям. Це демонструє важливість інноваційного мислення не лише для управління якістю, але й для загального успіху компанії [4].

Інновації є важливим елементом управління якістю, оскільки вони вводять нові та вдосконалені технології, які можуть покращити якість продуктів і послуг. Існує безліч інноваційних технологій, які допомагають підприємствам краще відстежувати та контролювати свої стандарти якості.

Автоматизація та робототехніка — це два приклади таких технологій, які можуть допомогти підвищити ефективність, зменшити помилки та поліпшити послідовність контролю якості. Аналітика даних — це ще одна передова технологія, яка може допомагати підприємствам краще розуміти потреби та уподобання своїх клієнтів і приймати рішення на основі даних для покращення якості [5].

Інші технології, такі як Штучний Інтелект (ШІ), Машинне Навчання (МН) та Інтернет Речей (IoT), можуть допомагати в управлінні якістю, надаючи

інсайти в реальному часі, моніторинг та управління, а також прогнозну аналітику. Ці технології можуть допомагати підприємствам виявляти та запобігати проблемам з якістю, а також покращувати загальну якість продуктів і послуг.

Для забезпечення високоякісних продуктів чи послуг важливий взаємозв'язок між інноваціями та постійним вдосконаленням у системі управління якістю. Інновації — це впровадження нових ідей чи методів, тоді як постійне вдосконалення передбачає впровадження невеликих, інкрементальних змін для поліпшення процесів з часом [5].

Інновації в управлінні якістю можуть призвести до розробки нових та більш ефективних методів відстеження та покращення якості продукту чи послуги. Наприклад, використання нових технологій, таких як автоматизація та штучний інтелект, може допомогти виявляти дефекти чи області для вдосконалення швидше та точніше.

З іншого боку, постійне вдосконалення передбачає постійний пошук шляхів удосконалення процесів та систем для того, щоб забезпечити, що продукти чи послуги відповідають або перевищують очікування клієнтів. Це включає в себе аналіз зворотного зв'язку від клієнтів, моніторинг виробничих процесів та впровадження невеличких змін для покращення ефективності та якості [5].

Інновації та постійне вдосконалення можуть допомогти організаціям виходити вперед у конкурентному середовищі, покращуючи якість, знижуючи витрати та збільшуючи задоволеність клієнтів. Приймаючи інновації та постійно вдосконалюючи свої практики управління якістю, організації можуть сприяти створенню культури відмінності, яка забезпечує довгостроковий успіх.

1.3 Роль веб-додатків у підвищенні ефективності системи контролю якості.

Веб-додатки у підвищенні ефективності системи контролю якості — це інтерактивні програми або засоби, розроблені для полегшення і оптимізації

процесів контролю якості в організації через використання веб-технологій. Вони дозволяють здійснювати збір, аналіз та спільний доступ до даних, пов'язаних з якістю продукції або послуг, у реальному часі та в режимі онлайн [6].

Веб-додатки дозволяють автоматизувати процес збору даних про якість продукції або послуг. Це може включати в себе створення онлайн-форм, де оператори можуть внести дані про виробництво, тестування та інші параметри контролю якості. Також вони надають можливість в реальному часі моніторити та аналізувати дані про якість, що збираються з різних джерел. Це допомагає оперативно виявляти відхилення в якості та вчасно реагувати на них.

Додатки для контролю якості можуть генерувати звіти та аналітичні дані, які допомагають приймати стратегічні рішення щодо покращення процесів виробництва та забезпечення якості. Ці додатки забезпечують можливість спільного доступу до інформації про якість для всіх учасників процесу контролю якості, включаючи менеджерів, інженерів, робочих тощо. Це сприяє кращій комунікації та співпраці в організації. І вони можуть автоматизувати рутинні завдання, такі як нагадування про перевірку якості, генерація сертифікатів якості та інші процеси, що можуть бути схильні до помилок [6].

Веб-додатки можуть легко інтегруватися з іншими системами у підприємстві, такими як системи управління виробництвом, системи складу, системи управління відносинами з клієнтами тощо. Вони грають важливу роль у підвищенні ефективності системи контролю якості через забезпечення доступу до даних, автоматизацію процесів і полегшення співпраці між різними сторонами. Розглянемо деякі способи, які веб-додатки сприяють підвищенню ефективності системи контролю якості:

Доступ до даних в режимі реального часу дозволяють співробітникам та керівництву миттєво отримувати доступ до важливих даних з будь-якого місця з підключенням до Інтернету. Це означає, що вони можуть відстежувати стан виробництва та контролювати якість продукції в режимі реального часу, що дозволяє негайно реагувати на будь-які аномалії чи проблеми [6].

Автоматизація процесів контролю якості можуть автоматизувати процеси збору та аналізу даних, що допомагає зменшити людський фактор та підвищує точність контролю якості. Наприклад, системи можуть автоматично аналізувати дані з сенсорів та сортувати продукцію за якістю. Веб-додатки дозволяють зберігати дані в одній централізованій системі, що полегшує доступ до них та управління ними. Це робить можливим швидкий пошук і аналіз інформації, що сприяє прийняттю інформованих рішень. А також вони дозволяють різним сторонам, включаючи постачальників та клієнтів, обмінюватися даними та інформацією про якість продукції. Це сприяє взаєморозумінню та підвищує транспарентність у ланцюжку постачання [7].

Веб-додатки можуть надавати засоби для аналізу даних та створення звітів про якість. Це допомагає виявляти тренди, недоліки та можливості для покращення.

За допомогою моніторингу віддалених об'єктів такі додатки можуть бути використані для віддаленого моніторингу об'єктів за допомогою сенсорів та веб-камер. Це особливо корисно для великих об'єктів або глобальних мереж виробництва.

1.4 Аналіз існуючих веб-додатків для контролю якості

Першим аналогом є система SAP Quality Management (QM). Вона є компонентом SAP ERP Central Component (ECC), який допомагає підприємствам впроваджувати та запускати процеси контролю якості. Він призначений для запобігання дефектам, забезпечення безперервного вдосконалення процесів і встановлення постійних програм контролю якості. Потенційні переваги для організації включають відповідність нормам якості виробництва, зниження операційних витрат і підвищення рівня задоволеності клієнтів [8].

SAP QM інтегрується з іншими компонентами ECC, такими як управління матеріалами, технічне обслуговування заводу, планування виробництва, продажі та розподіл, фінанси, контролінг і людські ресурси. Ця інтеграція

означає, що завдання SAP QM можна включати в завдання в межах цих інших компонентів ECC [8].

SAP QM використовується в основному в процесах контролю якості, які організації використовують для товарів під час проходження життєвого циклу продукту, включаючи перевірку товарів, коли вони надходять на об’єкт, коли вони проходять через виробництво та коли вони відвантажуються як готові товари (рисунок 1.5).

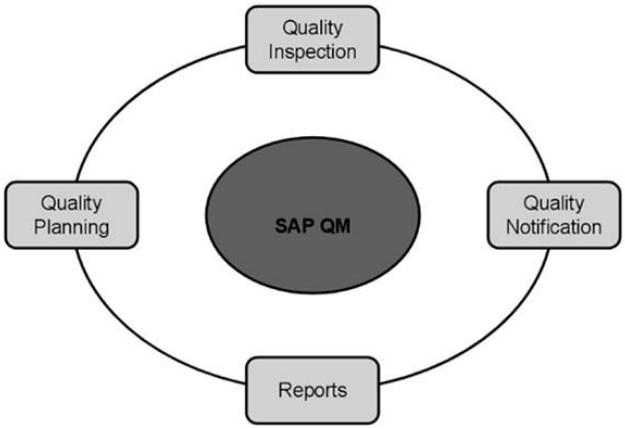

Рисунок 1.5 — SAP QM

SAP QM складається з інструментів для кількох взаємопов’язаних дій, які можна налаштувати для конкретних процесів організації. Кожен із цих інструментів складається з функцій, які є специфічними для цієї діяльності. Ці інструменти включають:

— планування якості (QM-PT), яке дозволяє налаштовувати та керувати планами перевірки якості, які визначають, які товари перевіряються, як відбуваються перевірки, характеристики товарів, що перевіряються, і тип обладнання, що використовується для виконання перевірок;

— перевірка якості (QM-IM), яка дозволяє визначити, чи відповідає продукт визначеним вимогам якості, і записати результати перевірок;

— сертифікати якості (QM-CA), які дозволяють засвідчити, що товари відповідають визначеним вимогам якості, клієнти можуть мати різні конкретні

критерії якості своїх товарів, і ви можете підтвердити сертифікати якості, щоб врахувати це;

— сповіщення про якість (QM-QN), які дозволяють записувати проблеми, які виникли з товарами, отриманими від постачальника, коли товари виробляються або про які повідомляють клієнти;

— контроль якості (QM-QC-AQC), що дозволяє керувати процесом контролю якості, наприклад, ви можете використовувати результати перевірки для оновлення рівнів якості; використовувати контрольні карти для контролю значень ознак; і оновлювати перевірки постачальників для процесу закупівель;

Другим аналогом є QMS Pro — це веб-додаток, який спеціалізується на управлінні якістю в організаціях. Ця платформа дозволяє планувати, моніторити та керувати процесами, пов'язаними з якістю продукції або послуг. Вона призначена для компаній, які прагнуть до відповідності стандартам якості та надає інструменти для забезпечення якості у виробництві або в наданні послуг [8].

Завдяки QMS Pro організації можуть ефективно визначати стандарти якості, створювати процедури та робочі інструкції, встановлювати вимоги до інспекції та тестування, а також відстежувати відхилення від них. Платформа дозволяє здійснювати аналіз результатів і видає звіти, які сприяють прийняттю рішень щодо покращення процесів та забезпечення якості (рисунок 1.6).

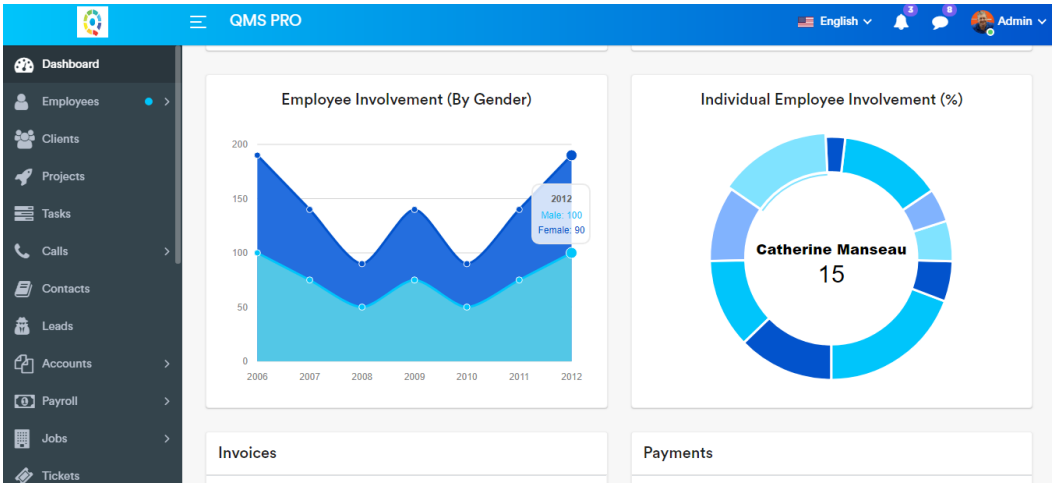

Рисунок 1.6 — QMS Pro

Окрім цього, QMS Pro може надавати інформацію щодо відповідності стандартам і регуляторним вимогам, що є важливим для підприємств, які мають дотримуватися певних нормативів. Програма також може підтримувати процес аудиту та сертифікації якості [8].

Однак, варто враховувати, що використання QMS Pro може вимагати інвестицій у вартість платформи та час на навчання персоналу для її використання. Також, вибір платформи для управління якістю повинен відповідати потребам конкретної організації та враховувати її розмір і галузь бізнесу.

Третім аналогом є Zoho Quality Management — це інноваційна веб-платформа, розроблена для покращення і контролю якості продукції та послуг в організаціях. Ця система надає різноманітні інструменти для створення, управління та моніторингу процесів, пов'язаних з якістю, і дозволяє підприємствам досягати високого стандарту якості та відповідати вимогам ринку [9].

Однією з ключових переваг Zoho Quality Management є його здатність створювати і впроваджувати стандарти якості для різних продуктів або послуг. Він дозволяє визначати стандарти, вимоги та критерії якості, а потім стежити за їх виконанням у реальному часі [9].

Важливою характеристикою є інтеграція Zoho Quality Management з іншими програмами та системами Zoho, що полегшує обмін інформацією та дозволяє організаціям використовувати повний спектр інструментів для управління бізнес-процесами (рисунок 1.7).

Зокрема, система забезпечує можливість генерувати докладні звіти і аналітику про якість, що сприяє прийняттю інформованих рішень для покращення якості продукції або послуг. Інтерфейс користувача простий та інтуїтивно зрозумілий, що робить роботу з системою зручною для користувачів на різних рівнях.

Однією з основних переваг Zoho Quality Management є здатність бути налаштованим під конкретні потреби кожної організації. Вона може бути

використана в різних галузях, від виробництва до обслуговування клієнтів, і надає інструменти для забезпечення високого стандарту якості та покращення конкурентоспроможності компанії [9].

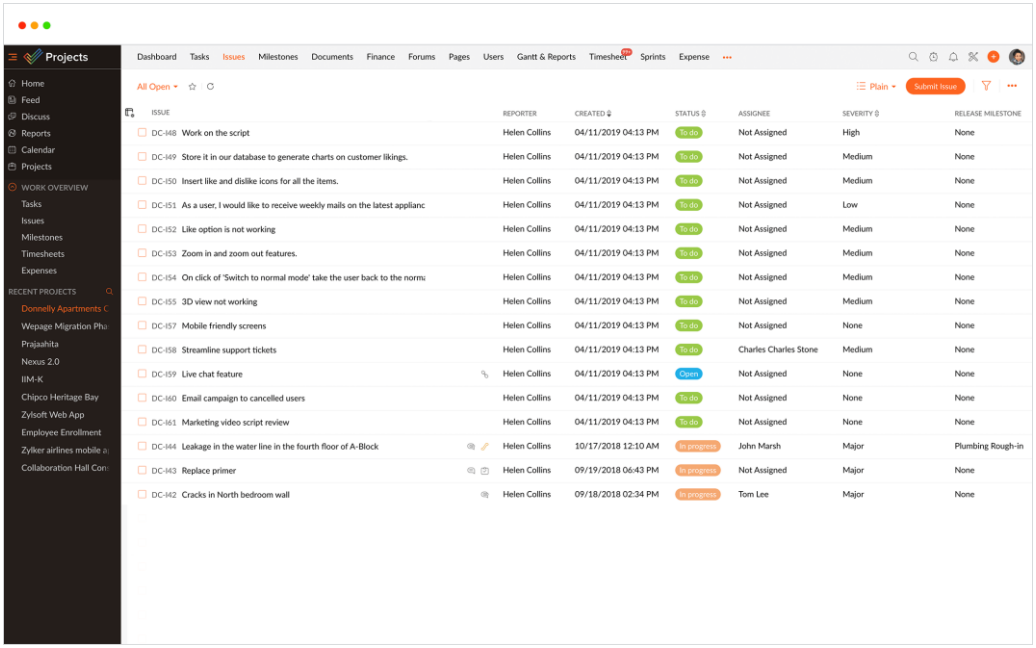

Рисунок 1.7 — Приклад Zoho Quality Management

Проте, важливо враховувати, що впровадження Zoho Quality Management може потребувати певного часу та ресурсів для навчання персоналу та налаштування системи. Також, вартість використання платформи може бути пов'язана із відповідною підпискою [9].

Останнім аналогом є ComplianceQuest. Це інноваційний веб-додаток, спрямований на забезпечення відповідності та контроль якості в галузі лікарських препаратів та медичних виробів. Цей інтегрований пакет рішень розроблений для оптимізації процесів управління якістю та забезпечення відповідності з найвищими стандартами та регуляціями в цій сфері [10].

Однією з ключових характеристик ComplianceQuest є його спроможність автоматизувати процеси, пов'язані з управлінням якістю продукції та дотриманням регуляторних вимог. Додаток дозволяє виробникам лікарських засобів та медичних виробів ефективно впоратися з вимогами щодо якості та безпеки своїх продуктів [10].

Однією з ключових функцій є автоматизована система ведення документації, яка дозволяє підтримувати та відстежувати всі необхідні документи, пов'язані з виробництвом та контролем якості. Це сприяє не лише забезпеченню дотримання стандартів, але і швидкому реагуванню на будь-які відхилення чи проблеми у виробничому процесі (рис 1.8).

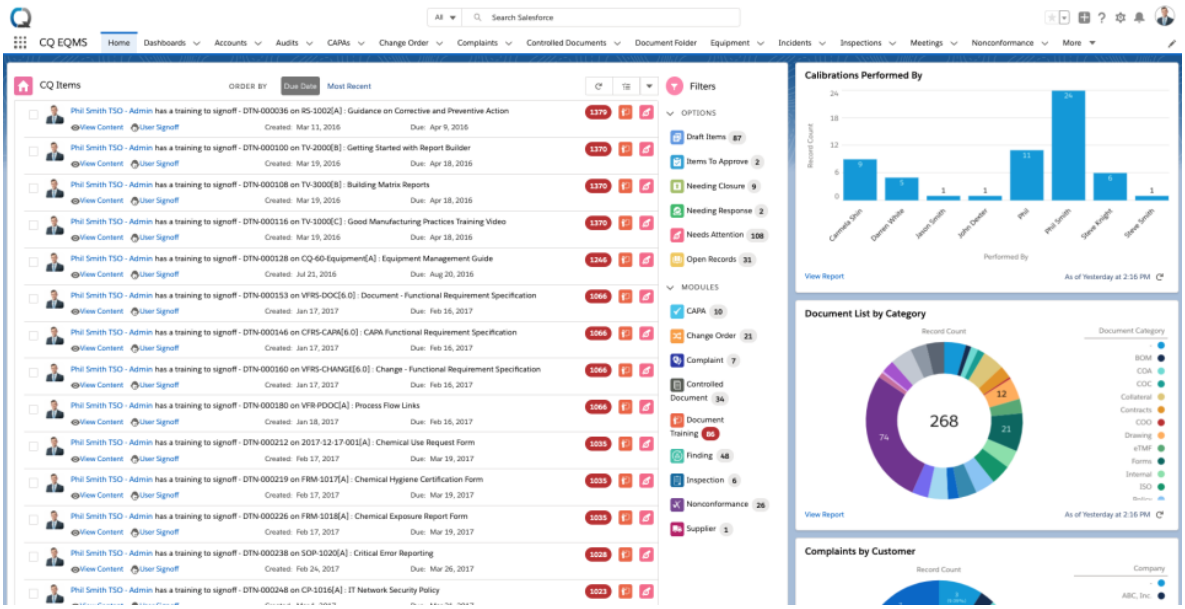

Рисунок 1.8 — Приклад ComplianceQuest

ComplianceQuest також включає модулі для відстеження відхилень та невідповідностей, що дозволяє оперативно реагувати на будь-які аномалії та негайно вирішувати проблеми якості. Це сприяє підтримці високих стандартів та запобіганню можливих ризиків для пацієнтів та споживачів.

Важливим елементом ComplianceQuest є інтеграція з іншими системами управління, що дозволяє створювати єдину інформаційну екосистему для управління всіма аспектами виробництва та контролю якості. Це підвищує ефективність та дозволяє здійснювати інтегрований підхід до управління якістю [10].

Загалом, ComplianceQuest представляє собою комплексне рішення, спрямоване на покращення процесів виробництва та контролю якості в лікарській та медичній індустріях, забезпечуючи дотримання високих стандартів якості та відповідність законодавству.

1.5 Аналіз та визначення якості продукції за допомогою інтелектуальних систем

Автоматизовані системи контролю якості — це програмне забезпечення та системи, які дозволяють автоматизувати процес контролю якості, включаючи системи моніторингу, аналізу даних та виявлення відхилень у виробництві.

Розглянемо виявлення відхилень у виробництві, на прикладі плитки шоколаду. Система захоплення зображень упаковки товарів призначена для використання у виробничих процесах для контролю якості, ідентифікації та відстеження упакованих товарів. Основні компоненти цієї системи включають камери або спеціалізовані пристрої, оптичні системи розпізнавання, системи освітлення, програмне забезпечення обробки зображень та систему зберігання та обробки даних.

Камери або пристрої розміщені на різних етапах виробництва автоматично захоплюють зображення упаковки товарів. У сучасних виробничих процесах використовуються різноманітні камери, спеціально призначені для захоплення зображень упаковки товарів. Ці камери володіють високою роздільною здатністю та можливістю пристосування до різних умов освітлення та оточуючого середовища. Вони відзначаються технологічною розробкою, що дозволяє автоматизовано захоплювати великі обсяги даних.

Ці камери використовуються для докладного аналізу упаковки товарів на різних етапах виробництва. Вони можуть бути обладнані спеціальними об'єктивами та сенсорами, які забезпечують високу чутливість та точність захоплення зображень. Технології застосовуються для автоматичного фокусування та корекції експозиції, щоб отримати якісні зображення навіть в умовах обмеженого освітлення.

Аналоги таких камер також включають у себе високорозвинуті моделі від провідних виробників, серед яких виділяються компанії, такі як Sony, Canon, Nikon та Basler. Виробники цих камер славляться своєю відмінною надійністю та високою якістю зображення, що робить їх популярними в різних галузях.

На ринку також присутні спеціалізовані виробники камер, спеціально призначених для застосувань у виробництві та автоматизованих системах контролю якості. Компанії, такі як Cognex та Keyence, відомі своїми високотехнологічними рішеннями, які відповідають вимогам сучасних виробничих процесів. Ці виробники спеціалізованих камер пропонують інноваційні та ефективні рішення для автоматизованих систем контролю якості, що сприяє підвищенню продуктивності та точності виробничих процесів у різних галузях промисловості. (рисунок 1.9).

Рисунок 1.9 — Приклад відеокамер фірми Cognex

Ці камери вирізняються широким спектром можливостей, включаючи вбудовані алгоритми розпізнавання об'єктів, можливість передачі даних в режимі реального часу та інтеграцію з іншими системами виробництва. Вони стають необхідним інструментом для сучасних виробництв, де важлива точність та ефективність контролю якості продукції.

Отримані зображення обробляються за допомогою комп'ютерного зору та алгоритмів штучного інтелекту для виявлення пошкоджень (подряпин, розривів), забруднень на упаковці та проблем у типографії.

Ці зображення потім передаються оптичним системам розпізнавання, які використовуються для автоматичної обробки та аналізу отриманих даних. Оптичні системи можуть розпізнавати текст, форми та кольори, використовуючи високоточні алгоритми.

Оптичні системи розпізнавання виглядають як своєрідні та функціонально відзначаються в сфері виробництва та контролю якості. Ці системи використовуються для автоматизованої обробки та аналізу великої кількості зображень, отриманих від камер або інших оптичних пристроїв (рисунок 1.10).

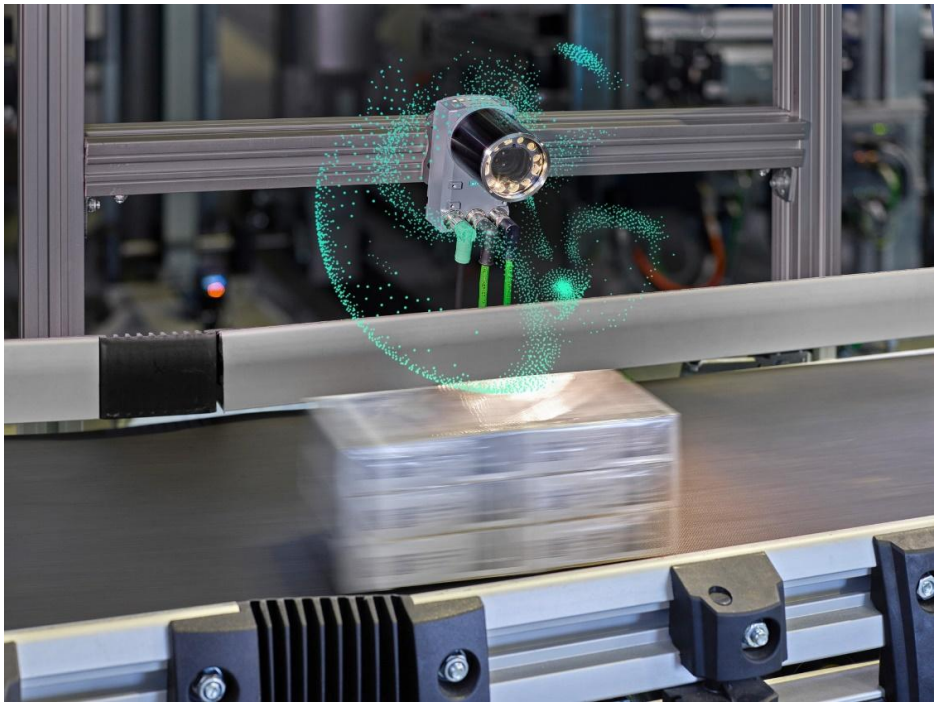

Рисунок 1.10 — Система оптичної ідентифікації

Їхнім головним завданням є розпізнавання об'єктів, тексту, форм та інших характеристик на зображеннях. Це досягається завдяки використанню високоточних алгоритмів обробки зображень, які можуть виявляти та аналізувати різноманітні деталі.

Оптичні системи розпізнавання пристосовані до різних завдань, таких як розпізнавання символів на упаковці, ідентифікація дефектів або відстеження руху об'єктів. Вони можуть працювати в реальному часі, забезпечуючи швидку обробку зображень та миттєвий вихід результатів.

Ці системи використовують різні типи сенсорів та об'єктивів, щоб отримати якісні та деталізовані зображення. Вони можуть автоматично коригувати параметри експозиції та фокусу для оптимальної якості результатів.

Оптичні системи розпізнавання грають важливу роль у виробничому процесі, де точність та швидкість є вирішальними факторами. Вони можуть використовуватися в різних галузях, включаючи харчову, фармацевтичну та автомобільну промисловості, де вимоги до якості та безпеки є високими.

Зокрема, системи освітлення грають теж важливу роль у забезпеченні оптимального освітлення для отримання якісних зображень та полегшенні роботи оптичних систем розпізнавання. Вони можуть адаптуватися до різних умов освітлення, щоб забезпечити стабільні умови для захоплення даних.

Отож, програмне забезпечення обробки зображень використовується для складання та аналізу даних, отриманих від камер та оптичних систем. Воно включає в себе алгоритми розпізнавання об'єктів, порівняння зі стандартами якості та відстеження об'єктів протягом виробничого процесу. Система зберігання та обробки даних забезпечує ефективне зберігання інформації про упаковку товарів. Ці дані можуть бути використані для подальшого аналізу, вдосконалення виробничих процесів та забезпечення високої якості продукції.

Робочий процес включає в себе захоплення зображень, їхню обробку та аналіз, виявлення відхилень від стандартів та відстеження об'єктів на різних етапах виробництва. У разі виявлення невідповідності встановленим стандартам, система може генерувати сигнали або сповіщення для операторів або автоматичної системи керування виробництвом. Усі виявлені аномалії класифікуються та аналізуються. Після аналізу система генерує звіт, де фіксується виявлені проблеми, та цей звіт автоматично відправляється у відповідну базу даних. Користувачі (різні рівні управління або інженери контролю якості) можуть отримати доступ до цих звітів через веб-додаток. Вони можуть переглядати звіти, завантажувати їх або отримувати сповіщення про виявлені проблеми.

2 АНАЛІЗ ТА ВИБІР ІНСТРУМЕНТІВ ДЛЯ РОЗРОБКИ ВЕБ-ДОДАТКУ ДЛЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ КОНТРОЛЮ ЯКОСТІ ВИРОБНИЦТВА

2.1 Архітектура «клієнт-серверної» взаємодії

Архітектура клієнт-сервер вказує на систему, яка забезпечує, доставляє та управляє більшістю ресурсів та послуг, які клієнт запитує. У цій моделі всі запити та послуги надсилаються через мережу, і це також називається мережевою обчислювальною моделлю чи мережею клієнт-сервер.

Ця архітектура відома як модель клієнт-сервер, є мережевим додатком, який розбиває завдання та навантаження між клієнтами та серверами, які розташовані на одній системі чи пов'язані комп'ютерною мережею. Зазвичай вона має кілька робочих станцій користувачів, ПК чи інших пристроїв, підключених до центрального сервера через Інтернет-з'єднання чи іншу мережу. Клієнт надсилає запит на дані, і сервер приймає та враховує запит, надсилаючи пакети даних назад користувачеві, якому вони потрібні [11].

Розглянемо її характеристики:

— клієнтські та серверні машини зазвичай потребують різних апаратних та програмних ресурсів і можуть бути від різних вендорів;

— мережа має горизонтальну масштабованість, яка збільшує кількість клієнтських машин, та вертикальну масштабованість, яка переміщає весь процес на більш потужні сервери або в конфігурацію з кількома серверами;

— один комп'ютерний сервер може надавати одночасно кілька послуг, хоча для кожної послуги потрібна окрема серверна програма;

— як клієнтські, так і серверні застосунки взаємодіють безпосередньо з транспортним протоколом, цей процес встановлює зв'язок та дозволяє сутностям відправляти та отримувати інформацію;

— як клієнтські, так і серверні комп'ютери потребують повного стеку протоколів, транспортний протокол використовує протоколи нижчого рівня для відправлення та отримання окремих повідомлень.

Наступна схема клієнт-сервер показує основи архітектури (рисунок 2.1).

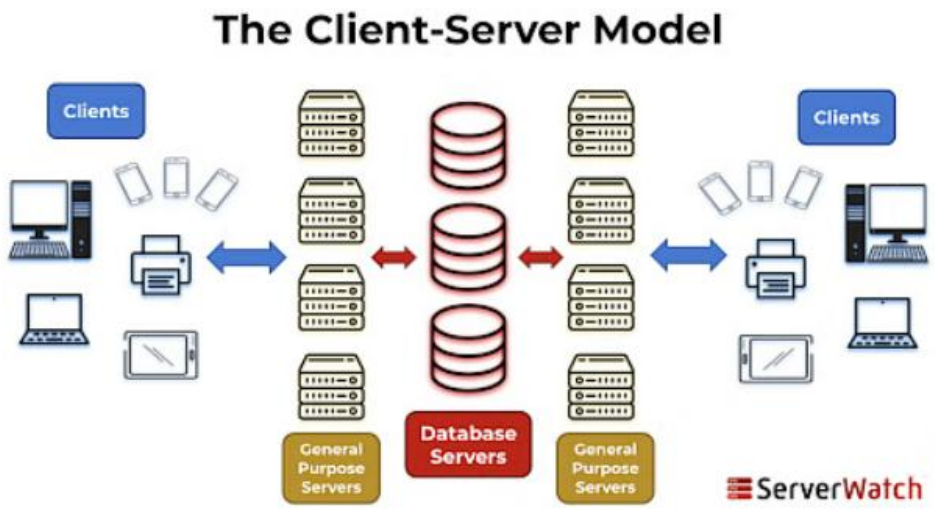

Рисунок 2.1 — Основи архітектури клієнт-серверної взаємодії

Архітектура клієнт-сервер має свої позитивні та негативні аспекти для сучасних цифрових споживачів. Розглянемо її переваги. Однією з ключових переваг архітектури клієнт-сервер є її модульність і масштабованість. Кожен компонент — клієнт та сервер функціонує незалежно, що сприяє легкості утримання та розвитку системи. Це стає особливо корисним у великих проектах, де розділення обов'язків полегшує розробку і підтримку. Додатково, архітектура клієнт-сервер дозволяє забезпечити безпеку та контроль доступу. Сервер може управляти доступом до ресурсів, визначати права користувачів, а також забезпечувати конфіденційність даних. Це робить архітектуру клієнт-сервер ефективним інструментом для створення надійних та безпечних систем. Також, завдяки цій архітектурі, можливо забезпечити ефективний обмін даними між клієнтом і сервером. Це важливо для забезпечення відзивчивості системи та швидкості обробки запитів. Усе це робить архітектуру клієнт-сервер важливим інструментом для розробки розподілених та ефективних додатків [11].

Звісно, архітектура клієнт-сервер не є абсолютно безпечною. Серед недоліків є те, що система може стати менш стійкою в разі відмови сервера. Якщо сервер перестає працювати або навіть недоступний через технічні або мережеві проблеми, то це може призвести до повного відключення клієнтських

додатків від доступу до ресурсів. Ще однією недолікою є те, що збільшення кількості клієнтів може призвести до перевантаження сервера. Якщо кількість запитів збільшується, сервер може не встигати ефективно відповідати на всі запити, що призводить до зниження продуктивності та затримок у обслуговуванні. Також, архітектура клієнт-сервер може викликати проблеми у випадку нестабільної мережі. Якщо з'єднання між клієнтом і сервером недостатньо стійке або має велику затримку, це може призвести до негативного впливу на швидкість і ефективність взаємодії. Недоліком також є висока залежність від сервера. Якщо сервер виявляється вразливим до атак або несправностей, це може суттєво підірвати функціональність усієї системи. Такий сценарій може призвести до втрати доступу до важливих ресурсів та обмеження можливостей користувачів [11].

Трирівнева архітектура клієнт-сервер складається з рівня презентації, відомого як інтерфейс користувача, рівня додатків, відомого як рівень служб, та рівня даних, який включає сервер баз даних. Трирівневу архітектуру можна розділити на три частини:

— рівень презентації (або клієнтський рівень) — цей рівень відповідає за інтерфейс користувача;

— рівень додатків (або бізнес-рівень) — цей рівень обробляє деталізовану обробку;

— рівень даних (або рівень даних) — цей рівень зберігає інформацію.

Система клієнта керує рівнем презентації; сервер додатків відповідає за рівень додатків, а серверна система контролює рівень даних.

На рисунку 2.2 зображено модель трирівневої архітектури клієнт-сервер .

Рисунок 2.2 — Модель трирівневої архітектури

2.2 Основи та практичні аспекти розробки web-додатків

Розробка веб-додатку, який автоматизує процес контролю якості, стає невід'ємною частиною сучасного виробництва. Забезпечуючи швидкий та ефективний моніторинг усіх етапів виробничого процесу, такий додаток дозволяє збільшити ефективність, знизити витрати та мінімізувати ризик виробничих невдач [12].

Веб-додаток пропонує зручний та інтуїтивно зрозумілий інтерфейс для користувачів, що дозволяє миттєво отримувати доступ до ключової інформації щодо якості виробництва. Завдяки цьому додатку, підприємство отримує повний контроль над якістю продукції, забезпечуючи надійність та відповідність стандартам.

Розробка веб-додатків — це комплексний процес, який включає в себе ряд ключових компонентів, необхідних для створення функціонального та ефективного продукту. Одним із головних елементів є фронтенд, що відповідає за взаємодію користувача з додатком через інтерфейс. Відправлення запитів до серверу та отримання відповідей відбувається через HTTP-протокол, а для забезпечення динамічності та взаємодії з базою даних використовують технології, такі як JavaScript, HTML та CSS [12].

Бекенд є іншим ключовим компонентом, відповідальним за обробку логіки додатку, зберігання та управління даними. Технології серверної розробки, такі як Node.js, Python, Ruby чи Java, використовуються для створення серверу, який обробляє запити від фронтенда та взаємодіє з базою даних.

База даних грає ключову роль у зберіганні та управлінні даними додатку. Різні системи управління базами даних (СУБД), такі як MySQL, PostgreSQL, MongoDB чи SQLite, використовуються для забезпечення ефективного зберігання та витягування інформації [12].

Також до основних компонентів розробки веб-додатків входить система контролю версій для управління кодом проекту, тестування для забезпечення

якості програмного продукту, а також розгортання та хостинг для публікації додатку в Інтернеті.

У розробці веб-додатків архітектура відіграє важливу роль у забезпеченні стабільності та безпеки. Розробники використовують певні архітектурні підходи, такі як мікросервісна архітектура або монолітна архітектура, для створення основи додатку. Вибір архітектури може вплинути на масштабованість, ефективність та можливості безпеки системи [12].

Безпека веб-додатків є надзвичайно важливою складовою. Інтеграція механізмів аутентифікації та авторизації, шифрування даних, валідація введених користувачем даних та інші практики безпеки допомагають запобігти атакам та зберегти конфіденційність та цілісність інформації.

Оптимізація відповідальна за покращення швидкодії та продуктивності веб-додатку. Це включає в себе оптимізацію коду, роботу з базою даних для швидшого витягування та оновлення інформації, кешування та інші стратегії для зниження часу завантаження сторінок та відповідей сервера [12].

Тестування грає ключову роль у забезпеченні якості продукту. Від модульних тестів до інтеграційних та системних тестів, тестування дозволяє виявляти та усувати помилки, перевіряти функціональність та забезпечувати коректну взаємодію всіх компонентів. Автоматизоване тестування сприяє ефективній розробці та збереженню стабільності додатку під час регулярних оновлень.

Хмарні технології відіграють важливу та перспективну роль у створенні веб-додатків, і їх важливість можна визначити з декількох ключових поглядів.

По-перше, хмарні технології надають розробникам доступ до великого спектру ресурсів, таких як обчислювальна потужність, зберігання даних, та інші сервіси, безпосередньо через Інтернет. Це дозволяє компаніям ефективно масштабувати свої веб-додатки в залежності від зростання обсягу користувачів або роботи з великими об'ємами даних.

По-друге, хмарні технології забезпечують високу доступність та надійність. За допомогою хмарних сервісів можна створювати резервні копії

даних, використовувати географічно розподілені сервери для забезпечення стабільності роботи, а також автоматизовані засоби моніторингу для вчасного виявлення та вирішення проблем.

По-третє, ефективне використання хмарних технологій дозволяє зменшити витрати на обладнання та інфраструктуру. Компанії можуть користуватися послугами, які вони реально використовують, не інвестуючи в власне обладнання, що дозволяє економити ресурси та знижувати витрати.

По-четверте, хмарні технології полегшують розгортання та оновлення веб-додатків. Вони надають інструменти для автоматизованого розгортання та керування версіями, що спрощує процес релізів та дозволяє швидко впроваджувати нові функції або виправлення.

По-п'яте, хмарні технології є основою для впровадження інноваційних рішень, таких як штучний інтелект, аналіз даних та Інтернет речей (IoT). З використанням хмарних платформ можна легко інтегрувати ці нові технології в веб-додатки, що розширює їхні можливості та функціональність.

Отож, взаємодія цих компонентів визначає ефективність, безпеку та функціональність веб-додатку, надаючи користувачам зручний та надійний інструмент для вирішення конкретних завдань чи задач [12].

2.3 Аналіз та вибір середовища та технологій для розробки web-додатку

Вибір середовища розробки є ключовим етапом у процесі створення веб-додатків. Кожне середовище має свої унікальні можливості, спеціалізовані для конкретних завдань. Таким чином, важливо здійснити порівняння різних варіантів середовищ розробки і вибрати оптимальний для вирішення конкретної задачі. У нашому випадку, метою є розробка веб-додатку для автоматизованої системи контролю якості виробництва [13].

Розглянемо різні варіанти та оберемо найоптимальніше середовище для розробки.

Visual Studio Code (VS Code) є потужним та популярним текстовим редактором, розробленим компанією Microsoft. Він визначається своєю

легкістю, гнучкістю та великим спектром можливостей для розробки програмного забезпечення. У порівнянні з іншими інтегрованими середовищами розробки (IDE), VS Code відрізняється своєю швидкістю завантаження та реагування на дії користувача, що робить його ідеальним інструментом для широкого спектру розробників [13].

VS Code підтримує багато мов програмування, включаючи популярні такі як JavaScript, Python, C#, Java та інші. Він оснащений вбудованими інструментами для роботи з Git, що полегшує ведення контролю версій і співпрацю в команді. Однією з ключових особливостей є високий рівень настроювання, який дозволяє кожному розробнику налаштовувати редактор під свої власні потреби та стиль (рисунок 2.3).

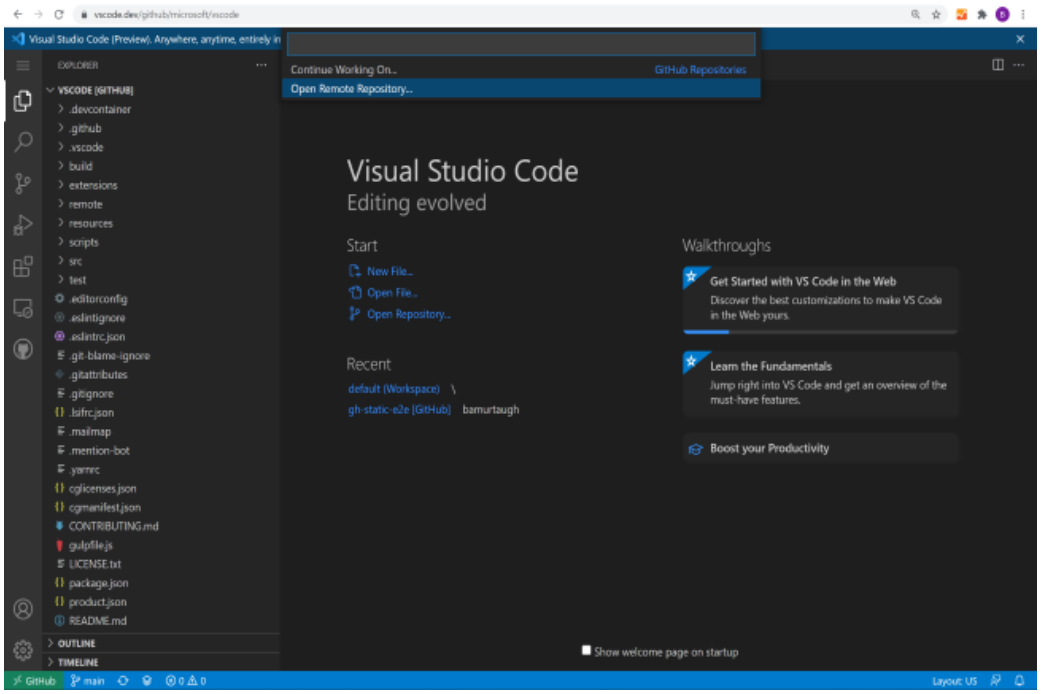

Рисунок 2.3 — Зовнішній вигляд середовища VS Code

Розширюваність є ще однією сильною стороною VS Code. Завдяки великій кількості розширень, розроблених спільнотою та власними командами, редактор може бути розширений для роботи з практично будь-якою мовою програмування чи технологією. Це дозволяє розробникам використовувати VS Code для різних проектів та задач, від веб-розробки до наукових досліджень [13].

Зручний інтерфейс користувача VS Code сприяє продуктивності, забезпечуючи зручну навігацію та швидкий доступ до необхідних функцій. Розробники також цінують інтегровані засоби для налагодження коду та відладки, які допомагають виявляти та виправляти помилки швидше та ефективніше.

IntelliJ IDEA визначається як високоефективне та інтелектуальне середовище розробки, створене компанією JetBrains. Це інтегроване середовище призначене переважно для роботи з мовами Java, Kotlin, Scala, Groovy, і іншими. IntelliJ IDEA славиться своєю здатністю автоматизувати багато аспектів розробки, що полегшує життя розробників і дозволяє їм фокусуватися на творчості [14].

Однією з ключових особливостей IntelliJ IDEA є його інтелектуальний редактор коду, який надає розумні підказки, рефакторинг, і автоматичне завершення коду. Редактор розуміє контекст коду і може пропонувати оптимальні варіанти доповнення або виправлення помилок.

IntelliJ IDEA підтримує розробку великої кількості фреймворків і технологій, що робить його відмінним інструментом для корпоративних та веб-проектів. Відзначається високою продуктивністю, сприяє підтримці тестування, автоматизованому відлагодженню, та іншим розробничим процесам.

Іншою важливою характеристикою IntelliJ IDEA є його система плагінів, яка дозволяє розширювати функціональність редактора та пристосовувати його до конкретних потреб розробника. JetBrains підтримує та поновлює це середовище розробки, роблячи його відмінним вибором для професіоналів.

Інтерфейс користувача IntelliJ IDEA відзначається зручністю та інтуїтивною навігацією, що дозволяє розробникам швидко та легко зорієнтуватися в проекті. Всі ці фактори роблять IntelliJ IDEA важливим інструментом для розробки програмного забезпечення в мовах, що базуються на JVM (Java Virtual Machine) [14].

Sublime Text — це легкий, швидкий та дуже популярний текстовий редактор, який надав дуже зручний інтерфейс для розробників. Його головними

особливостями є простота використання, висока продуктивність та багатофункціональність. Завдяки цим характеристикам Sublime Text став популярним серед розробників різного рівня досвіду.

Один із ключових елементів привабливості Sublime Text полягає в його легкості та швидкості. Редактор відзначається мінімалістичним інтерфейсом, який не заважає розробникові та полегшує швидкий доступ до функцій. Підтримка великої кількості мов програмування і лексичне підсвічування коду роблять Sublime Text універсальним для різних типів проектів [14].

Sublime Text має високий рівень настроювання, що дозволяє користувачам адаптувати редактор під свої потреби. Велика кількість плагінів та пакетів розширює його функціональність, надаючи додаткові можливості та інструменти для роботи з різними мовами та технологіями (рисунок 2.4).

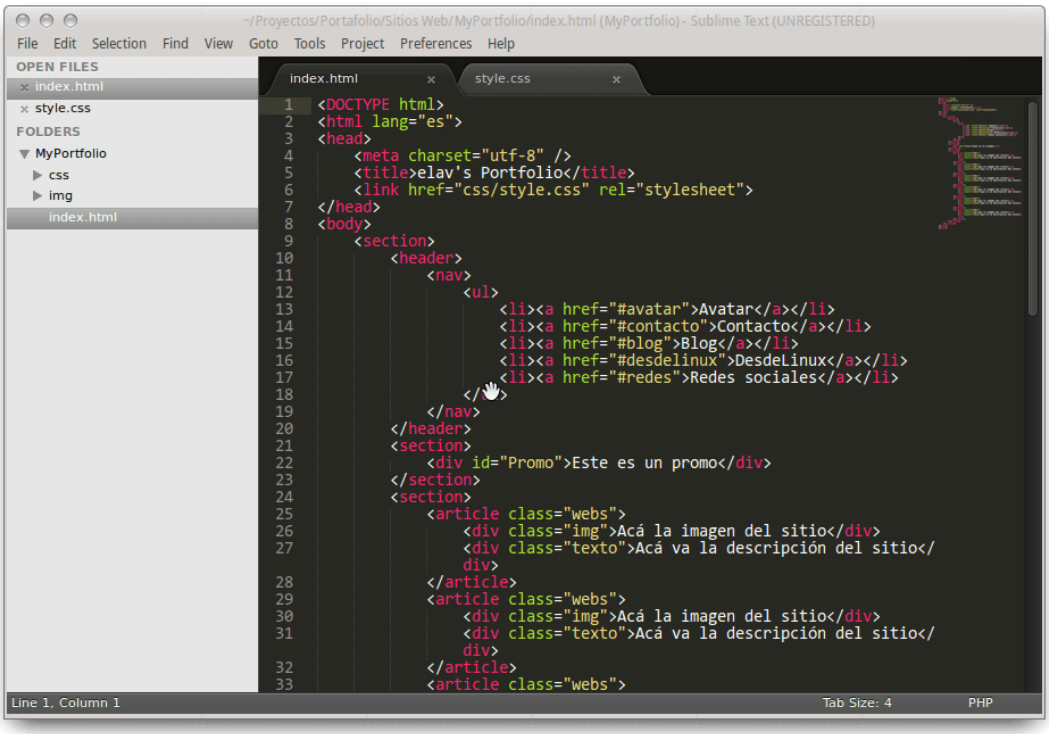

Рисунок 2.4 — Зовнішній вигляд середовища Sublime Text

Важливою особливістю Sublime Text є його активна спільнота та постійна підтримка розробниками. Це дозволяє швидко виправляти помилки, вдосконалювати функціонал та забезпечувати високу якість редактора [14].

Atom — це вільний та відкритий текстовий редактор, створений компанією GitHub. Відзначається своєю легкістю використання та високою ступенем розширюваності. Atom визначається своїм інтуїтивним інтерфейсом, що робить його привабливим для широкого кола розробників.

Однією з ключових особливостей Atom є його розширюваність. Редактор підтримує велику кількість пакетів і тем, що дозволяє користувачам налаштовувати його функціонал для відповіді на свої потреби. Це робить Atom гнучким інструментом для роботи з різними мовами програмування та технологіями [14].

Atom надає розумні підказки, автоматичне доповнення та вбудовану систему пошуку, що полегшує роботу розробників і дозволяє їм більше уваги приділяти творчості. Інтерфейс користувача розрахований на зручну навігацію та візуальну ясність, що допомагає швидше реалізовувати ідеї.

Високий рівень підтримки різних мов програмування робить його універсальним інструментом для розробників. Ви можете легко працювати з HTML, CSS, Python, JavaScript і багатьма іншими мовами, користуючись зручним середовищем Atom.

Однією з переваг Atom є його інтеграція з платформою GitHub. Відправлення змін в репозиторій, перегляд коду та співпраця в команді відбувається безпосередньо в редакторі, забезпечуючи зручність та швидкість обміну інформацією [15].

Завдяки системі пакетів та розширень, ви можете налаштувати Atom під свої потреби, додаючи функціонал та покращуючи продуктивність. Розробники також впізнають автоматичне доповнення коду, підказки та швидке виявлення помилок, що значно полегшує процес написання коду.

Не тільки редактор, Atom — це спільнота та ресурси, які відкривають перед вами безмежні можливості розробки. Завдяки активній спільноті і обміну досвідом, ви можете швидко розвивати свої навички та ефективно працювати над проектами. Atom - це інструмент, який не просто використовується, але

стає частиною вашого робочого процесу, допомагаючи вам досягати успіху у світі розробки програмного забезпечення.

Розглянемо порівняльну таблицю із середовищами для програмування (таблиця 2.1).

Таблиця 2.1 — Порівняльна таблиця із середовищами

Особливості	Atom	Sublime Text	IntelliJ IDEA	Visual Studio Code
Вартість	Безкоштовний	Пробний період, після чого — платно	Існує Community— версія (безкоштовна), Professional— версія (платна)	Безкоштовний
Розширюваність	Висока	Висока	Висока	Висока
Підтримка мов програмування	Широкий спектр	Широкий спектр	Головна: Java, Kotlin; Плагіни: багато мов	Широкий спектр
Швидкість	Помірна	Висока	Висока	Висока
Інтерфейс користувача	Зручний та налаштовуваний	Привабливий, зручний у використанні	Професійний та інтуїтивно зрозумілий	Зручний та сучасний
Підтримка Git	Вбудована	Вбудована	Вбудована	Вбудована
Автоматичне доповнення	Так	Так	Так	Так

Отож, розглянувши усі переваги та недоліки було обрано як середовище VS Code.

2.3.1 Вибір технологій для клієнтської частини

Для розробки клієнтської частини необхідні такі технології як HTML, CSS, JS та фреймворк Bootstrap. Розглянемо кожну з технологій.

HTML (HyperText Markup Language) є основною мовою розмітки для створення веб-сторінок та веб-додатків. Вона визначає структуру та семантику контенту веб-сторінки за допомогою тегів. Ось деякі ключові аспекти HTML:

— HTML використовує теги для розміщення та організації контенту, теги починаються з `<` і закінчуються `>`;

— HTML документ має стандартну структуру, що включає `<html>`, `<head>` (заголовок), та `<body>` (тіло), в `<head>` можна вказати заголовок сторінки, підключити зовнішні стилі та скрипти;

— елементи є структурними одиницями, які визначають контент, а атрибути надають додаткову інформацію елементам;

— має багато тегів, які надають семантику структурі документа. Наприклад, теги `<header>`, `<nav>`, `<article>`, `<section>`, `<footer>` допомагають чітко визначити частини сторінки та їхню призначеність;

— підтримує вставку різноманітного мультимедіа, такого як зображення (`<img>`), аудіо (`<audio>`), та відео (`<video>`), вони можуть вбудовуватися безпосередньо в сторінку або посилати на зовнішні ресурси;

— дозволяє створювати форми для взаємодії з користувачем, теги, такі як `<form>`, `<input>`, `<select>`, та `<button>`, дозволяють збирати та обробляти дані.

CSS, або Cascading Style Sheets, є мовою стилізації для веб-документів. Вона надає можливість визначати зовнішній вигляд елементів HTML, керуючи їхнім розташуванням, кольорами, шрифтами та іншими стилевими властивостями. CSS дозволяє розробникам створювати естетично приємний та відповідний дизайн для веб-сторінок без прив'язки стилів безпосередньо в HTML-коді [15].

Розмір, колір, тип шрифту, відступи, вирівнювання — це всі аспекти, які можна контролювати за допомогою CSS. Важливою особливістю CSS є

каскадність, що дозволяє визначати стилі на різних рівнях, забезпечуючи гнучкість та контроль над зовнішнім виглядом елементів [16].

Застосовуючи CSS, розробники можуть створювати різноманітні макети, визначати анімації, робити веб-сторінки адаптивними для різних пристроїв та забезпечувати високий рівень користувацької доступності. CSS допомагає створити єдинообразний та привабливий вигляд для веб-додатків, сприяючи покращенню користувацького досвіду та ефективності веб-розробки [16].

JavaScript, чи просто JS, є мовою програмування, яка широко використовується для розробки веб-додатків. Його можливості не обмежуються лише веб-браузерами, і він також використовується у середовищі серверного програмування (зокрема, за допомогою Node.js). JavaScript є невід'ємною частиною повноцінного стеку веб-розробки, взаємодіючи з HTML та CSS для створення динамічних та інтерактивних веб-сайтів.

JS використовується для обробки подій, взаємодії з користувачем та маніпулювання DOM (Document Object Model). Він забезпечує веб-сайти можливістю реагувати на події, такі як кліки миші, натискання клавіш, анімації та багато інших. Вона підтримує об'єктно-орієнтований, функціональний та процедурний підходи до програмування, що робить його гнучким та потужним інструментом для розробників [16].

Однією з ключових особливостей JS є його можливість взаємодії з сервером без перезавантаження сторінки, завдяки асинхронним запитам AJAX (Asynchronous JavaScript and XML). Це робить його ідеальним для розробки односторінкових додатків (SPA).

Нові стандарти, такі як ECMAScript 6 і пізніші версії, додають нові можливості та зручності для розробників, включаючи класи, стрілкові функції, деструктуризацію та інші покращення мови [17].

Однією з головних переваг є широке використання — вона підтримується у більшості сучасних веб-браузерів, а також може використовуватися на серверній стороні за допомогою платформи Node.js.

Завдяки своїй динамічності, JavaScript дозволяє реагувати на події, які відбуваються на веб-сторінці, такі як кліки миші чи введення користувача. Це робить його ідеальним для створення живих та інтерактивних інтерфейсів.

З іншого боку, використання JavaScript може призвести до проблем, таких як крос-браузерні розбіжності, коли код може вести себе по-різному в різних браузерах. Також, оскільки весь код виконується на клієнтському браузері, це може збільшувати обсяг роботи для пристрою користувача та впливати на швидкість завантаження сторінки [17].

З безпекового погляду, JavaScript може бути вразливим до атак, таких як впровадження зловмисного коду (XSS). Тому важливо дотримуватися належних стандартів безпеки при розробці.

Bootstrap є однією з популярних технологій для розробки клієнтської частини веб-додатків. Він представляє собою відкритий фреймворк для розробки веб-сайтів і веб-додатків з використанням HTML, CSS і JavaScript. Розглянемо деякі основні аспекти Bootstrap та його переваги:

— надає готові компоненти та стилі для створення мобільно-адаптивних інтерфейсів, це означає, що веб- додаток буде добре виглядати та працювати на різних пристроях, від маленьких мобільних телефонів до великих екранів настільних комп'ютерів.

— постачається з багатьма готовими компонентами, такими як навігаційні панелі, форми, кнопки, каруселі, модальні вікна та багато інших, це значно спрощує процес створення стильних та функціональних інтерфейсів.

— використовує гнучку систему сітки, яка полегшує розташування елементів на сторінці, це сприяє вирівнюванню та адаптації макету до різних розмірів екрану.

— визначає стандартні стилі для багатьох елементів HTML, тому розробникам не потрібно занадто вдаватися в ручну стилізацію, класи Bootstrap можна легко додавати до HTML— коду для отримання бажаного вигляду та функціоналу.

— підтримує всі сучасні браузери, що робить його надійним фреймворком для розробки веб- додатків з урахуванням різних браузерних вимог.

Інтеграція Bootstrap виявляється важливим кроком для оптимізації та стандартизації вигляду вашого веб-додатка, прискорюючи весь процес розробки. Bootstrap надає готові компоненти та стилі, що дозволяють створювати ефективний та сучасний дизайн без необхідності великої кількості власного CSS та JavaScript коду. Інтеграція Bootstrap — це лише початок, і далі слід враховувати інші бібліотеки, фреймворки та інструменти, які можуть допомогти вам досягти повного потенціалу вашого веб-додатка [18].

2.3.2 Вибір технологій для серверної частини

Серверна частина веб-додатків є ключовою складовою, яка відповідає за обробку запитів від клієнтської сторони, взаємодію з базою даних, та забезпечення логіки додатка. Для створення ефективної та надійної серверної частини було обрано Python, Node.js та базу даних MySQL [18].

Розглянемо детальніше кожен із них.

Python — це потужна та універсальна мова програмування, яка визначається своєю простотою та читабельністю коду. Заснована у 1991 році Гвідо ван Россумом, вона швидко завоювала популярність завдяки своїй лаконічності та великому співтовариству розробників. Python дозволяє вирішувати широкий спектр завдань, починаючи від веб-розробки і закінчуючи штучним інтелектом.

Основна перевага Python — його простота та читабельність коду. Це робить мову доступною навіть для новачків, а також допомагає підтримувати та розширювати проекти. Python використовує динамічну типізацію, що дозволяє зосереджуватися на розробці, а не на оголошенні типів даних.

Однією з головних рис Python є його універсальність. Він використовується для різноманіття завдань, включаючи веб-розробку, аналіз даних, наукове моделювання та робототехніку. Розширюваність мови забезпечується багатофункціональністю та великою кількістю бібліотек.

Python є мовою високого рівня, що дозволяє розробникам концентруватися на логіці програми, а не на деталях роботи з пам'яттю чи обробці виключень. Це сприяє швидкій розробці та спрощує тестування коду.

Python активно використовується в галузі штучного інтелекту та машинного навчання. Інструменти, такі як TensorFlow та PyTorch, використовуються для навчання нейронних мереж та розв'язання складних завдань.

Мова має об'єктно-орієнтовану структуру, що сприяє структуруванню коду та забезпечує повторне використання компонентів. Це робить Python привабливим для великих та складних проектів.

Розробники Python активно співпрацюють між собою, сприяючи створенню великої кількості відкритого програмного забезпечення. Це включає в себе бібліотеки, фреймворки та інші корисні інструменти, які допомагають в роботі над різними завданнями.

Python має велику та активну спільноту розробників, що дозволяє швидко отримати допомогу та розв'язати проблеми. Форуми, блоги та конференції регулярно обговорюють нові можливості та кращі практики.

Однією з ключових особливостей Python є його переносимість. Код, написаний на Python, може працювати на різних операційних системах без великих модифікацій.

Python постійно розвивається. Випуски нових версій регулярно додають нові функції та поліпшення, що дозволяє мові залишатися актуальною та конкурентоспроможною.

Python має вбудований фреймворк для написання та виконання тестів, який називається `unittest`. Цей фреймворк дозволяє розробникам створювати юніт-тести для перевірки окремих частин коду чи програмних компонентів.

`unittest` спрощує написання тестів за допомогою класів та методів. Розробники можуть описати очікувані результати та умови для своїх функцій чи класів, а потім викликати тести для автоматичного перевірки. Ось приклад структури юніт-теста на Python.

Лістинг 2.1 — Юніт-тест на Python

```
import unittest
class MyTestCase(unittest.TestCase):
    def setUp(self):
        # Підготовка до виконання тестів
    def test_addition(self):
        result = 1 + 2
        self.assertEqual(result, 3)
    def test_subtraction(self):
        result = 5 - 2
        self.assertEqual(result, 3)
    def tearDown(self):
        # Завершення роботи після виконання тестів
if __name__ == '__main__':
    unittest.main()
```

У цьому прикладі є два тести: `test_addition` та `test_subtraction`. Кожен тест порівнює результат виконання математичної операції з очікуваним значенням за допомогою методу `assertEqual`.

`setUp` та `tearDown` - це методи, які викликаються перед початком тестування та після його завершення відповідно. Вони використовуються для налаштування та очищення ресурсів, які можуть бути необхідні для тестування.

Ще однією популярною бібліотекою для юніт-тестів у Python є `pytest`. Вона пропонує простий та зручний синтаксис, що робить процес написання тестів більш приємним і ефективним. Приклад тесту на `pytest`:

Лістинг 2.2 — Приклад тесту на `pytest`

```
    result = 1 + 2
    assert result == 3
def test_subtraction():
    result = 5 - 2
```

```
assert result == 3
```

Обидві бібліотеки дозволяють легко і ефективно виконувати тести, допомагаючи розробникам переконатися, що їх код працює так, як очікується, і вчасно виявляти помилки чи неполадки.

Node.js — це середовище виконання JavaScript, побудоване на двигуні V8, який використовується у браузері Google Chrome. Однак вона застосовується на серверному боці для розробки ефективних та масштабованих веб- додатків. Це революційне рішення в світі веб-розробки, яке дозволяє використовувати JavaScript для створення серверних застосунків.

Однією з ключових особливостей Node.js є його асинхронний та подійно-орієнтований підхід. Замість блокування виконання коду через чекання завершення операцій введення/виведення, він використовує зворотні виклики та обробники подій, що дозволяє обробляти багато операцій паралельно, підвищуючи продуктивність додатків [19].

Node.js володіє широкою підтримкою громади розробників та активно розвивається. Велика кількість пакетів і модулів, доступних через пакетний менеджер npm, полегшує створення та розширення функціональності серверних додатків. Це середовище виконання дозволяє розробникам використовувати JavaScript для побудови ефективних мережевих застосунків, включаючи веб-сервери та мікросервіси. Node.js стає популярним вибором для створення застосунків реального часу, які потребують швидкої відповіді на події [19].

Для налаштування Node.js, спочатку необхідно завантажити його з офіційного сайту та встановити на ваш комп'ютер. Інсталятор автоматично додасть Node.js та npm (пакетний менеджер для JavaScript) до системи. Після встановлення, ви можете відкрити командний рядок або термінал та перевірити версії за допомогою команд `node-v` та `npm-v`.

Після успішної інсталяції ви можете створити свій перший проект, створивши новий каталог та використовуючи команду `npm init` для

налаштування проекту. Це ініціює процес створення файлу `package.json`, який міститиме інформацію про ваш проект та його залежності [20].

Після цього ви можете встановити необхідні модулі для вашого проекту, використовуючи команду `npm install`. Наприклад, якщо вам потрібен веб-фреймворк Express.js, ви можете встановити його за допомогою `npm install express`. Модулі будуть встановлені локально у вашому проекті, а їхні залежності будуть зазначені в файлі `package.json`.

Далі, ви можете створити свій основний файл програми, наприклад `index.js`, та почати писати свій код. При цьому вам слід вказати точку входу у вашому файлі `package.json`, щоб Node.js знаходив ваш основний файл при запуску [20].

Для запуску додатка використовуйте команду `node index.js`. Якщо ви бажаєте змусити сервер перезавантажуватися автоматично при змінах у коді, ви можете встановити пакет `nodemon` (`npm install nodemon`) та запускати ваш сервер командою `nodemon index.js`.

Також, ви можете використовувати різні змінні середовища, такі як `NODE_ENV`, для налаштування режиму розробки або виробництва. Це дозволить вам встановлювати різні параметри залежно від умов.

Потрібно також зберегти зміни та використовувати системи контролю версій, такі як Git, для ведення історії змін у проекті [20].

MySQL є однією з найпопулярніших відкритих реляційних систем управління базами даних (СУБД). Вона розроблена для забезпечення ефективного та надійного зберігання даних у великих та маленьких проектах. Ось короткий огляд основних характеристик та особливостей MySQL.

MySQL використовує мову запитів SQL (Structured Query Language) для взаємодії з базою даних. Вона підтримує стандартні операції, такі як SELECT, INSERT, UPDATE, DELETE, що дозволяє ефективно опрацьовувати дані у базі.

Однією з головних переваг MySQL є його відкритий код та безкоштовність використання. Це робить його доступним для широкого кола розробників та підприємств, що використовують відкриті технології [20].

MySQL підтримує транзакції, що робить його придатним для використання в транзакційних системах, де важливо забезпечити консистентність та надійність даних. Багатокористувацька та багатопроцесорна підтримка MySQL дозволяє обслуговувати багато запитів одночасно та ефективно масштабуватися під велику кількість користувачів.

MySQL також має вбудовані засоби для резервного копіювання та відновлення даних, що дозволяє забезпечити безпеку та захист від втрати інформації. Однією з особливостей MySQL є його розширюваність. Він може використовувати різні сховища для зберігання даних, такі як InnoDB, MyISAM, та інші, залежно від конкретних потреб проекту [20].

Інтерфейс командного рядка та графічний інтерфейс (наприклад, MySQL Workbench) надають розробникам зручний спосіб взаємодії з базою даних та виконання адміністративних завдань (рисунок 2.5).

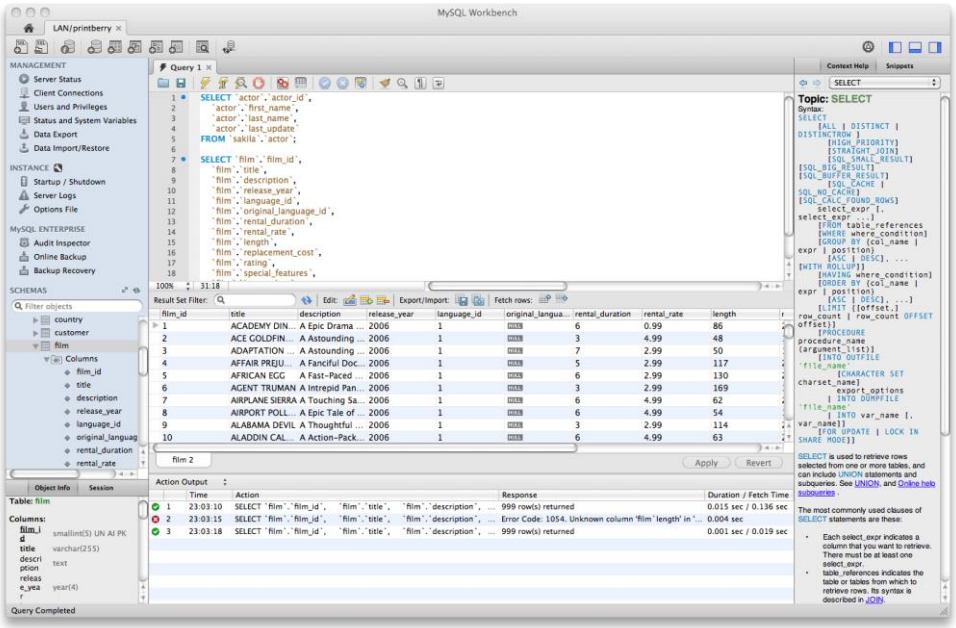

Рисунок 2.5 — Зовнішній вигляд MySQL Workbench

Крім того, MySQL є крос-платформеним рішенням, що дозволяє йому працювати на різних операційних системах, від Windows до Linux та macOS. Ця універсальність робить його легко інтегрованим у різні середовища.

Висока продуктивність є ще однією важливою рисою MySQL. Вона дозволяє ефективно обробляти велику кількість запитів та взаємодіяти з базами даних, надаючи високу швидкодію.

Надійність та стабільність є ключовими аспектами MySQL, що роблять його популярним в корпоративних середовищах. Система володіє високим рівнем безпеки та забезпечує ефективне управління даними.

MySQL активно підтримується великою спільнотою розробників та виробником Oracle Corporation. Це гарантує постійний розвиток та підтримку продукту, включаючи виправлення помилок та випуск нових версій.

Загалом, MySQL залишається надійним, потужним та гнучким інструментом для управління базами даних, що відповідає потребам різноманітних проектів та додатків.

3 РОЗРОБКА ВЕБ-ДОДАТКУ ДЛЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ КОНТРОЛЮ ЯКОСТІ ВИРОБНИЦТВА

3.1 Основні етапи розробки веб-додатку

Розробка веб-додатку — це комплексний процес, що включає в себе кілька ключових етапів. Починаючи з аналізу вимог, команда розробників ретельно вивчає потреби користувачів та бізнес-вимоги, визначаючи функціональні та технічні вимоги до додатку. Цей етап визначає основний напрямок роботи та об'єм функцій, які повинні бути реалізовані.

Після аналізу переходиться до етапу проектування, де встановлюється структура бази даних та архітектура додатку. Розробники створюють макети інтерфейсу, забезпечуючи відповідність дизайну та користувацького досвіду визначеним вимогам. Важливо на цьому етапі вибрати оптимальні технології та інструменти для подальшої розробки [21].

Слідуючим етапом є фаза розробки, де програмісти переходять до написання коду, реалізуючи функціональність та забезпечуючи взаємодію між різними компонентами системи. Цей етап вимагає уважності до деталей та відповідності створеного програмного коду встановленим стандартам та вимогам.

Після завершення розробки наступає етап тестування, на якому проводяться різновиди тестів — від модульних до системних, щоб виявити та виправити всі можливі помилки та недоліки. Це необхідний етап для забезпечення надійності та стабільності додатку [22].

Після успішного тестування додаток готовий до впровадження. Розгортання може відбуватися на власному сервері або у хмарному середовищі в залежності від обраної стратегії. Підготовка до впровадження включає в себе налаштування середовища та бази даних.

Останнім етапом є супровід та підтримка, що включає в себе постійний моніторинг та оптимізацію продуктивності. Крім того, розробники готові вносити корективи та вдосконалення в залежності від зворотного зв'язку користувачів та ринкових змін [22].

При авторизації або реєстрації на сайті, користувач ініціює взаємодію із системою, починаючи зі введення своїх особистих даних. Система перевіряє ці дані на валідність та наявність у базі даних. У випадку реєстрації, якщо користувача з такими даними ще не існує, йому надається можливість створити обліковий запис. Якщо користувач вже зареєстрований, він має можливість авторизуватися, вводячи свої облікові дані [23].

Далі, система перевіряє правильність введених даних та забезпечує доступ користувачу до відповідного облікового запису. У випадку невірного введення даних, система повідомляє користувача про помилку та можливість її виправлення. Блок-схема загального алгоритму роботи додатку зображена на рисунку 3.1.

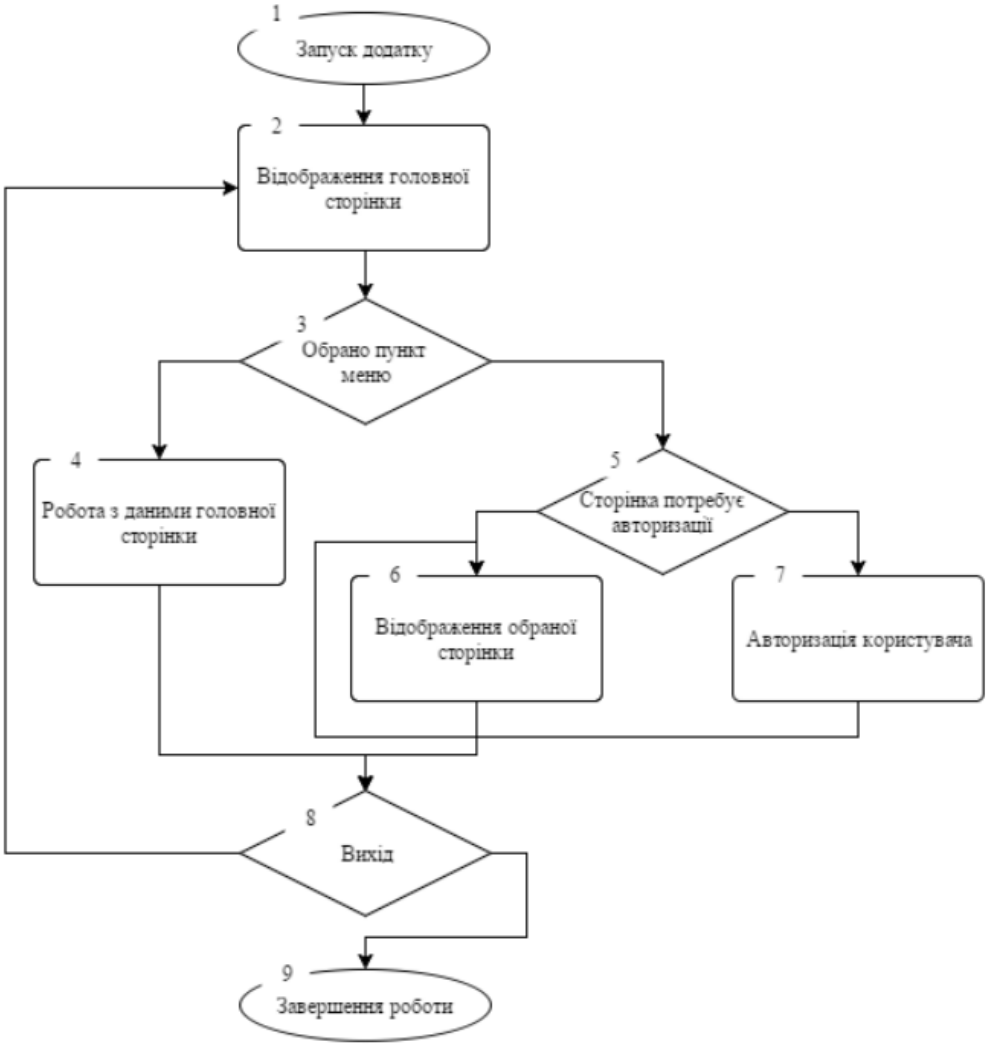

Рисунок 3.1 — Блок-схема загального алгоритму роботи додатку

Після успішної авторизації чи реєстрації, користувач має доступ до свого облікового кабінету, де він може виконувати різні дії в залежності від функціоналу сайту. Авторизація дозволяє системі ідентифікувати користувача та надавати йому персоналізований досвід взаємодії з ресурсом.

Таким чином, процес авторизації чи реєстрації є ключовим елементом взаємодії користувача з веб-сайтом, забезпечуючи безпечний та персоналізований доступ до функціоналу системи.

При створенні системи з розділеною авторизацією для користувачів та адміністраторів, процес авторизації стає більш диференційованим.

Користувач, який реєструється чи авторизується, вказує свої особисті дані, такі як ім'я користувача та пароль. Система перевіряє ці дані, і якщо вони вірні, дозволяє користувачеві увійти в особистий кабінет.

Адміністратор, з іншого боку, використовує свої унікальні дані для входу в систему, які можуть бути, наприклад, адміністративний логін та пароль. Ці дані перевіряються системою, і в разі успішної авторизації адміністратор отримує доступ до панелі адміністрування.

Такий підхід дозволяє відокремити функціонал для звичайних користувачів та адміністраторів, надаючи кожній групі відповідні можливості та права доступу. Користувачі отримують доступ до особистого кабінету та основного функціоналу, тоді як адміністратори можуть використовувати додаткові інструменти та контролювати різні аспекти системи через панель адміністрування.

Це забезпечує ефективний та безпечний доступ до системи для обох користувачів, забезпечуючи при цьому відмінену адміністративну контроль та безпеку. Блок— схема алгоритму роботи додатку з розділеною авторизацією наведена у додатку Б.

3.2 Розробка алгоритму взаємодії IoT пристроїв з веб-додатком

Алгоритм автоматизації контролю якості виробництва є сучасним та інноваційним підходом до забезпечення високої якості продукції в

промисловості. В основі цього алгоритму лежать принципи теорії автоматизації та інтеграції передових веб-технологій.

Сучасні підходи до контролю якості використовують технології IoT та сенсорів для збору даних в режимі реального часу. Теорія керування якістю і підходи до статистичного аналізу використовуються для розробки ефективних стратегій виявлення та управління відхиленнями.

Використання алгоритмів машинного навчання для аналізу даних та виявлення аномалій базується на теорії великих даних та статистичних моделей. Це дозволяє системі вчитися та адаптуватися до змін у виробничому середовищі.

Застосування веб-технологій для створення інтуїтивного інтерфейсу базується на принципах веб-розробки та дизайну, що забезпечує зручну взаємодію з користувачем.

Розробимо алгоритм взаємодії технонологій IoT і веб-технологій.

Система контролю якості розпочинає свою роботу з використання передових технологій сенсорів та IoT пристроїв, які знаходяться на різних етапах виробництва. Ці сенсори вимірюють різноманітні параметри, такі як температура, вологість, тиск, а також інші характеристики, ключові для забезпечення якості продукції.

Отримані дані автоматично передаються до центральної системи через шифрований канал зв'язку. Це гарантує не лише швидкість передачі, але й конфіденційність інформації, оскільки якість виробництва є критичною для конкурентоспроможності підприємства.

Система використовує потужні алгоритми машинного навчання для глибокого аналізу надходячих даних. На основі цього аналізу визначаються стандартні параметри та робочі границі для кожного параметра, і система автоматично встановлює параметри контролю.

Виявлені аномалії в роботі виробництва чітко визначаються, і система надає детальні аналітичні дані, що дозволяє оперативно реагувати на будь-які неполадки та миттєво коригувати процес.

Виявлені аномалії призводять до автоматичного висилання сповіщень відповідальним особам через різні канали зв'язку — електронну пошту, месенджери чи SMS. Це не лише забезпечує вчасну реакцію, але й дозволяє оперативно створювати групи реагування для кожного конкретного випадку.

Реакція на виявлені аномалії може бути автоматичною, через вбудовані механізми керування, або вручну, в залежності від складності ситуації та вимог замовника.

Розроблений веб-додаток надає користувачам доступ до реального часу зібраних даних. Панель керування включає в себе графіки, діаграми, показники ефективності та стану виробництва. Це створює чіткий та зрозумілий звіт, який полегшує прийняття рішень.

Зокрема, інтуїтивний дизайн інтерфейсу веб-додатка робить користування системою максимально простим та зручним. Користувачі можуть легко переглядати дані, порівнювати різні показники, та приймати оперативні рішення для покращення якості виробництва.

Всі оброблені дані зберігаються в центральній базі даних для подальшого аналізу та створення звітів. Аналіз тенденцій забезпечує можливість планування стратегій контролю якості та визначення напрямків подальшого удосконалення виробництва.

Система не лише виявляє відхилення, але й пропонує оптимізаційні рішення для поліпшення процесів виробництва. Зокрема, застосування алгоритмів машинного навчання дозволяє системі вчитися з часом та пристосовуватися до змін виробничого середовища.

Система також надає можливість встановлення взаємозв'язків між різними показниками, що сприяє створенню синергії між виробничими процесами та можливостями контролю якості. Це дозволяє виробнику не лише реагувати на поточні проблеми, а й стратегічно планувати подальший розвиток. Система також включає функції моніторингу та діагностики обладнання. Вона може виявляти потенційні проблеми зі справністю обладнання та висилати сповіщення про необхідність технічного обслуговування. Це не лише

забезпечує продуктивність виробництва, але і передбачає можливі збої, зменшуючи час простою та витрати на ремонт.

Така система дозволяє проводити аналіз витрат на виробництво та оптимізацію ресурсів. Це дозволяє ефективно використовувати матеріали, енергію та інші ресурси, що в результаті призводить до зменшення витрат та підвищення прибутковості виробництва.

Цей комплексний підхід до автоматизації контролю якості виробництва створює ефективну систему, яка не лише виявляє проблеми, але й активно сприяє вдосконаленню виробничого процесу та підвищенню конкурентоспроможності підприємства. Блок-схема алгоритму взаємодії наведена у додатку В.

3.2 Програмна реалізація веб-додатку

При розробці веб-додатку було вирішено використовувати стек технологій, що базується на клієнт-серверній архітектурі. На стороні клієнта використовуються HTML, CSS та JavaScript для створення інтерфейсу, а на стороні сервера використовується мова програмування Node.js та Python для обробки запитів та управління базою даних.

Головна сторінка веб-додатку містить загальну інформація для користувача та навігаційне меню що зображено на рисунку 3.2 та в яке входить:

— вкладка з авторизацію чи реєстрацією;
— панель управління;
— виявлення дефектів;
— стандарти якості;
— звіти про якість;
— здійснення налаштування.

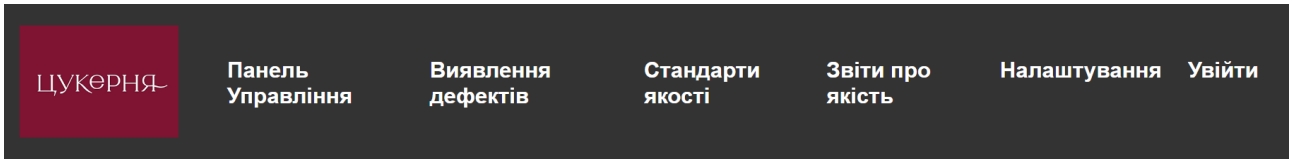

Рисунок 3.2 — Навігаційне меню

Використовуючи AJAX (Asynchronous JavaScript and XML), здійснюється асинхронний обмін даними між клієнтом та сервером, що дозволяє оновлювати інформацію без перезавантаження сторінки. Для цього використаємо наступний код, який написано у лістингу 3.1.

Лістинг 3.1 — Асинхронний обмін даними

```
function fetchData() {
  const xhr = new XMLHttpRequest();
  xhr.onreadystatechange = function() {
    if (xhr.readyState === XMLHttpRequest.DONE) {
      if (xhr.status === 200) {
        const data = JSON.parse(xhr.responseText);
      } else {
        console.error('Помилка отримання даних з сервера'); }  }  };
  xhr.open('GET', '/api/data', true);
  xhr.send(); }
```

На сторінці звіти про якість продукції додамо завантажимо декілька документів, що відповідають за звіти упродовж року на підприємстві. Необхідно додати кнопку, аби користувач міг завантажити дані файли собі на комп’ютер. Реалізуємо функцію для виведення звітів.

Лістинг 3.2 — Функція для виведення звітів

```
function displayReports() {
    const reportsContainer = document.getElementById('qualityReports');
    reportsContainer.innerHTML = '';
    reports.forEach((report, index) => {
      const reportElement = document.createElement('div');
      reportElement.classList.add('qualityReport');
      reportElement.innerHTML =
`<strong>${report.title}</strong><br>${report.content}`;
      const downloadButton = document.createElement('button');
```

```
downloadButton.classList.add('downloadButton');
downloadButton.innerHTML = 'Завантажити документ';
downloadButton.onclick = () => downloadReport(index);
reportElement.appendChild(downloadButton);
reportsContainer.appendChild(reportElement);    });   }
```

Ця функція відповідає за виведення звітів на веб- сторінці. Коли вона викликається, вона очищує контейнер для звітів на сторінці (елемент з ідентифікатором 'qualityReports'), а потім для кожного звіту з масиву `reports` створює новий HTML елемент `<div>` з відповідним класом 'qualityReport'. У цьому елементі виводиться заголовок та вміст звіту.

Для кожного звіту також додається кнопка 'Завантажити документ', яка викликає функцію `downloadReport(index)`, де `index` — це індекс поточного звіту у масиві. При натисканні на цю кнопку буде викликана функція `downloadReport(index)`, яка в свою чергу розпочне процес завантаження відповідного текстового файлу звіту.

Усі створені елементи додаються до контейнера для звітів на сторінці. Таким чином, функція `displayReports()` виводить на сторінці інформацію про всі доступні звіти та надає можливість завантажити кожен з них у вигляді текстового файлу. Приклад сторінки зі звітами зображено на рисунку 3.3

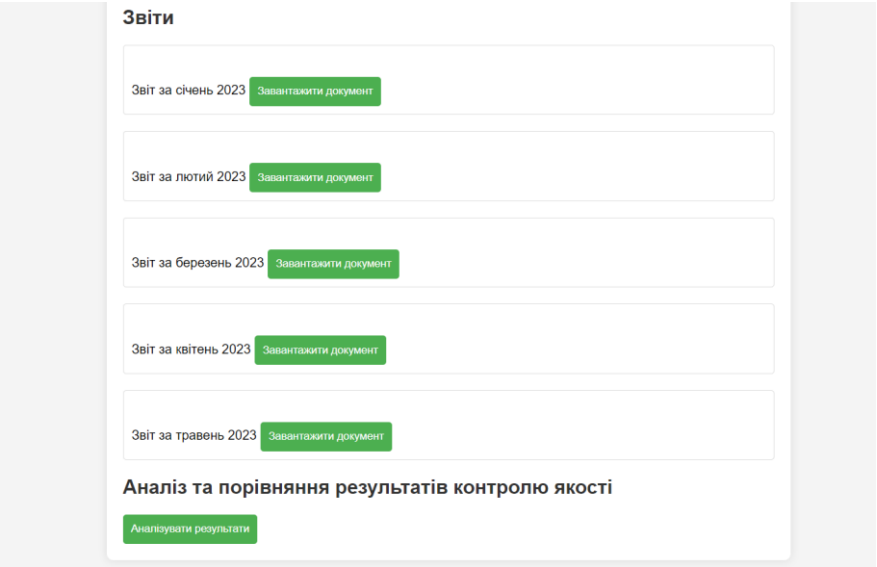

Рисунок 3.3 — Сторінка із згенерованими звітами

Напишемо функцію для завантаження документу.

Лістинг 3.3 — Функція для завантаження документу

```
function downloadReport(index) {
    const report = reports[index];
    const blob = new Blob([`${report.title}\n\n${report.content}`], { type: 'text/plain' });
    const url = URL.createObjectURL(blob);
    const a = document.createElement('a');
    a.href = url;
    a.download = `${report.title.replace(/\s+/g, '_')}.txt`;
    document.body.appendChild(a);
    a.click();
    document.body.removeChild(a);
    URL.revokeObjectURL(url);
  }
```

Ця функція створює та завантажує текстовий файл, представляючи конкретний звіт. Вона бере інформацію з об'єкту звіту, створює текстовий Blob, створює URL для нього, імітує клік на створеному посиланні для завантаження файлу, а потім очищує створені DOM— елементи та звільняє ресурси. У результаті користувач може легко завантажити текстовий файл з вмістом звіту. Після завантаження користувач може переглянути файл.

Тепер перейдемо до розробки сторінки із відстеження дефектів на виробництві. Вона має містити поля із статусом та назвою дефекта, кнопку для зберігання та відображення його у таблиці для звітів.

Напишемо код для даної форми. На рисунку 3.4 зображено вже готовий результат, який було отримано після написання даного коду.

Лістинг 3.4 — Код для відображення дефектів

```
<form id="defectForm">
    <label for="defectName">Назва дефекту:</label>
```

```
    <input type="text" id="defectName" name="defectName" required>
    <label for="defectStatus">Статус дефекту:</label>
    <select id="defectStatus" name="defectStatus" required>
      <option value="новий">Новий</option>
      <option value="в роботі">В роботі</option>
      <option value="вирішений">Вирішений</option>
    </select>
    <button type="submit">Зберегти дефект</button>
</form>
```

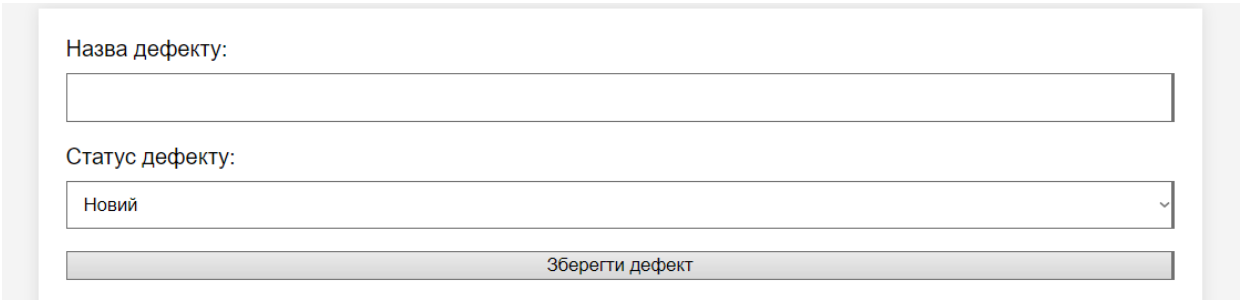

Рисунок 3.4 — Приклад сторінки із відстеження дефектів

На сторінці із панеллю управління необхідно додати графік, у якому буде зазначатись кількість відсотків товарік із виявленими дефектами та інше. Для створення графіка можна використовувати бібліотеки для візуалізації даних, наприклад, Chart.js. Ось функція на JavaScript для створення графіка на сторінці з панеллю управління.

Лістинг 3.5 — Функція для створення графіка

```
const defectData = [20, 15, 30, 25];
function createDefectChart() {
    // Отримати контекст canvas, на якому буде малюватися графік
    const ctx = document.getElementById('defectChart').getContext('2d');
    // Створити графік, використовуючи Chart.js
    const defectChart = new Chart(ctx, {
        type: 'bar',
```

```
        data: {
            labels: ['Січень', 'Лютий', 'Березень', 'Квітень'], // Замініть це на ваші місяці або періоди
            datasets: [{
                label: 'Відсоток товарів із дефектами',
                data: defectData,
                backgroundColor: 'rgba(255, 99, 132, 0.2)',
                borderColor: 'rgba(255, 99, 132, 1)',
                borderWidth: 1
            }]
        },
        options: {
            scales: {
                y: {
                    beginAtZero: true,
                    max: 100
                } }
    }); }
window.onload = createDefectChart;
```

У функції createDefectChart, ви побачите, як створюється новий екземпляр графіка, встановлюються дані для відображення та надаються опції конфігурації. Функція призначена для створення вертикального стовпчастого графіка, який відображає відсоток товарів із дефектами протягом заданого періоду. Щоб використовувати Chart.js, слід включити бібліотеку на сторінці, наприклад, вставивши посилання на неї у ваший HTML— код.

У сторінці з налаштуванням необхідно розробити базові налаштування для користувача. Наприклад основні елементи на цій сторінці включають в себе:

— вибір рівня доступу: користувач може обрати рівень доступу, такий як "повний доступ", "обмежений доступ" чи "тільки перегляд".

— вибір налаштувань сповіщень: користувач може вибрати, чи хоче він отримувати сповіщення.

— вибір кольору сповіщень: користувач може обрати кольорову схему для відображення сповіщень.

— вибір мови інтерфейсу: користувач може вибрати мову для відображення інтерфейсу додатка чи веб- сайту.

Функціонал сторінки дозволяє зберігати обрані налаштування та виводити повідомлення про успіх або помилку в залежності від результату збереження.

Детальний опис коду сторінок наведено у додатках Г та Д.

Тепер перейдемо до розробки серверної частини додатку. Використаємо Node.js.

Створимо новий каталог для проекту та введемо команду в терміналі (або командному рядку): mkdir adminPanelApp cd adminPanelApp npm init –y. Встановимо Express та інші необхідні залежності: npm install express body-parser. Після цього необхідно створити файл `server.js` і додати наступний код:

Лістинг 3.6 — Підключення до серверу

```
const express = require('express');
const bodyParser = require('body— parser');
const mysql = require('mysql2');
const app = express();
const port = 3000;
const db = mysql.createConnection({
  host: 'localhost',
  user: 'your_mysql_username',
  password: 'your_mysql_password',
  database: 'adminPanelDB',
});
db.query(`
  CREATE TABLE IF NOT EXISTS settings (
```

```
    id INT AUTO_INCREMENT PRIMARY KEY,
    userManagement VARCHAR(255),
    notificationSettings BOOLEAN,
    notificationColor VARCHAR(255),
    language VARCHAR(255)
  )
`, (err) => {
  if (err) {
    console.error('Помилка при створенні таблиці:', err.message);
  } else {
    console.log('Таблиця створена або вже існує');
  }
});
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));
app.post('/api/settings', (req, res) => {
  const newSettings = req.body;
  db.query('INSERT INTO settings SET ?', newSettings, (err, result) => {
    if (err) {
      console.error('Помилка при збереженні налаштувань:', err.message);
      res.status(500).json({ error: 'Виникла помилка.' });
    } else {
      console.log('Нові налаштування:', newSettings);
      res.json({ message: 'Налаштування збережено!' });
    }
  });
});
app.get('/', (req, res) => {
    res.send({ message: 'Hello WWW!' }); });
app.listen(port, () => {
```

```
  console.log(`Сервер запущено на порті ${port}`);
});
```

Запустимо сервер, використовуючи команду: node server.js. Тепер, коли сервер запущено, можна звернутися до нього у браузері. Перейдемо за адресою http://localhost: 3000/.

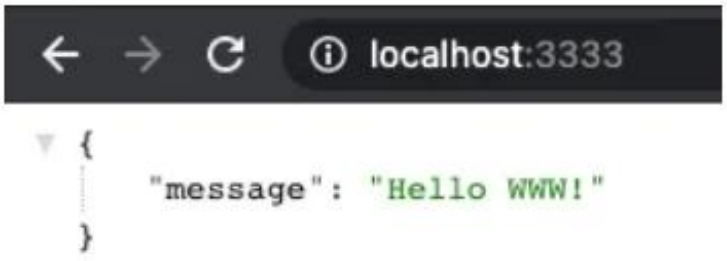

Рисунок 3.5 — Налаштування серверу

Отож, налаштування пройшло успішно. Тепер перейдемо до розробки бази даних.

На основі аналізу предметної галузі формується перелік програмних сутностей та атрибутів сутностей, представлених класами та їх учасниками (таблиця 3.1). Ця таблиця містить 4 сутності, а саме : товари, дефекти, звіти про якість та користувачі.

Таблиця 3.1 — Перелік сутностей та їх атрибутів

Сутність	Атрибути
Товари	product_id (PK), product_name, description, quality_manager_id (FK to Users)
Дефекти	defect_id (PK), defect_name, description, responsible_user_id (FK to Users)
Звіти про якість	report_id (PK), product_id (FK to Products), defect_id (FK to Defects), user_id (FK to Users), report_date, comments
Користувачі	user_id (PK), username, password, role

Перейдемо до розробки кожної з таблиць. Сутність товари містить чотири поля та один зв’язок з сутністю користувачі : унікальний ідентифікатор товару, назва товару, опис товару, зовнішній ключ, що посилається на ідентифікатор користувача (users), який відповідає за контроль якості для цього товару.

Напишемо код для її створення.

Лістинг 3.7 — Таблиця з продукцією

```
CREATE TABLE IF NOT EXISTS Products (
  product_id INT AUTO_INCREMENT PRIMARY KEY,
  product_name VARCHAR(255),
  description TEXT,
  quality_manager_id INT,
  FOREIGN KEY (quality_manager_id) REFERENCES Users(user_id)
);
```

Перейдемо до розробки наступної таблиці. Сутність дефекти містить унікальний ідентифікатор дефекту, назва дефекту, опис дефекту, зовнішній ключ, що посилається на ідентифікатор користувача (users), який відповідає за виявлення та вирішення цього дефекту. Ось код для її створення.

Лістинг 3.8 — Таблиця з дефектами

```
CREATE TABLE IF NOT EXISTS Defects (
  defect_id INT AUTO_INCREMENT PRIMARY KEY,
  defect_name VARCHAR(255),
  description TEXT,
 responsible_user_id INT,
 FOREIGN KEY (responsible_user_id) REFERENCES Users(user_id)
);
```

Таблиця звітів про якість містить :

— унікальний ідентифікатор звіту;

— зовнішній ключ, що посилається на ідентифікатор товару (products), до якого відноситься звіт;

— зовнішній ключ, що посилається на ідентифікатор дефекту (defects), виявленого в звіті;

— зовнішній ключ, що посилається на ідентифікатор користувача (users), який створив звіт;

— дата створення звіту;

— коментарі та додаткова інформація до звіту.

Таблиця "Звіти про якість" містить інформацію про результати контролю якості продукції, включаючи зв'язки з конкретним товаром, виявленими дефектами та користувачем, який створив звіт. Це дозволяє системі відстежувати та аналізувати якість продукції та виявлені недоліки для подальшого вдосконалення процесів виробництва.

Таблиця з користувачами має такі поля як Унікальний ідентифікатор користувача, Ім'я користувача, Хеш пароля користувача, Роль користувача (адміністратор, оператор контролю якості). Лістинг по створенню таблиць наявний у додатку Е.

Тепер необхідно забезпечити безпеку у нашій системі, слід вжити ряд заходів:

— використовувати надійний алгоритм хешування (bcrypt);

— зберігати тільки хеші паролів у базі даних, а не самі паролі;

— встановити адекватну вартість (cost) для алгоритму хешування для затратності обчислення хешу.

Лістинг 3.9 — Використання бібліотеки bcrypt у Node.js:

```
const bcrypt = require('bcrypt');
const saltRounds = 10; // Кількість раундів для генерації солі
// Функція для хешування пароля
async function hashPassword(password) {
    try {
        const salt = await bcrypt.genSalt(saltRounds);
```

```
        const hashedPassword = await bcrypt.hash(password, salt);
        return hashedPassword;
    } catch (error) {
        throw new Error('Error hashing password');
    }
}
// Функція для перевірки пароля при вході
async function comparePassword(inputPassword, hashedPassword) {
    try {
        const match = await bcrypt.compare(inputPassword, hashedPassword);
        return match;
    } catch (error) {
        throw new Error('Error comparing passwords');
    }
}
// Приклад використання:
const plainPassword = 'mySecurePassword';
// Хешуємо пароль
hashPassword(plainPassword)
    .then(hashedPassword => {
        console.log('Hashed Password:', hashedPassword);
        // Перевіряємо пароль при вході
        comparePassword('incorrectPassword', hashedPassword)
            .then(match => {
                console.log('Password Match:', match);
            })
            .catch(error => {
                console.error(error.message);
            });
    })
```

```
.catch(error => {
    console.error(error.message);
});
```

Детальний лістинг для забезпечення безпеки наявний у додатку Ж. У цьому коді забезпечена можливість автентифікації користувача за допомогою пари ім'я користувача — пароль, а також генерація та перевірка JWT-токенів. Токени використовуються для авторизації користувачів та доступу до захищених маршрутів. Маршрут /protected є прикладом захищеного маршруту, доступного тільки для автентифікованих користувачів.

4 ТЕСТУВАННЯ ТА РЕКОМЕНДАЦІЇ КОРИСТУВАЧУ

4.1 Опис технологій тестування додатку

Веб-тестування — це тестування програмного забезпечення, яке зосереджується на веб- додатках. Комплексне тестування веб-системи до її впровадження може виявити та вирішити проблеми ще до того, як система стане доступною для громадськості. Ці проблеми охоплюють аспекти безпеки веб-додатку, основні функціональні можливості сайту, його доступність для користувачів з обмеженими та повними можливостями, здатність адаптуватися до різноманітних робочих столів, пристроїв і операційних систем. Крім того, важливими є готовність до очікуваного трафіку, здатність витримати значний приріст користувачів і відновлювальність після значного збільшення трафіку, обидва аспекти пов'язані з тестуванням навантаження.

Розглянемо різні аспекти технологій тестування веб- додатків, які важливі для забезпечення якості продукту.

Функціональне тестування спрямоване на перевірку того, чи виконуються очікувані функції веб-додатку. Наприклад, для інтернет-магазину це може включати тестування функціоналу додавання товарів у кошик, оформлення замовлення та оплату.

Наприклад, тестування опції "Додати до кошика" для переконливості, що товари дійсно додаються до кошика, і його вміст відображається коректно.

Відмовостійкість тестування спрямоване на перевірку того, як веб-додаток поводиться в умовах відмов або помилок. Це включає тестування відновлення додатку після втрати з'єднання або помилок сервера.

Наприклад тестування поведінки веб-додатку під час втрати з'єднання з Інтернетом та переконання, що користувач отримає адекватне повідомлення.

Тестування безпеки охоплює перевірку на вразливості та атаки на веб-додаток. Це може бути тестування на вибірковий доступ, перехоплення даних або SQL-ін'єкції. Наприклад, спроба введення шкідливих команд в поля вводу для перевірки, чи вдається веб-додатку відфільтрувати такі вразливості.

Тестування сумісності включає в себе перевірку того, як добре працює веб-додаток на різних браузерах та пристроях.

Тестування продуктивності визначає, як добре веб-додаток працює при великому навантаженні, забезпечуючи швидку реакцію та високу швидкість відгуку.

Тестування взаємодії користувача орієнтоване на перевірку зручності та ефективності взаємодії з веб-додатком. Це включає аналіз інтерфейсу користувача, перевірку логіки навігації та функціоналу. Наприклад, тестування взаємодії з різними типами користувачів, переконання, що інтерфейс і функції зрозумілі і зручні для всіх категорій користувачів.

І на кінець, автоматизоване тестування використовує інструменти та скрипти для автоматизації виконання тестів. Це дозволяє прискорити процес тестування та забезпечити більшу покриття. Наприклад це може бути написання автоматизованих тестів для перевірки основних функцій веб-додатку, які можна виконати автоматично при кожній зміні коду.

В процесі розробки веб-додатків велике значення приділяється тестуванню користувацького інтерфейсу (UI). Ця складова включає в себе ряд тестів, які оцінюють функціональність та взаємодію між користувачем та інтерфейсом. Мета полягає в переконанні, що веб-додаток працює правильно, а користувач може з легкістю взаємодіяти з його елементами.

Важливим аспектом веб-тестування є перевірка відповідності дизайну та функціональності вимогам. У цьому контексті проводяться різноманітні тести, які переконують, що користувач може легко пересуватися між різними сторінками, правильно вводити дані, натискати кнопки та посилання, а також взаємодіяти з різними частинами інтерфейсу.

Крім того, важливим етапом тестування є впевненість, що веб-додаток адаптивно відображається на різних пристроях. Також перевіряється валідація форм, зручність взаємодії та ефективність повідомлення користувачу про можливі помилки чи некоректні введення. Усе це спрямовано на забезпечення приємного та ефективного використання веб-додатку.

Розробка веб-додатку вимагає глибокого тестування його серверної частини для гарантії надійності та ефективності. Почнемо з написання юніт-тестів для конкретних функцій та методів серверу, щоб впевнитися в їхній правильній роботі в ізоляції.

Далі, інтеграційні тести дозволять перевірити взаємодію різних компонентів серверу під час обробки запитів та формування відповідей. У випадку веб-додатків з відкритим API, критично створити тести для перевірки функціональності API-маршрутів.

На наступному етапі слід орієнтуватися на тести продуктивності, щоб з'ясувати, як сервер веде себе при великому обсязі запитів. Також, варто включити тести безпеки для виявлення потенційних вразливостей перед можливими атаками.

Для автоматизації процесу тестування, розгляньте можливість створення автоматизованих тестів, які будуть запускатися при змінах у коді. Використовуйте фреймворки для тестування, такі як Mocha чи Jest, для ефективного виконання тестових сценаріїв.

Необхідно також враховувати, що тестування - це динамічний процес, і регулярне оновлення тестів, особливо при внесенні змін до коду, гарантує високу якість та надійність серверної частини вашого веб-додатку.

4.2 Тестування веб-додатку

Першим кроком відкриємо веб-додаток у браузері, перше що ми бачимо це сторінку, де потрібно обрати: зареєструватись чи авторизуватись. Після натискання відкриється форма для реєстрації чи авторизації (рисунок 4.1).

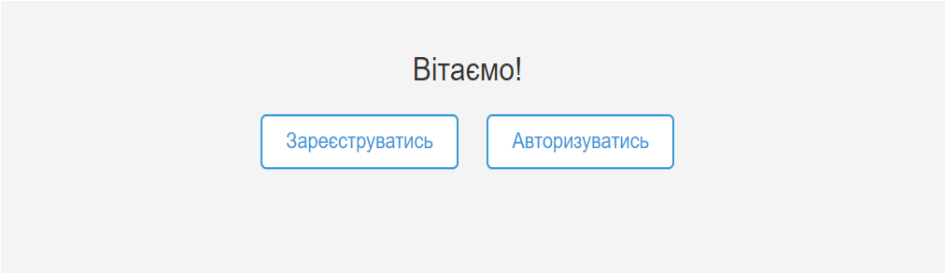

Рисунок 4.1 — Вхід у систему

Спочатку розглянемо процес реєстрації користувача. Якщо натиснути на кнопку зареєструватись відкриється форма із реєстрацією. Для того щоб зареєструватись користувач повинен заповнити всі поля (рисунок. 4.2). Після натискання на кнопку зареєструватись, з'явиться повідомлення що реєстрація пройшла успішно. Якщо ж хоча б одне поле не заповнене, з'явиться повідомлення, що слід заповнити усі обов'язкові поля.

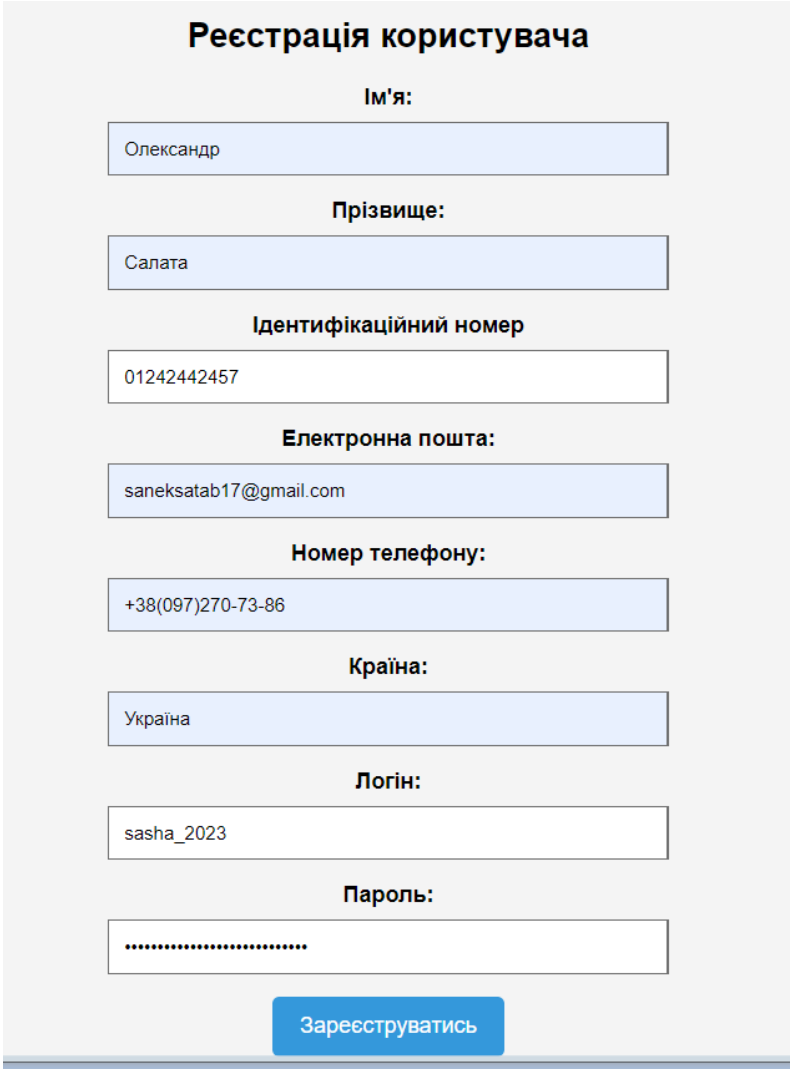

Рисунок 4.2 — Форма для реєстрації

Для того аби авторизуватись необхідно заповнити усі поля та вибрати роль, за якою ви хочете зайти (рисунок 4.3). Якщо ви є адміністратором, слід ввести ваш ідентифікаційний номер як працівника на виробництві. Якщо ви є простим користувачем просто пропустіть це поле. Сторінки веб-додатку будуть відрізнятись, в залежності за якою роллю ви увійшли.

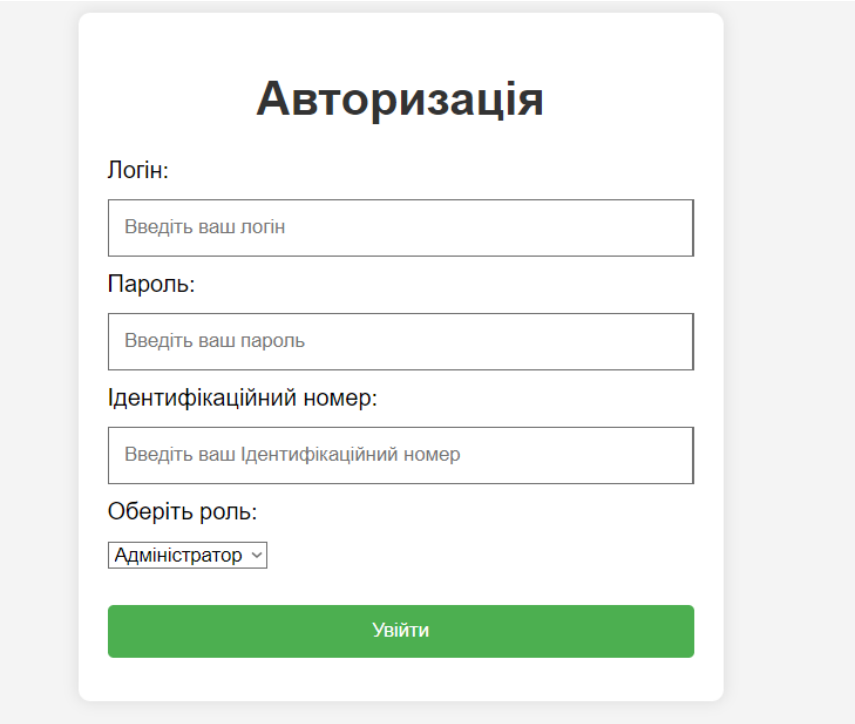

Рисунок 4.3 — Форма для авторизації

Після авторизації переходимо на головну сторінку додатку. На рисунку 4.4 наведено вигляд сторінка зі сторони адміністратора.

Рисунок 4.4 — Головна сторінка веб-додатку

Натиснувши у навігаційному меню на вкладку панель управління, завантажиться сторінка із наявним графіком роботи, який постійно змінюється в залежності від відсотка виявлених дефектів, які система буде передавати у базу даних (рисунок 4.5).

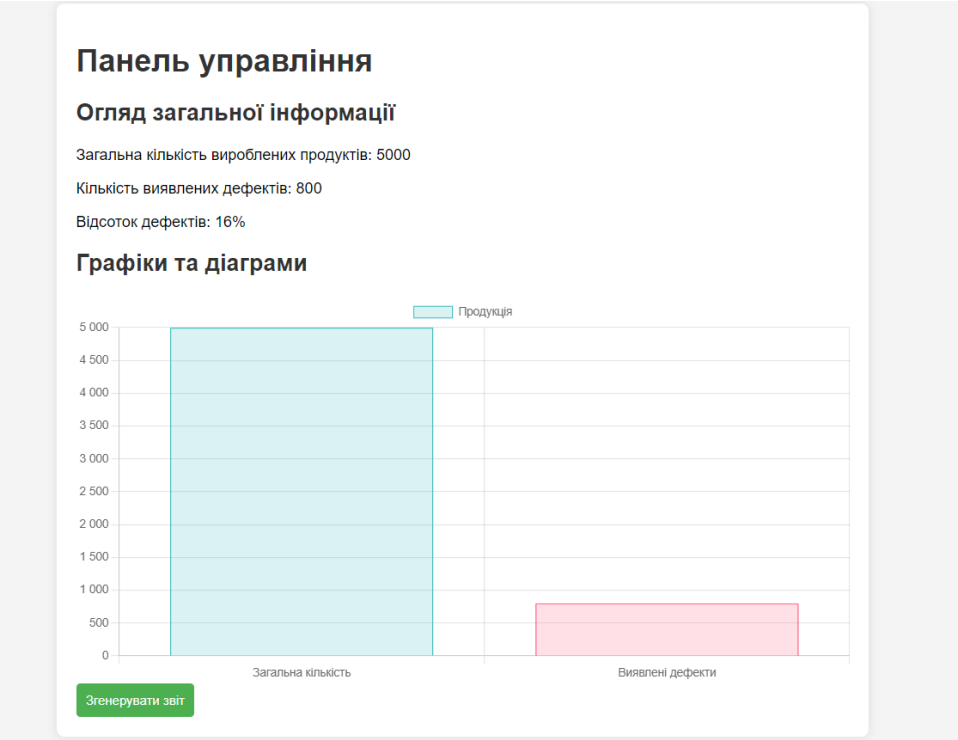

Рисунок 4.5 — Сторінка із панелю управління

Тепер перейдемо до сторінки із виявлення дефектів, для цього слід натиснути на кнопку у навігаційному меню. На цій сторінці адміністратор може ввести дефект та згенерувати звіт. Користувачеві буде видно лише наявні згенеровані звіти.

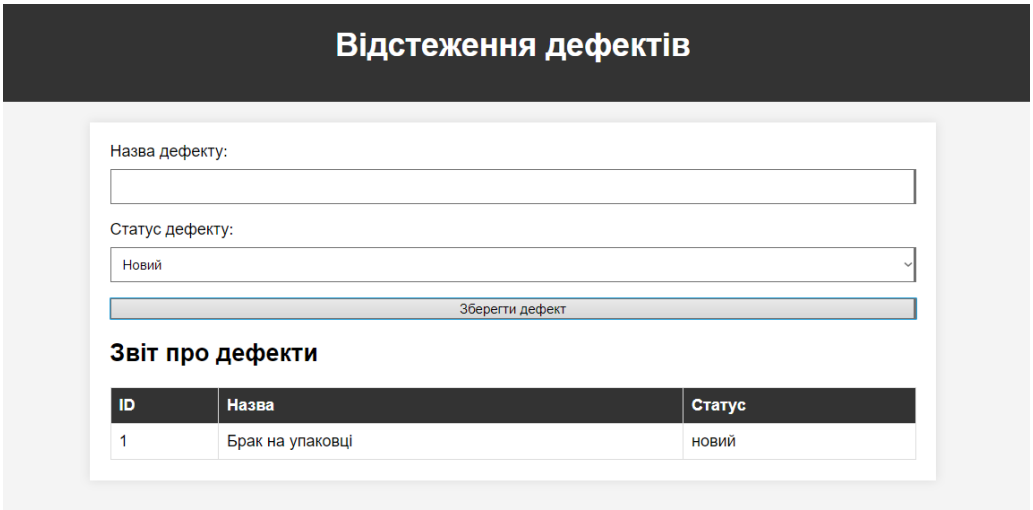

Рисунок 4.6 — Сторінка із відстеженням дефектів

Для того аби переглянути звіти про якість товару на виробництві, слід натиснути на кнопку звіти про якість. Тоді відкриється сторінка, де будуть усі

згенеровані звіти за кожен окремий місяць. Їх можна буде завантажити собі на комп’ютер (рисунок 4.7).

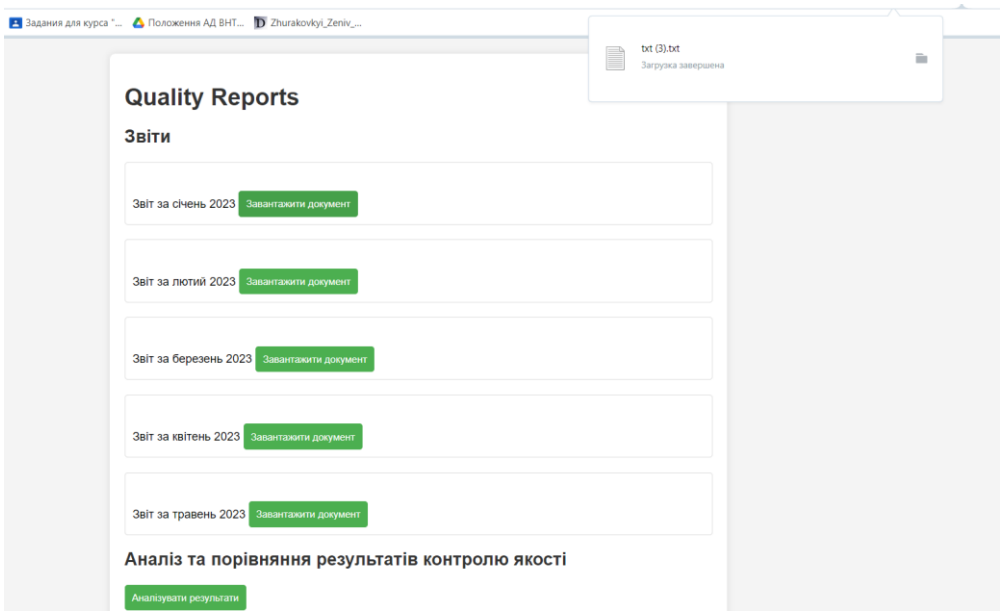

Рисунок 4.7 — Сторінка із звітами про якість продукції

У налаштуваннях адміністратор може обмежити на деякий час доступ користувачів до даних, може змінити мову веб-додатку та здійснити інші налаштування (рисунок 4.8).

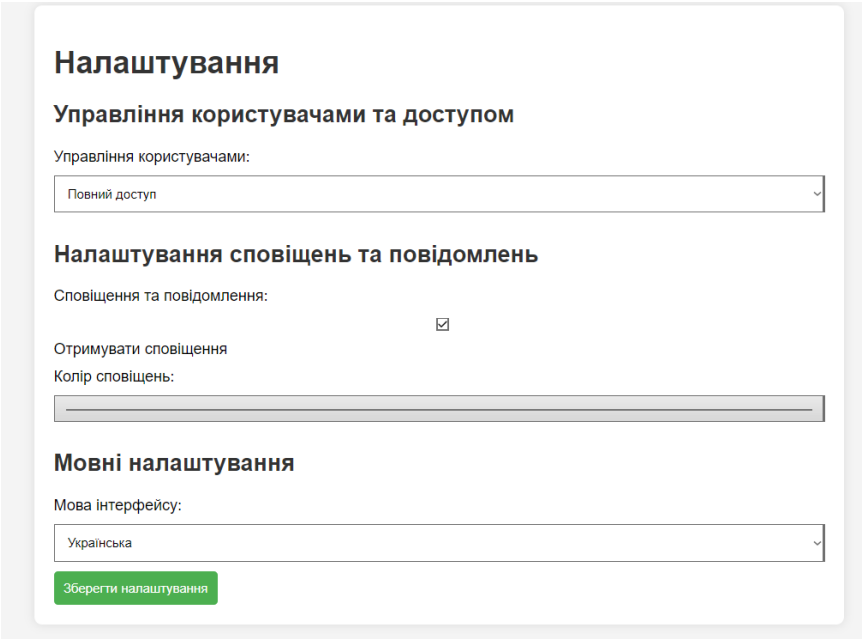

Рисунок 4.8 — Сторінка із налаштуванням

Аналізуючи функціонал серверної частини веб-додатку, розробимо юніт-тести для перевірки ключових аспектів її функціональності та надійності. Напишемо декілька юніт-тестів.

Лістинг 4.1 — Перевірка Аутентифікації Користувача

```
def test_user_authentication():
    # Створення тестового користувача
    user = User(username='testuser', password='testpassword')
    # Запит на сервер для аутентифікації
    response = app.test_client().post('/login', data=dict(
        username='testuser',
        password='testpassword'
    ))
    # Перевірка коду відповіді сервера
    assert response.status_code == 200
    # Перевірка, чи користувач входить в систему
    assert 'authenticated' in session
    assert session['authenticated']
    # Перевірка додаткових параметрів користувача
    assert current_user == user
```

У цьому тесті ми створюємо тестового користувача та передаємо дані для аутентифікації на сервер. Після обробки запиту перевіряємо, чи код відповіді від сервера - 200, що свідчить про успішну аутентифікацію. Далі ми перевіряємо, чи користувач вже аутентифікований та його параметри сесії. Цей тест покликаний гарантувати правильну реакцію серверу на запит аутентифікації та забезпечити безпечність сесій.

Напишемо юніт-тест для перевірки створення нового запису.

Лістинг 4.2 — Перевірка створення нового запису

```
def test_create_new_post():
    # Створення тестового користувача та автентифікація
```

```
user = User(username='testuser', password='testpassword')
login_user(user)
# Запит на сервер для створення нового запису
response = app.test_client().post('/create_post', data=dict(
    title='Test Post',
    content='This is a test post content.'
))
# Перевірка коду відповіді сервера
assert response.status_code == 200
# Перевірка, чи новий запис додано до бази даних
assert Post.query.filter_by(title='Test Post').first() is not None
```

У цьому тесті ми симулюємо створення нового поста на сервері. Спочатку ми автентифікуємо тестового користувача, щоб впевнитися, що він має право на створення записів. Потім ми надсилаємо дані на сервер для створення нового поста і перевіряємо, чи код відповіді 200. Останнім етапом є перевірка, чи новий пост дійсно збережено у базі даних. Цей тест гарантує, що функціональність створення записів працює вірно та зберігає дані коректно.

Тепер напишемо юніт-тест на перевірку відправки запиту на оновлення даних.

Лістинг 4.3 — Перевірка відправки запиту на оновлення даних

```
def test_update_user_data():
    # Створення тестового користувача та автентифікація
    user = User(username='testuser', password='testpassword')
    login_user(user)
    # Запит на сервер для оновлення даних користувача
    response = app.test_client().post('/update_user', data=dict(
        username='updated_username',
        email='updated_email@example.com'
    ))
```

```
    # Перевірка коду відповіді сервера
    assert response.status_code == 200
    # Перевірка, чи дані користувача оновлено в базі даних
    updated_user = User.query.filter_by(username='updated_username').first()
    assert updated_user is not None
    assert updated_user.email == 'updated_email@example.com'
```

У даному тесті ми перевіряємо правильність оновлення даних користувача на сервері. Спочатку ми автентифікуємо тестового користувача та відправляємо дані на сервер для оновлення. Після цього ми перевіряємо, чи код відповіді 200, що свідчить про успішне оновлення. Крім того, ми перевіряємо, чи дані користувача дійсно оновлені в базі даних. Цей тест гарантує правильність функціональності оновлення користувача та збереження змін у базі.

Далі напишем тест для функції валідації паролю.

Лістинг 4.4 — Функція валідації паролю

```
def test_password_validation():
    # Тестуємо функцію валідації паролю
    assert validate_password('StrongPassword123') == True
    assert validate_password('weak') == False
    assert validate_password('12345678') == False
    assert validate_password('SecureP@ss') == True
```

У цьому тесті ми передаємо різні паролі до функції `validate_password` і перевіряємо, чи повертається очікуване значення (True або False) в залежності від силки паролю.

Напишемо юніт-тест для функції видалення користувача.

Лістинг 4.5 — Функція видалення користувача

```
def test_remove_user():
    # Створення тестового користувача
```

```
    test_user = User(username='testuser', email='test@example.com')
    # Додаємо користувача до бази даних
    add_user_to_database(test_user)
    # Видаляємо користувача
    remove_user(test_user)
    # Перевіряємо, чи користувача вже немає в базі даних
    assert user_not_in_database(test_user)
```

У цьому тесті ми додаємо тестового користувача до бази даних, а потім видаляємо його за допомогою функцій `add_user_to_database` і `remove_user`. Після цього перевіряємо, чи користувача вже немає в базі даних за допомогою функції `user_not_in_database`.

І на кінець напишемо тест для функції редагування даних користувача.

Лістинг 4.6 — Функція редагування даних користувача

```
def test_edit_user_data():
    # Створення тестового користувача
    test_user = User(username='testuser', email='test@example.com')
    # Додаємо користувача до бази даних
    add_user_to_database(test_user)
    # Редагуємо дані користувача
    new_email = 'updated_email@example.com'
    edit_user_data(test_user, email=new_email)
    # Перевіряємо, чи дані користувача змінилися в базі даних
    updated_user = get_user_by_username('testuser')
    assert updated_user.email == new_email
```

У результаті написання тестів для аутентифікації користувача виявилося, що можна успішно взаємодіяти з сервером та отримувати очікувані результати. Тести враховують важливі аспекти, такі як коректність аутентифікації, наявність сесії та відповідність активного користувача тестовим даним.

Створення нового поста також пройшло успішно, що вказує на правильне функціонування логіки створення записів на сервері. Тести для оновлення даних користувача вказують на успішність цього процесу, включаючи коректність збереження змін у базі даних.

Загалом, написані тести виявилися ефектив ним інструментом для забезпечення правильності та стабільності важливих функціональностей додатку. Юніт-тести допомагають уникнути потенційних помилок, поліпшують розуміння коду та сприяють підтримці високої якості програмного забезпечення.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Оцінювання комерційного потенціалу розробки

Метою проведення комерційного та технологічного аудиту є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Метою проведення комерційного аудиту є оцінювання комерційного потенціалу впровадження методів та засобів, розробленої системи адаптивного тестування знань.

Для проведення технологічного аудиту було залучено 3— х незалежних експертів Вінницького національного технічного університету к.т.н., доцента Добровольську Наталію Вікторівну., к.т.н., доцента Снігура Анатолія Васильовича., к.т.н., доцента Черняка Олександра Івановича з кафедри обчислювальної техніки. Аудит науково-технічної розробки та її комерційного потенціалу проведено за допомогою таблиці 5.1, застосовуючи п’ятибальну шкалу оцінювання за 12— ма критеріями оцінки.

Таблиця 5.1 — Критерії оцінювання комерційного потенціалу розробки

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Критерій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогі	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів

Продовження таблиці 5.1

4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово—промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві

Продовження таблиці 5.1

11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10— ти років	Термін реалізації ідеї від 3— х до 5— ти років. Термін окупності інвестицій більше 5— ти років	Термін реалізації ідеї менше 3— х років. Термін окупності інвестицій від 3— х до 5— ти років	Термін реалізації ідеї менше 3— х років. Термін окупності інвестицій менше 3— х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

В таблиці 5.2 наведено результати оцінювання науково-технічного рівня і комерційного потенціалу розробки.

Таблиця 5.2 — Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	1. Добровольська Н. В.	2. Снігур А. В.	3. Черняк О. І.
	Бали, виставлені експертами:		
1	2	4	1
2	3	2	3
3	4	3	2
4	3	1	2
5	1	3	4

Продовження таблиці 5.2

6	2	3	3
7	1	4	4
8	2	3	2
9	3	3	2
10	3	3	2
11	4	2	4
12	3	2	2
Сума балів	$СБ_1$=31	$СБ_2$=33	$СБ_3$=31
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_1^3 СБ_i}{3} = \frac{31+33+31}{3} = 31.6$		

В таблиці 5.3 наведено шкалу оцінки комерційного потенціалу розробки.

Таблиця 5.3 — Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0— 10	Низький
11— 20	Нижче середнього
21— 30	Середній
31— 40	Вище середнього
41— 48	Високий

За результатами розрахунків, наведених в таблиці 5.2 та шкалою оцінки наведеної в таблиці 5.3 можна зробити висновок щодо рівня комерційного

потенціалу розробки. Середньоарифметична сума балів, виставлених експертами склала 31.6, що відповідає рівню «вище середнього».

Досягнення високого комерційного потенціалу відбулося завдяки значному зниженню витрат ресурсів і часу, витрачених на проведення тестування знань.

В якості аналога для розробки веб-додатку було обрано Jira — це інструмент для управління проектами та відстеження дефектів, розроблений компанією Atlassian. Серед недоліків можна зазначити, що деякі користувачі можуть відзначати високий рівень складності, особливо для новачків, і можливість перевантаження інформацією. Також, в залежності від конфігурації, Jira може вимагати часу для налагодження та адаптації до конкретних потреб команди. У розробці дана проблема вирішується шляхом спрощення інтерфейсу та розвантаження інформації по групах.

У таблиці 5.4 наведені основні технічні показники аналога і нового програмного продукту

Таблиця 5.4 — Основні технічні показники аналога і нового програмного продукту

Показники	Аналог	Нова розробка	Відношення параметрів нової розробки до параметрів аналога
Надійність	95%	95%	1/1
Сумісність	96%	96%	1/1
Супровід	90%	91%	1/1
Брендування	85%	85%	1/1
Зв'язок з HRM системою	45%	95%	1/2

Отож, з отриманих результатів у таблиці 5.4 можна зробити висновок що існує зацікавленість серед деяких сторін у нововведеннях розроблених в наслідок виконання роботи.

5.2 Прогнозування витрат на виконання науково-дослідної роботи

Витрати на здійснення науково-дослідної роботи розраховуються за наступними категоріями: витрати на оплату праці, відрахування на соціальні заходи, матеріали, програмне забезпечення для наукових робіт, накладні (загальновиробничі) витрати та ін. Обрахуємо витрати за кожною категорією.

Заробітна плата кожного із залучених осіб визначається за такою формулою:

$$З_0 = \sum_{i=1}^{K} \frac{M_{ni} * t_i}{T_p} \text{ (грн)} \quad (5.1)$$

де k — кількість посад працівників, залучених до процесу дослідження і розробки;

M_{ni} — місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.;

$T_р$— число робочих днів в місяці; приблизно $T_р = 22$;

t — кількість робочих днів роботи працівника.

Для проектування і розробки веб-додатку було залучено таких працівників: веб- розробник, дизайнер та тестувальник. Посадові оклади, число днів роботи та витрати на компенсацію наведено в таблиці 5.4

Таблиця 5.4 — Компенсація спеціаліста в дослідницькій установі

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату грн.
Веб- розробник	35000	1590	28	44520
Дизайнер	25000	1136	15	17040
Тестувальник	20000	909	5	4545
Всього				66105

Додаткова заробітна плата $З_д$ всіх робітників, які приймали участь в розробці нового технічного рішення розраховується за формулою (5.2) як 10 — 15 % від основної заробітної плати робітників як премія.

На даному підприємстві додаткова заробітна плата нараховується в розмірі 10% від основної заробітної плати.

$$З_д = (З_о + З_р) * \frac{Н_{дод}}{100\%} \quad (5.2)$$

$$З_д = 0{,}1 * 66105 = 6610{,}5 \text{ грн}$$

Нарахування на заробітну плату $Н_{ЗП}$ робітників, які брали участь у виконанні роботи, розраховуються за формулою (5.3):

$$З_н = (З_о + З_р + З_д) * \frac{Н_{ЗП}}{100} \text{ (грн)} \qquad (5.3)$$

де $З_о$ — основна заробітна плата розробників, грн.;

$З_д$ — додаткова заробітна плата всіх розробників та робітників, грн.;

$Н_{ЗП}$ — ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, % .

Основна ставка єдиного внеску на загальнообов'язкове державне соціальне страхування на 2023 рік — 22 %, тоді:

$$З_н = (66105 + 6610{,}5\ ) * 0.22 = 15{,}997{,}41(\text{грн})$$

Розрахуємо витрати на матеріали, пристрої, засоби, які використовують при виготовленні одиниці продукції. Розраховуються, згідно їх номенклатури, за формулою:

$$К=\sum_{i=1}^{n} Н_i \cdot Ц_i \cdot К_i \quad , \qquad (5.4)$$

де $Н_i$ — кількість комплектуючих *i*— го виду, шт.;

$Ц_i$ — покупна ціна комплектуючих *i*— го найменування, грн.;

$К_i$ — коефіцієнт транспортних витрат (1,1…1,15).

Потрібно закладати витрати на доставку у вигляді коефіцієнту транспортних витрат — 1.1. Інформацію про використані матеріали та комплектуючі наведено у таблиці 5.5.

Таблиця 5.5 — Матеріали, що використані на розробку

Найменування матеріалу	Ціна за одиницю, грн.	Витрачено	Вартість витраченого матеріалу, грн.
Ноутбук	35000	2	70000
Всього			70000
З врахуванням коефіцієнта транспортування			77000

Розрахуємо витрати на програмне забезпечення, яке необхідне для проектування та розробки веб-додатку. Балансову вартість програмного забезпечення розраховують за формулою 5.5:

$$В_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot С_{пргі} \cdot К_i, \qquad (5.5)$$

де $Ц_{iпрг}$ — ціна придбання/використання одиниці програмного засобу цього виду, грн;

$С_{пргі}$ — кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

$К_i$ — коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо ($К_i$ = 1,10…1,12).

k — кількість найменувань програмних засобів.

Отримані результати наведено в таблиці 5.6.

Таблиця 5.6 — Витрати на використання програмних

Найменування устаткування	Час використання, місяців	Ціна за місяць, грн	Вартість, грн
Підписка Visual Studio Code	2	2500	5000
Підписка MS SQL Server Standard	2	1700	3400
Всього			8400

Розрахуємо амортизаційні відрахування по кожному виду обладнання, устаткування яке використовувалось для проектування та розробки системи адаптивного тестування знань.

$$А_{обл} = \frac{Ц_б}{Т_в} * \frac{t_{вик}}{12} \quad (5.6)$$

де $Ц_б$ — балансова вартість даного виду обладнання (приміщень), грн.;

$t_{вик}$ — час користування;

$Т_в$ — термін використання обладнання (приміщень), цілі місяці.

Для розробки функціоналу та дизайну веб- долатку використовувався ноутбук вартістю 35000 грн. Для тестування використовувався інший ноутбук вартістю 35000 грн. Амортизаційні відрахування наведено в таблиці 5.7.

Таблиця 5.7 — Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Ноутбук	35000	2	2	1400
Офісне приміщення	2000000	30	2	11111
Всього				125511

Витрати на енергію визначаються на основі витрат на одиницю продукції та тарифів на енергію за допомогою формули:

$$В_е = В \cdot П \cdot Ф \cdot К_п \text{ [грн]}, \quad (5.5)$$

де В — вартість 1кВт електроенергії;

П — установлена потужність обладнання, кВт;

Ф — фактична кількість годин роботи комп'ютера при створенні програмного продукту, годин;

$К_п$ — коефіцієнт використання потужності.

Отже, витрати на енергію становлять:

$$В_е = 7 \cdot 0{,}4 \cdot 520 \cdot 0{,}3 = 436{,}8 \text{ (грн)}.$$

Витрати за доступ до Інтернет можна розрухувати за формулою:

$$В_{ді} = Ц_{ді} \cdot Т \text{ [грн]}, \quad (5.6)$$

де $Ц_{ді}$ — це ціна доступу за місяць;

Т — кількість місяців використання доступу до мережі.

$$В_{ді} = 230 \cdot 3 = 690{,}00 \text{ (грн)}.$$

Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати робітників за формулою 5.7

$$B_{ін} = З_о \cdot 100\% \text{ [грн]}. \quad (5.7)$$

$$B_{ін} = 66105 \cdot 0{,}5 = 33052 \text{ (грн)}.$$

Дана стаття витрат охоплює витрати на управління організацією, оплату службових відряджень, витрати на утримання, ремонт та експлуатацію основних засобів, витрати на опалення, освітлення, водопостачання, охорону праці тощо. Накладні (загальновиробничі) витрати $B_{нзв}$ можна прийняти як 120% від суми основної заробітної плати розробників та робітників, які виконували дану МКНР, тобто:

$$\text{В}_{\text{нзв}} = \left(\text{З}_{\text{о}} + \text{З}_{\text{р}}\right) \cdot \frac{\text{Н}_{\text{нзв}}}{100}, \quad (5.8)$$

де $\text{Н}_{\text{нзв}}$ — норма нарахування за статтею «Загальновиробничі витрати».

$$\text{В}_{\text{нзв}} = 66105 * 1.2 = 79326 \text{ грн}$$

Сума всіх статей витрат дає в результаті витрати на проведення дослідження та розробку адаптивної системи тестування знань і розраховується за формулою:

$$\text{В} = \text{З}_{\text{о}} + \text{З}_{\text{дод}} + \text{З}_{\text{н}} + \text{К}_{\text{в}} + \text{В}_{\text{прг}} + \text{А}_{\text{обл}} + \text{В}_{\text{е}} + \text{В}_{\text{ін}} + \text{В}_{\text{нзв}} \quad (5.9)$$

$$\text{В} = 66105 + 6610{,}5 + 15997{,}41 + 77000 + 8400 + 125511 + 436.8 + 690 + 33052 + 79326 = 413128{,}71$$

Загальні витрати ЗВ на завершення роботи та оформлення її результатів розраховуються за формулою:

$$\text{ЗВ} = \frac{\text{В}}{\eta}, \quad (5.10)$$

де η — коефіцієнт, який характеризує стадію виконання даної НДР.

Так, якщо розробка знаходиться:

— на стадії науково— дослідних робіт, то $\beta \approx 0{,}1$;

— на стадії технічного проектування, то $\beta \approx 0{,}2$;

— на стадії розробки конструкторської документації, то $\beta \approx 0{,}3$;

— на стадії розробки технологій, то $\beta \approx 0{,}4$;

— на стадії розробки дослідного зразка, то $\beta \approx 0{,}5$;

— на стадії розробки промислового зразка, $\beta \approx 0{,}7$;

— на стадії впровадження, то $\beta \approx 0{,}9$.

Звідси:

$$ЗВ = \frac{413128{,}71}{0.5} = 826257{,}42 \text{ грн.}$$

Витрати на виконання наукової роботи та впровадження її результатів становитиме 826 257,42 грн.

5.3 Розрахунок економічної ефективності науково-технічної розробки

Економічна ефективність дозволяє спрогнозувати чистий прибуток, який може бути отриманий від впровадження розробленого веб-додатку.

При розрахунку економічної ефективності потрібно обов’язково враховувати зміну вартості грошей у часі, оскільки від вкладення інвестицій до отримання прибутку минає чимало часу.

Для розрахунку збільшення чистого прибутку підприємства $\Delta П_i$, для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки використовується формула формулою 5.11:

$$\Delta П_i = \sum_1^n \left(\Delta Ц_o \cdot N + Ц_o \cdot \Delta N\right)_i \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\upsilon}{100}\right) \quad (5.11)$$

де $\Delta Ц_0$ — покращення основного оціночного показника від впровадження результатів розробки у даному році.

N — основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження нової розробки;

ΔN — покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки:

Цо — основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів розробки;

n — кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки:

λ — коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт *л* = 0,8333.

ρ — коефіцієнт, який враховує рентабельність продукту. р = 0,3;

ϑ — ставка податку на прибуток. У 2023 році — 18%.

В результаті впровадження наукової розробки покращується якість певного продукту, що дозволяє підвищити ціну його реалізації на 2000 грн. Кількість користувачів збільшиться протягом першого року на 100, другого — 200, третього — 300. Реалізація продукції до впровадження результатів наукової розробки складала 2000 користувачів, а її ціна — 20000 грн. Розрахуємо показник прибутку впродовж трьох років відносно базового.

$$\Delta П_1 = 10000 \cdot 1 + (20000 + 2000) \cdot 200) \cdot \textit{0,8333} \cdot \textit{0,3} \cdot \textit{(1 — 18/100)} = 900433{,}8{,}98 \text{ (грн)}.$$

$$\Delta П_1 = (10000 \cdot 1 + (20000 + 2000) \cdot 400) \cdot 0{,}8333 \cdot 0{,}3 \cdot (1 — 18/100) = 1798825{,}80 \text{ (грн)}.$$

$$\Delta П_1 = (10000 \cdot 1 + (20000 + 2000) \cdot 600) \cdot 0{,}8333 \cdot 0{,}3 \cdot (1 — 18/100) = 2697217{,}80 \text{ (грн)}.$$

Далі за формулою 5.12 розраховують приведену вартість збільшення всіх чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t} \quad (5.12)$$

де ΔПі — збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково— технічної розробки, грн;

Т — період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації розробки, роки;

τ — ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,05…0,15. В Україні рівень інфляції за підсумком 2023 року склав 10.6%, прогнозований рівень на 2024 рік — 8.5% ;

t — період часу (в роках) від моменту початку впровадження розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$ПП = \frac{900433{,}8{,}98}{(1+0{,}15)^1} + \frac{1798825{,}8}{(1+0{,}15)^2} + \frac{2697217{,}8}{(1+0{,}15)^3} = 6\,916\,662 \text{ (грн).}$$

Отже, розрахунки показують, що комерційний ефект від впровадження розробки виражається у значному збільшенні чистого прибутку підприємства.

Основними показниками, які визначають доцільність фінансування наукової розробки інвестором, є абсолютна і відносна ефективність вкладених інвестицій та термін їх окупності.

Якщо $E_{абс} > 0$, то результат від проведення наукових досліджень та їх впровадження принесе прибуток, але це також ще не свідчить про те, що інвестор буде зацікавлений у фінансуванні даного проекту (роботи).

Розрахуємо абсолютну ефективність інвестицій, вкладених у реалізацію проекту. Ставка дисконтування τ дорівнює 0,1. Отримаємо:

Розраховуємо величину початкових інвестицій PV, які розробник (замовник) має вкласти для здійснення науково— технічної розробки.

Для цього можна використати формулу:

$$PV = k_{розр} \cdot ЗВ \text{ [грн]}, \quad (5.13)$$

де розр k — коефіцієнт, що враховує витрати розробника (замовника) на впровадження науково— технічної розробки. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай розр k =2...5, але може бути і більшим;

ЗВ — загальні витрати на проведення науково— технічної розробки та оформлення її результатів, грн.

$$PV = 2 \cdot 826257{,}42 = 1\,652\,514{,}84 \text{ грн}$$

Тоді абсолютний економічний ефект $E_{абс}$ або чистий приведений дохід (NPV, Net Present Value) для потенційного інвестора від можливого впровадження та комерціалізації розробки становитиме:

$$E_{абс} = ПП - PV \quad (5.14)$$

де ПП — приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково— технічної розробки, грн;

PV — теперішня вартість початкових інвестицій, грн.

$$E_{абс} = 6\,916\,662 - 1\,652\,514{,}84 = 5264147{,}16 \text{ грн}$$

Оскільки величина економічного ефекту має велике додатне значення, це свідчить про потенційну зацікавленість інвесторів у впровадженні та комерціалізацію розробки.

5.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Для остаточного прийняття рішення про впровадження розробки та виведення її на ринок необхідно розрахувати внутрішню економічну дохідність

$Е_в$ або показник внутрішньої норми дохідності (IRR, Internal Rate of Return) вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь— яку розробку вкладати буде економічно недоцільно.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $Е_в$. Для цього користуються формулою 5.15:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{5.15}$$

де $Е_{абс}$ — абсолютний економічний ефект вкладених інвестицій, грн; PV — теперішня вартість початкових інвестицій, грн;

$Т_ж$ — життєвий цикл науково— технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, роки.

$$Е_в = \sqrt[3]{1 + \frac{5264147,16}{1\,652\,514,84}} - 1 = 0.72 = 72\%$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою 5.16:

$$\tau = d + f, \tag{5.16}$$

де d — середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d= (0,14…0,2);

f−показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,1).

$$\tau_{min} = 0.14 + 0{,}5 = 0.19$$

Так як $E_в > \tau_{min}$ то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Далі розраховується період окупності інвестицій, вкладених у реалізацію проекту за формулою 5.17.

$$T_{ок} = \frac{1}{E_в} \tag{5.17}$$

де Ев — внутрішня економічна дохідність вкладених інвестицій.

$$\mathrm{T_{ок}} = \frac{1}{0.72} = 1.38 \text{ роки}$$

Термін окупності складає 3 роки, що свідчить про доцільність фінансування даної наукової розробки.

5.5 Результати економічного аналізу

В даному розділі було проведено оцінку комерційного потенціалу розробки програмного забезпечення для адаптивного тестування знань. Економічний потенціал склав значення, яке є вище середнього.

Також було зпрогнозовано витрати на проектування та реалізацію системи за необхідними статтями витрат, які склали 826 257,42 грн.

Обрахунок економічної ефективності та терміну окупності вкладених інвестицій показали, що розробка є економічно доцільною та привабливою для потенційних інвесторів. Термін окупності інвестицій складає 1.38 роки.

ВИСНОВКИ

В результаті виконання дипломної роботи було створено веб- додаток для автоматизованої системи контролю якості виробництва. Розроблювальний веб-додаток, насамперед, виступає як інноваційний каталізатор для підприємств, що мають на меті підвищення якості своєї продукції. Його важливість полягає в тому, що він створює інтелектуальне та автоматизоване середовище для нагляду та управління виробничим процесом. Забезпечуючи широкий спектр функцій, додаток стає невід'ємним інструментом для підприємств, щоб оптимізувати якість своєї продукції та досягати високих стандартів ефективності.

У першому розділі було проведено аналіз сучасних методів та технологій системи контролю якості виробництва, а саме огляд сучасних інформаційних технологій для автоматизації контролю якості, вплив інноваційних технологій на контроль якості, роль веб-додатків у підвищенні ефективності системи контролю якості та аналіз аналогів системи.

У другому розділі було проаналізовано та обрано інструменти для розробки веб-додатку для автоматизованої системи контролю якості виробництва. Описано архітектуру «клієнт-серверної» взаємодії, основи та практичні аспекти розробки web-додатків. Проаналізовано середовища та технології, які потрібні у розробці веб-додатку.

У третьому розділі було здійснено розробку веб-додатку, його інтерфейс та функціонал. Розроблено базу даних, створено ER-діаграму та блок-схему алгоритму роботи додатку.

У четвертому розділі було проведено тестування користувацького інтерфейсу.

У п’ятому розділі було виконано економічні розрахунки, в результаті яких зроблено висновок, що потенційна зацікавленість інвесторів у фінансуванні розробки є досить високою.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Салата О. Л. Добровольська Н. В. «Web-додаток для автоматизованої системи контролю якості виробництва». Матеріали конференції «LI Науково-технічна конференція підрозділів Вінницького національного технічного університету (2023)», Вінниця, 2023. [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2024/paper/view/19700

2. Internet of Things, IoT [Електронний ресурс]. — Режим доступу: https://www.it.ua/knowledge-base/technology-innovation/internet-veschej-internet-of- things-iot

3. Технологія комп'ютерного зору IoT [Електронний ресурс]. — Режим доступу: https://brainberry.ua/uk/newsroom/blog/computer-vision-technology-common- mistakes

4. Контроль якості продукції та премії в області якості [Електронний ресурс]. Режим доступу: https://elib.lntu.edu.ua/sites/default/ files/elib_upload/Самчук%20Л.М/page6.html

5. Сучасні технології [Електронний ресурс]. — Режим доступу:https://gigacloud.ua/blog/navchannja/scho-take-shtuchnij-intelekt-istorija-vid-ta-skladovi

6. Галлеєв В. І. Управління якістю: проблеми, перспективи: навчальний посібник / В. І. Галлеєв, М. К. Варгіна. М. Вільямс, 2017. — 272 с.

7. Пономарьов С. В. Управління якістю продукції. Введення в систему менеджменту якості: навчальний посібник / С. В. Пономарьов. — М.: Стандарти та якість, 2019. — 248 с.

8. Quality Management [Електронний ресурс]. — Режим доступу: https://help.sap.com/docs/SAP_S4HANA_ONPREMISE/2bc3ee8d1c83404e8cf62418640004f2/cbc48c570c9b7010e10000000a441470.html

9. Zoho Quality Management [Електронний ресурс]. — Режим доступу: https://www.zoho.com/projects/project-management-for-qa-teams.html

10. ComplianceQuest [Електронний ресурс]. — Режим доступу: https://www.compliancequest.com

11. Клієнт— серверна архітектура [Електронний ресурс]. — Режим доступу: https://medium.com/@IvanZmerzlyi/клієнт-серверна-архітектура-та-рол-серверів-9893d8048229

12. Основи та практичні аспекти розробки web-додатків [Електронний ресурс]. — Режим доступу: http://sites.znu.edu.ua/webprog/lect/1191.ukr.html

13. What is HTML? [Електронний ресурс] — режим доступу: https://www.hostinger.com/tutorials/what-is-html

14. Основи CSS [Електронний ресурс] — режим доступу: https://www.w3.org/Style/Examples/007/units.en.html

15. CSS для початківців [Електронний ресурс] — режим доступу: https://www.tutorialspoint.com/css/css_measurement_units.htm

16. What is JavaScript [Електронний ресурс]. Режим доступу: https://www.tutorialsteacher.com/javascript/what-is-javascript Дата звернення: Травень 21, 2020.

17. JS Theories [Електронний ресурс] — режим доступу: https://is.theorizeit.org/wiki/Main_Page

18. Мішин В. М. Управління якістю: підручник / В. М. Мішин. — М.:ІНФРА— М, 2019. — 463 с.

19. Пономарьов С. В. Управління якістю продукції. Інструменти і методи менеджменту якості: навчальний посібник / С. В. Пономарьов, С. В. Міщенко, В. Я. Білобрагін. — М.: Стандарти та якість, 2019. — 248 с.

20. Світкін М. З. Процесний підхід при впровадженні системи менеджменту якості в організації / М. З. Світкін // Стандарти і якість. — 2020. - №3. -С. 74 -77.

21. Методичні вказівки до виконання магістерських кваліфікаційних робіт студентами спеціальності 123 «Комп'ютерна інженерія» другого (магістерського) рівня вищої освіти денної та заочної форм навчання. / Укладачі О.Д. Азаров, О.В. Дудник, С.І. Швець − Вінниця : ВНТУ, 2023. − 57с.

22. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. — Вінниця : ВНТУ, 2021. 42 с.,

23. Все що потрібно знати про Progressive Web [Електронний ресурс] — Режим доступу до ресурсу: https://habr.com/ru/company/wrike/blog/481240/

24. Методи життєвого циклу [Електронний ресурс] — Режим доступу до ресурсу: https://dev-gang.ru/article/kak-ponjat-metody-zhiznennogo-cikla-komponenta-vreactjs-m3v6725v7q/ 7

25. Node.js Documentation [Електронний ресурс] — Режим доступу до ресурсу:https://nodejs.org/dist/latest-v14.x/docs/api/

ДОДАТОК А

Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ

проф., д.т.н.. Азаров О.Д..

"29" вересня 2023 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

"Web-додаток для автоматизованої системи контролю якості виробництва "
08-54.МКР.040.00.000 ПЗ

Науковий керівник: доцент
к.пед.н. каф.ОТ

_____ Добровольська Н. В.

Студент групи 2КІ-22м

_____ Салата О. Л.

ВНТУ 2023

1 Підстава для виконання магістерської кваліфікаційної роботи (МКР)

1.1 Актуальність теми дослідження полягає в тому, що розроблювальний додаток забезпечує швидкий та ефективний моніторинг усіх етапів виробничого процесу та дозволяє збільшити ефективність, знизити витрати та мінімізувати ризик виробничих невдач. Такий веб-додаток є актуальним не лише для підприємств, але й для клієнтів, які отримають гарантію якості та можливість впливати на виробництво за допомогою ефективного моніторингу.

1.2 Наказ про затвердження теми МКР.

2 Мета МКР і призначення розробки

2.1 Мета роботи — розробка веб-додатку для автоматизованої системи контролю якості виробництва, з можливістю покращення інформаційне обслуговування працівників, завдяки IoT-технологіям.

2.2 Призначення розробки — визначається необхідністю створення веб-додатку, що дозволить підприємству отримувати повний контроль над якістю продукції,

3 Вихідні дані для виконання МКР

3.1 Проведення аналізу існуючих методів та принципів.

3.2 Розробка алгоритму роботи веб- додатку.

3.4 Проведення верифікації та аналізу отриманих результатів.

3.5 Виконання розрахунків для доведення доцільності нової розробки з економічної точки зору.

4 Вимоги до виконання МКР

Головна вимога — що веб-додаток має надавати інформаційну підтримку користувачам, щодо якості продукції на виробництві.

5 Етапи МКР та очікувані результати

Етапи роботи та очікувані результати приведено в Таблиці А.1.

Таблиця А.1 — Етапи МКР

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз існуючих технологій, огляд аналогів.	19.09.2023	27.09.2023	Розділ 1
2	Визначення архітектури веб-додатку	6.10.2023	18.10.2023	Розділ 2
3	Розробка алгоритму та функціоналу веб-додатку	18.10.2023	25.10.2023	Розділ 3
4	Тестування системи	25.10.2023	28.10.2023	Розділ 4
4	Підготовка економічної частини	5.11.2023	15.11.2023	Розділ 5
6	Оформлення пояснювальної записки, графічного матеріалу і презентації	4.12.2023	8.12.2023	ПЗ, графічний матеріал і презентація
7	Підготовка і підпис супроводжуючих документів, нормоконтроль та тест на плагіат	11.12.2023	15.12.2023	Оформленні документи

6 Матеріали, що подаються до захисту МКР

До захисту подаються: пояснювальна записка МКР, графічні і ілюстративні матеріали, протокол попереднього захисту МКР на кафедрі, відгук наукового керівника, відгук опонента, протоколи складання державних екзаменів, анотації до МКР українською та іноземною мовами.

7 Порядок контролю виконання та захисту МКР

Виконання етапів графічної та розрахункової документації МКР контролюється науковим керівником згідно зі встановленими термінами. Захист МКР відбувається на засіданні Екзаменаційної комісії, затвердженої наказом ректора.

8 Вимоги до оформлювання та порядок виконання МКР

8.1 При оформлюванні МКР використовуються:

— ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— ГОСТ 2.104–2006 «Єдина система конструкторської документації. Основні написи»;

— методичні вказівки до виконання магістерських кваліфікаційних робіт зі спеціальності 123 — «Комп’ютерна інженерія»;

— документи на які посилаються у вище вказаних.

8.2 Порядок виконання МКР викладено в «Положення про кваліфікаційні роботи на другому (магістерському) рівні вищої освіти СУЯ ВНТУ–03.02.02 П.001.01:21

ДОДАТОК Б

Блок-схема алгоритму роботи додатку з розділеною авторизацією

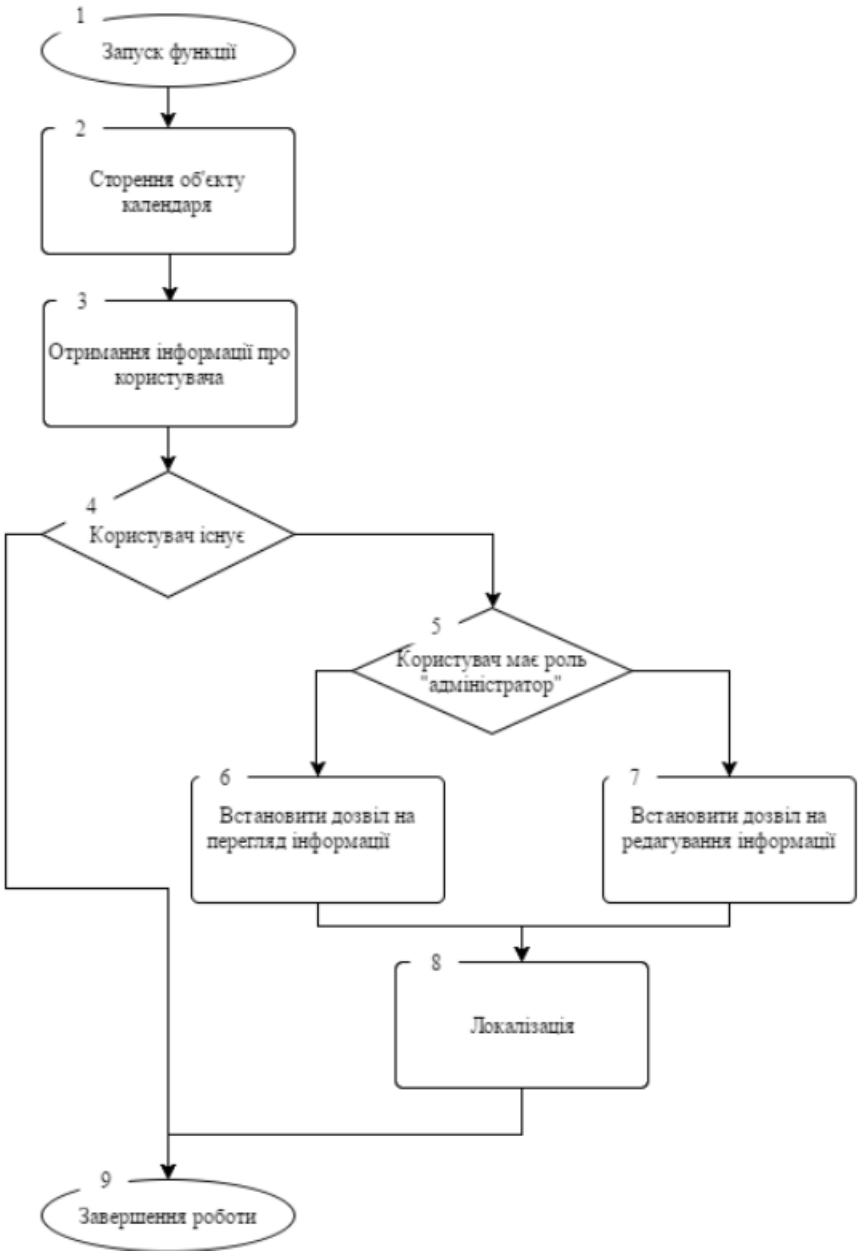

Рисунок Б.1 — Блок-схема алгоритму роботи додатку з розділеною авторизацією

ДОДАТОК В

Блок-схема алгоритму взаємодії IoT та веб технологій

Рисунок В.1 — Блок-схема алгоритму взаємодії IoT та веб технологій

ДОДАТОК Г

Лістинг сторінки з панелю управління

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF— 8">
    <meta name="viewport" content="width=device— width, initial— scale=1.0">
    <title>Панель управління</title>
    <!— —  Підключення бібліотеки Chart.js — — >
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <style>
        body {
            font— family: Arial, sans— serif;
            margin: 0;
            padding: 0;
            background— color: #f4f4f4;
        }

        main {
            max— width: 800px;
            margin: 20px auto;
            padding: 20px;
            background— color: #fff;
            border— radius: 8px;
            box— shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        }
```

```
        h1, h2 {
            color: #333;
        }
        p {
            margin— bottom: 10px;
        }
        canvas {
            margin— top: 20px;
        }
        button {
            padding: 10px;
            background— color: #4CAF50;
            color: #fff;
            border: none;
            border— radius: 4px;
            cursor: pointer;
        }
        button:hover {
            background— color: #45a049;
        }
    </style>
</head>
<body>
<main>
    <h1>Панель управління</h1>
    <h2>Огляд загальної інформації</h2>
```

```
    <p>Загальна кількість вироблених продуктів: 5000</p>
    <p>Кількість виявлених дефектів: 800</p>
    <p>Відсоток дефектів: 16%</p>
    <h2>Графіки та діаграми</h2>
    <!— —  Додавання canvas для графіка — — >
    <canvas id="qualityChart" width="400" height="200"></canvas>
    <button onclick="generateReport()">Згенерувати звіт</button>
</main>
<script>
    // Дані для графіка
    const productionData = {
        labels: ['Загальна кількість', 'Виявлені дефекти'],
        datasets: [{
            label: 'Продукція',
            data: [5000, 800],
            backgroundColor: [
                'rgba(75, 192, 192, 0.2)',
                'rgba(255, 99, 132, 0.2)',
            ],
            borderColor: [
                'rgba(75, 192, 192, 1)',
                'rgba(255, 99, 132, 1)',
            ],
            borderWidth: 1,
        }]
    };
```

```
    // Налаштування графіка
    const chartOptions = {
        scales: {
            y: {
                beginAtZero: true
            }
        }
    };
    // Створення графіка за допомогою Chart.js
    const qualityChart = new Chart(document.getElementById('qualityChart'), {
        type: 'bar',
        data: productionData,
        options: chartOptions
    });
    // Функція для генерації звіту
    function generateReport() {
        // Ваш код для генерації звіту тут
        alert("Звіт згенеровано!");
    }
</script>
</body>
</html>
```

ДОДАТОК Д

Лістинг сторінки з виявленням дефектів

```
<html lang="en">
<head>
    <meta charset="UTF— 8">
    <meta name="viewport" content="width=device— width, initial— scale=1.0">
    <title>Відстеження дефектів</title>
    <style>
        body {
            font— family: Arial, sans— serif;
            margin: 0;
            padding: 0;
            background— color: #f4f4f4;
        }

        header {
            background— color: #333;
            color: #fff;
            padding: 1rem;
            text— align: center;
        }
        main {
            max— width: 800px;
            margin: 20px auto;
            padding: 20px;
            background— color: #fff;
```

```
    box— shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}
form {
    display: flex;
    flex— direction: column;
}
label {
    margin— bottom: 8px;
}
input, select {
    margin— bottom: 16px;
    padding: 8px;
}
table {
    width: 100%;
    border— collapse: collapse;
    margin— top: 20px;
}
th, td {
    border: 1px solid #ddd;
    padding: 8px;
    text— align: left;
}
th {
    background— color: #333;
    color: #fff;
```

```
        }
    </style>
</head>
<body>
    <header>
        <h1>Відстеження дефектів</h1>
    </header>
    <main>
        <form id="defectForm">
            <label for="defectName">Назва дефекту:</label>
            <input type="text" id="defectName" name="defectName" required>
            <label for="defectStatus">Статус дефекту:</label>
            <select id="defectStatus" name="defectStatus" required>
                <option value="новий">Новий</option>
                <option value="в роботі">В роботі</option>
                <option value="вирішений">Вирішений</option>
            </select>
            <button type="submit">Зберегти дефект</button>
        </form>
        <h2>Звіт про дефекти</h2>
        <table>
            <thead>
                <tr>
                    <th>ID</th>
                    <th>Назва</th>
                    <th>Статус</th>
```

```
            </tr>
        </thead>
        <tbody id="defectTableBody">
        </tbody>
    </table>
</main>
<script>
    const defectForm = document.getElementById('defectForm');
    const defectTableBody = document.getElementById('defectTableBody');
    defectForm.addEventListener('submit', function(event) {
        event.preventDefault();
        const defectName = document.getElementById('defectName').value;
        const defectStatus = document.getElementById('defectStatus').value;
        const newRow = document.createElement('tr');
        newRow.innerHTML = `
            <td>${defectTableBody.children.length + 1}</td>
            <td>${defectName}</td>
            <td>${defectStatus}</td>
        defectTableBody.appendChild(newRow);
        // Очищення форми
        defectForm.reset();
    });
</script>
</body>
</html>
```

ДОДАТОК Е

Лістинг коду бази даних

```
<!DOCTYPE html>
— — Створення таблиці "Products"
CREATE TABLE IF NOT EXISTS Products (
  product_id INT AUTO_INCREMENT PRIMARY KEY,
  product_name VARCHAR(255),
  description TEXT,
  quality_manager_id INT,
  FOREIGN KEY (quality_manager_id) REFERENCES Users(user_id)
);
— — Створення таблиці "Defects"
CREATE TABLE IF NOT EXISTS Defects (
  defect_id INT AUTO_INCREMENT PRIMARY KEY,
  defect_name VARCHAR(255),
  description TEXT,
  responsible_user_id INT,
  FOREIGN KEY (responsible_user_id) REFERENCES Users(user_id)
);
— — Створення таблиці "QualityReports"
CREATE TABLE IF NOT EXISTS QualityReports (
  report_id INT AUTO_INCREMENT PRIMARY KEY,
  product_id INT,
  defect_id INT,
  user_id INT,
  report_date DATE,
  comments TEXT,
  FOREIGN KEY (product_id) REFERENCES Products(product_id),
  FOREIGN KEY (defect_id) REFERENCES Defects(defect_id),
  FOREIGN KEY (user_id) REFERENCES Users(user_id)
```

```
);

— — Створення таблиці "Users"
CREATE TABLE IF NOT EXISTS Users (
  user_id INT AUTO_INCREMENT PRIMARY KEY,
  username VARCHAR(255),
  password VARCHAR(255),
  role VARCHAR(50)
);
```

ДОДАТОК Ж

Лістинг забезпечення безпеки у базі даних

```
const express = require('express');
const bcrypt = require('bcrypt');
const jwt = require('jsonwebtoken');
const { Op } = require('sequelize');
const UserModel = require('./models/User');
const app = express();
const port = 3000;
const secretKey = 'your— secret— key';
app.use(express.json());
// Генерація JWT— токена
function generateToken(userId, username, role) {
  const payload = { userId, username, role };
  return jwt.sign(payload, secretKey, { expiresIn: '1h' });
}
// Створення нового користувача
app.post('/users', async (req, res) => {
  try {
    const { username, password } = req.body;
    // Хешування пароля
    const saltRounds = 10;
    const hashedPassword = await bcrypt.hash(password, saltRounds);
    // Збереження нового користувача
    const newUser = await UserModel.create({
      username,
      password: hashedPassword,
      role: 'user',
    });
```

```
    // Генерація JWT— токена для нового користувача
    const token = generateToken(newUser.user_id, newUser.username, newUser.role);
    res.status(201).json({ user: newUser, token });
  } catch (error) {
    console.error(error);
    res.status(500).send('Internal Server Error');
  }
});
// Автентифікація користувача та генерація токена
app.post('/login', async (req, res) => {
  try {
    const { username, password } = req.body;
    // Знаходження користувача за іменем
    const user = await UserModel.findOne({
      where: {
        username: {
          [Op.eq]: username,
        },
      },
    });
    // Перевірка правильності пароля
    if (user && (await bcrypt.compare(password, user.password))) {
      // Генерація JWT— токена для автентифікованого користувача
      const token = generateToken(user.user_id, user.username, user.role);
      res.json({ user, token });
    } else {
      res.status(401).send('Invalid username or password');
    }
  } catch (error) {
    console.error(error);
```

```
    res.status(500).send('Internal Server Error');
  }
});
// Захищений маршрут, доступний тільки для автентифікованих користувачів
app.get('/protected', (req, res) => {
  // Доступ до інформації користувача з JWT— токена
  const { userId, username, role } = req.user;
  res.json({ userId, username, role, message: 'This is a protected route' });
});
// Middleware для перевірки та розпакування JWT— токена
function authenticateToken(req, res, next) {
  const token = req.header('Authorization');
  if (!token) {
    return res.status(401).send('Access denied. Token is required.');
  }
  jwt.verify(token, secretKey, (err, user) => {
    if (err) {
      return res.status(403).send('Invalid token');
    }
    req.user = user;
    next();
  });
}
app.get('/protected', authenticateToken, (req, res) => {
  res.send('This is a protected route.');
});
// Слухаємо сервер на вказаному порті
app.listen(port, () => {
  console.log(`Server is running at http://localhost:${port}`);
})
```

ДОДАТОК И

Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

Назва роботи: Web-додаток для автоматизованої системи контролю якості виробництва

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ кафедра обчислювальної техніки
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 96,3% Схожість 3,7%

Аналіз звіту подібності (відмітити потрібне):

✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Захарченко С.М.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Салата О. Л.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Добровольська Н.В.
(підпис) (прізвище, ініціали)