Додаток А

Презентаційні слайди для захисту кваліфікаційної роботи

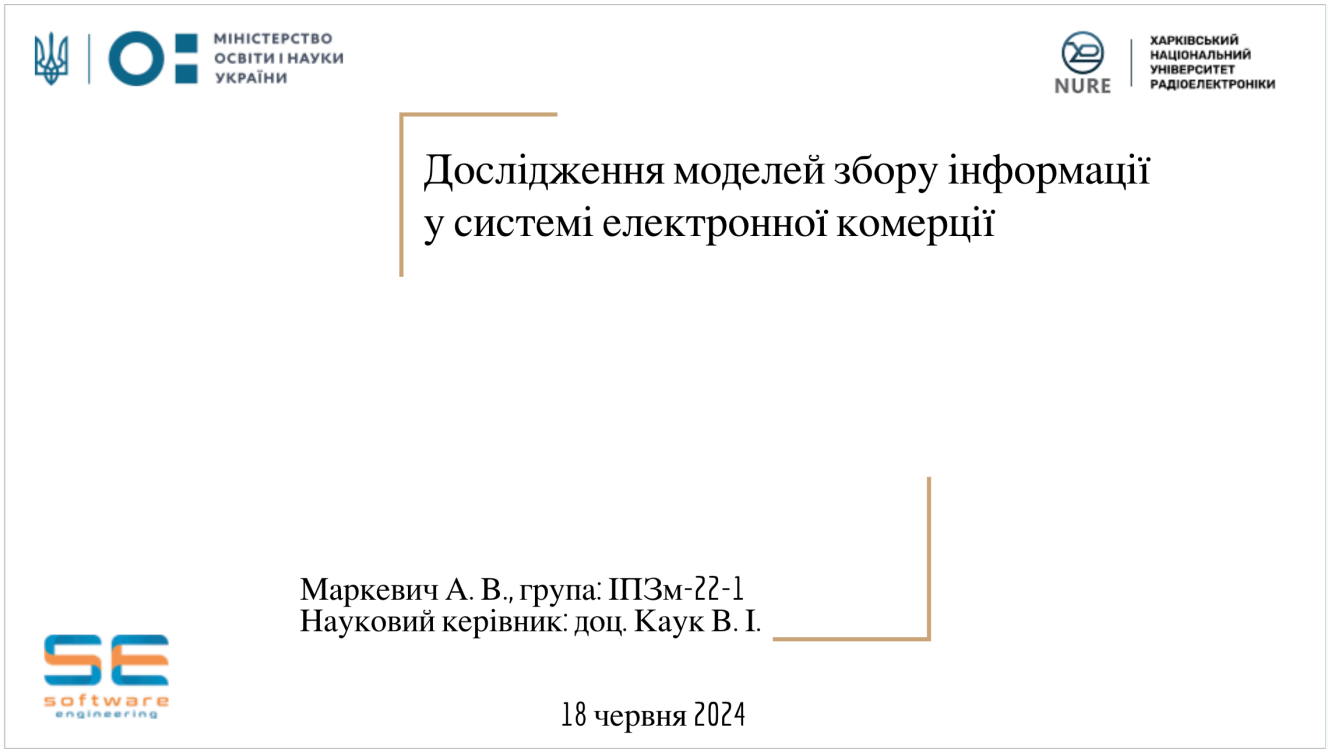



Рисунок А.1 – Перший слайд: «Титульний»

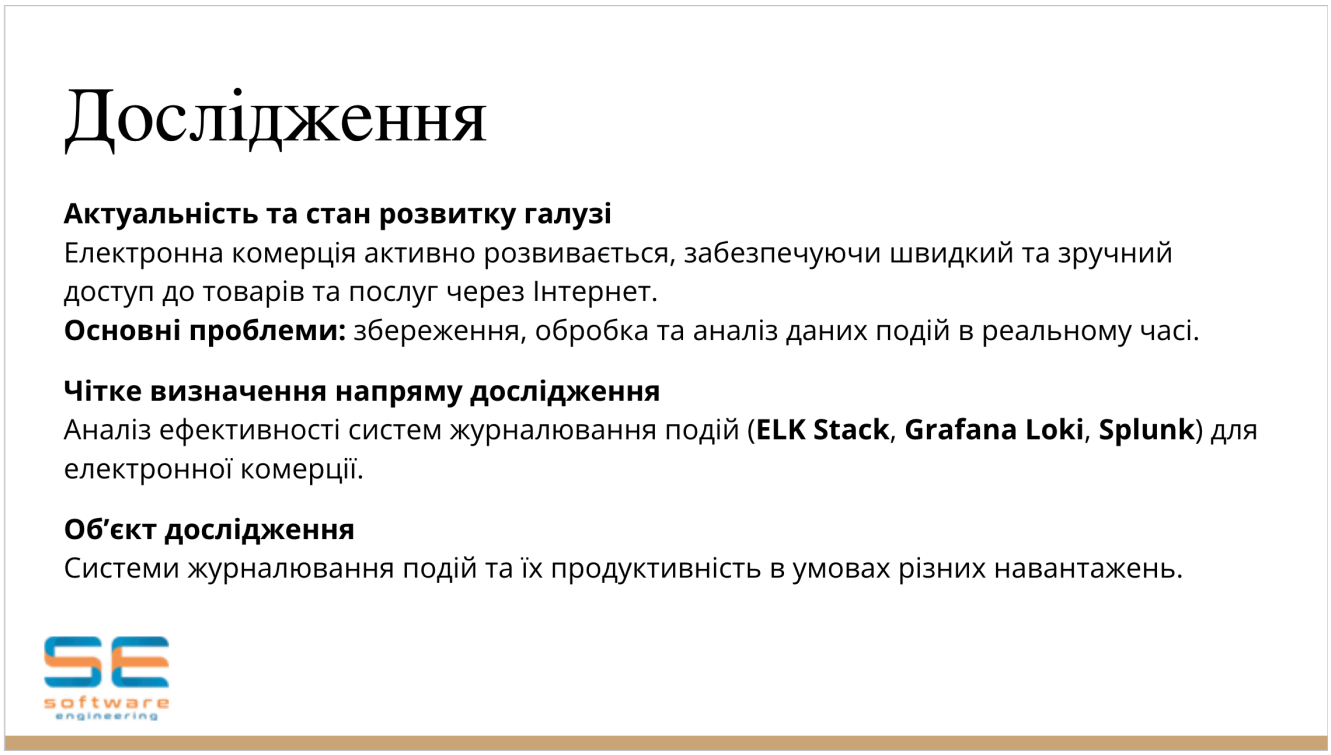



Рисунок А.2 – Другий слайд: «Дослідження»

# Огляд літератури (аналогів)

***Перелік основних джерел та теорій у галузі***
Принципи функціонування та архітектури **Elasticsearch (ELK Stack)**, **Grafana Loki** та **Splunk**.

***Зазначення прогалин у наявних дослідженнях***
Недостатня інформація про порівняльний аналіз продуктивності та ефективності зберігання даних цих систем в умовах різних навантажень.

Рисунок А.3 – Третій слайд: «Огляд літератури (аналогів)»

# Постановка задачі

**Формулювання проблеми**
Визначення оптимальної системи журналювання подій для електронної комерції з точки зору продуктивності, ефективності зберігання та можливості горизонтального масштабування.

**Опис очікуваних результатів**

- Порівняння продуктивності ELK Stack, Grafana Loki та Splunk при різних навантаженнях.
- Визначення оптимальної системи для зберігання великих обсягів даних.
- Аналіз можливості та ефективності горизонтального масштабування систем.

Рисунок А.4 – Четвертий слайд: «Постановка задачі»

# Методологія

**Опис використаних методів дослідження**

- Вимірювання продуктивності в реальному часі.
- Оцінка ефективності зберігання даних.
- Аналіз можливості горизонтального масштабування.

**Інструментарій та технології, використані в роботі**

- Java Microbenchmark Harness для вимірювання продуктивності.
- Інструмент побудови Gradle, фреймворк Spring Boot.
- Docker для розгортання ELK Stack та Grafana Loki.
- Splunk Web застосунок.

Рисунок А.5 – П’ятий слайд: «Методологія»

# Архітектура система для проведення експериментального дослідження

**Схема архітектури розробленої системи**
Архітектура включає основний модуль (common), модулі elk, loki, splunk та налаштування запуску JMH тестів

**Опис ключових компонентів**

- Java застосунок для генерації та відправки подій.
- ELK Stack, Grafana Loki, Splunk для збереження та обробки подій.
- Docker для контейнеризації та розгортання систем.

Рисунок А.6 – Шостий слайд: «Архітектура система для проведення експериментального дослідження»

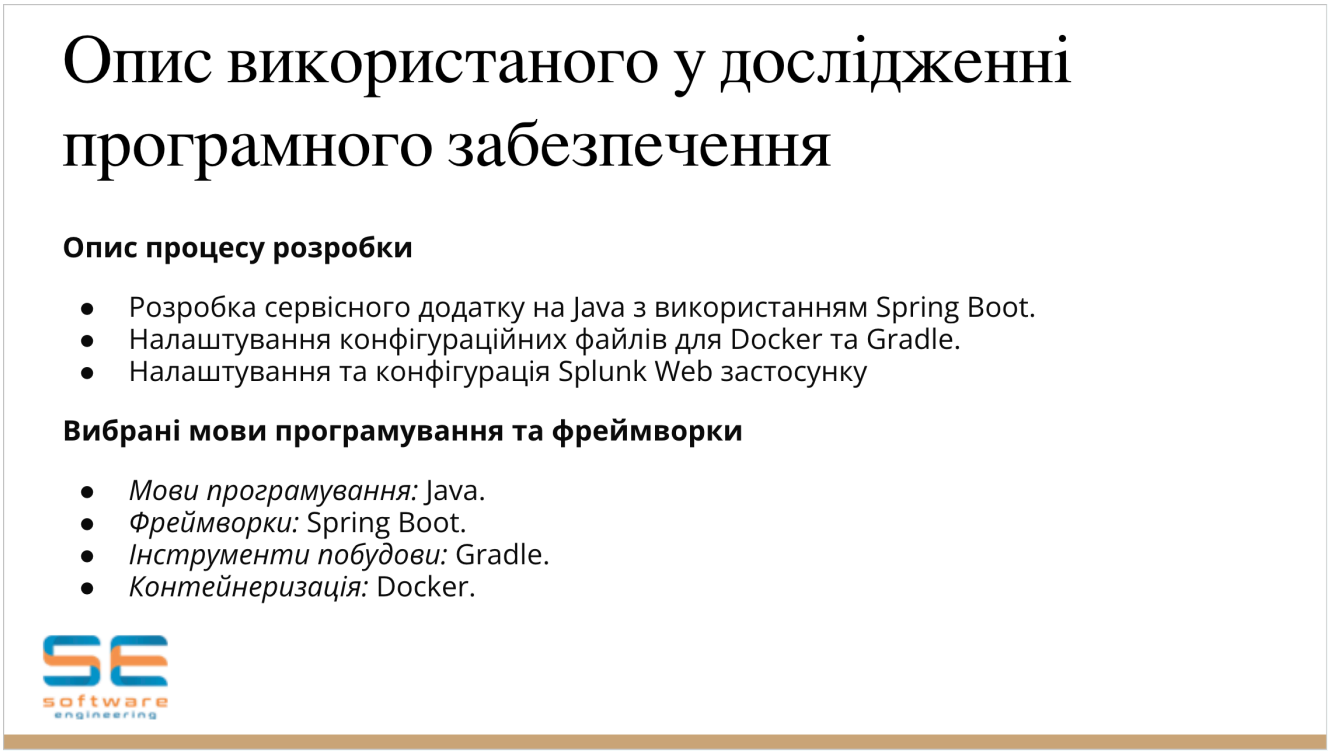



Рисунок А.7 – Сьомий слайд: «Опис використаного у дослідженні програмного забезпечення»

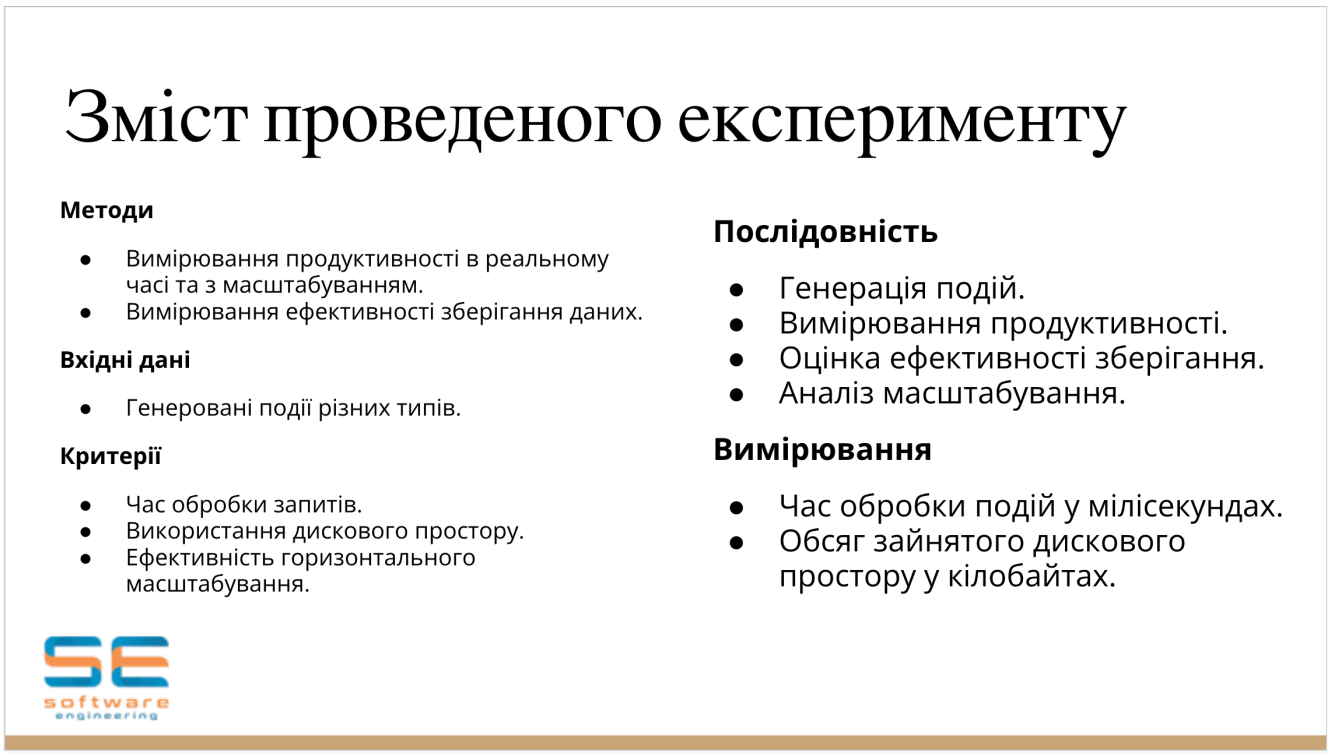



Рисунок А.8 – Восьмий слайд: «Зміст проведеного експерименту»

# Вимірювання продуктивності у реальному часі

| Кількість ітерацій | ELK Stack | Grafana Loki | Splunk |
|---|---|---|---|
| 100 | 1.274 ± 0.061 | 1.290 ± 0.035 | 1.336 ± 0.072 |
| 1000 | 1.260 ± 0.010 | 1.284 ± 0.021 | 1.294 ± 0.005 |
| 10000 | 1.294 ± 0.009 | 1.322 ± 0.012 | 1.333 ± 0.008 |
| 100000 | 1.321 ± 0.004 | 1.354 ± 0.005 | 1.380 ± 0.011 |

**(ms)**

Рисунок А.9 – Дев’ятий слайд: «Вимірювання продуктивності у реальному часі»

Вимірювання продуктивності у реальному часі

| Кількість ітерацій | ELK Stack | Grafana Loki | Splunk |
|---|---|---|---|
| 100 | 1.274 ± 0.061 | 1.290 ± 0.035 | 1.336 ± 0.072 |
| 1000 | 1.260 ± 0.010 | 1.284 ± 0.021 | 1.294 ± 0.005 |
| 10000 | 1.294 ± 0.009 | 1.322 ± 0.012 | 1.333 ± 0.008 |
| 100000 | 1.321 ± 0.004 | 1.354 ± 0.005 | 1.380 ± 0.011 |

**(ms)**

Вимірювання ефективності зберігання

| Кількість ітерацій | ELK Stack | Grafana Loki | Splunk |
|---|---|---|---|
| 100 | 4577 | 8 | 36 |
| 1000 | 10393 | 1328 | 56 |
| 10000 | 23613 | 2632 | 76 |
| 100000 | 67523 | 7860 | 108 |

**(Kb)**

Рисунок А.10 – Десятий слайд: «Вимірювання продуктивності та ефективності»

Вимірювання продуктивності з масштабуванням у реальному часі

| Кількість ітерацій | ELK Stack | Grafana Loki | Splunk |
|---|---|---|---|
| 100 | 1.216 ± 0.015 | 1.251 ± 0.009 | 1.284 ± 0.009 |
| 1000 | 1.234 ± 0.051 | 1.269 ± 0.025 | 1.304 ± 0.016 |
| 10000 | 1.282 ± 0.013 | 1.362 ± 0.029 | 1.405 ± 0.035 |
| 100000 | 1.395 ± 0.005 | 1.491 ± 0.010 | 1.522 ± 0.014 |

**(ms)**

Вимірювання ефективності зберігання з масштабуванням

| Кількість ітерацій | ELK Stack | Grafana Loki | Splunk |
|---|---|---|---|
| 100 | 34980 *(34.16Mb)* | 6568 *(6.4Mb)* | 52 |
| 1000 | 116336 *(113.61Mb)* | 15752 *(15.4Mb)* | 72 |
| 10000 | 169697 *(165.72Mb)* | 28852 *(28.2Mb)* | 88 |
| 100000 | 505630 *(493.78Mb)* | 97020 *(94.7Mb)* | 136 |

**(Kb)**

Рисунок А.11 – Одинадцятий слайд: «Вимірювання продуктивності та ефективності з масштабуванням»

# Аналіз отриманих результатів

**ELK Stack**

Показав найкращу продуктивність, що робить його підходящим для систем з високим навантаженням, де важлива швидкість обробки подій. Підтверджено ефективність його використання у високонавантажених системах.

**Grafana Loki**

Виявився збалансованим рішенням між продуктивністю та ефективністю зберігання даних. Це гарний вибір для середовищ, де потрібен баланс між продуктивністю і економією ресурсів, відкриваючи нові можливості для компромісу.

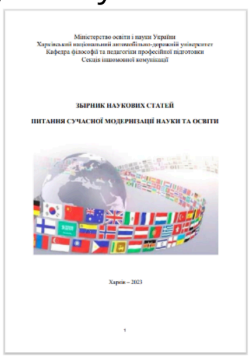
**Splunk**

Продемонстрував найефективніше використання дискового простору, що робить його ідеальним для систем з обмеженими ресурсами для зберігання. Виявлено нові можливості для оптимізації використання дискового простору у таких системах.

Рисунок А.12 – Дванадцятий слайд: «Аналіз отриманих результатів»

# АПРОБАЦІЯ ДОСЛІДЖЕННЯ

Матеріали дослідження були опубліковані у збірник з конференцій




ЗБІРНИК НАУКОВИХ СТАТЕЙ
ПИТАННЯ СУЧАСНОЇ МОДЕРНІЗАЦІЇ
НАУКИ ТА ОСВІТИ

Рисунок А.13 – Тринадцятий слайд: «Апробація дослідження»

# Підсумки

Проведене дослідження показало, що системи журналювання подій ELK Stack, Grafana Loki та Splunk мають різні переваги та недоліки залежно від потреб електронної комерції. ELK Stack виявився найефективнішим при великих обсягах запитів, роблячи його оптимальним вибором для систем з високим навантаженням. Splunk забезпечив ефективне використання дискового простору, що є критично важливим для середовищ з обмеженими ресурсами, тоді як Grafana Loki запропонував збалансоване рішення між продуктивністю та ефективністю зберігання даних.

Дослідження також вказало на необхідність подальшого аналізу інших сучасних систем журналювання, таких як Fluentd чи Graylog, для визначення їхньої ефективності та продуктивності. Розробка та тестування гібридних підходів, які поєднують переваги кількох систем журналювання, можуть допомогти досягти оптимальної продуктивності та ефективності. Крім того, оптимізація існуючих рішень з метою зменшення використання ресурсів, підвищення безпеки та поліпшення масштабованості є важливим напрямком для подальших досліджень.

Таким чином, отримані результати можуть бути використані підприємствами електронної комерції для покращення управління журналами, підвищення продуктивності та забезпечення надійності систем. Результати дослідження створюють міцну основу для подальших досліджень у галузі систем журналювання, сприяючи розвитку нових технологій та методів.

Рисунок А.14 – Чотирнадцятий слайд: «Підсумки»

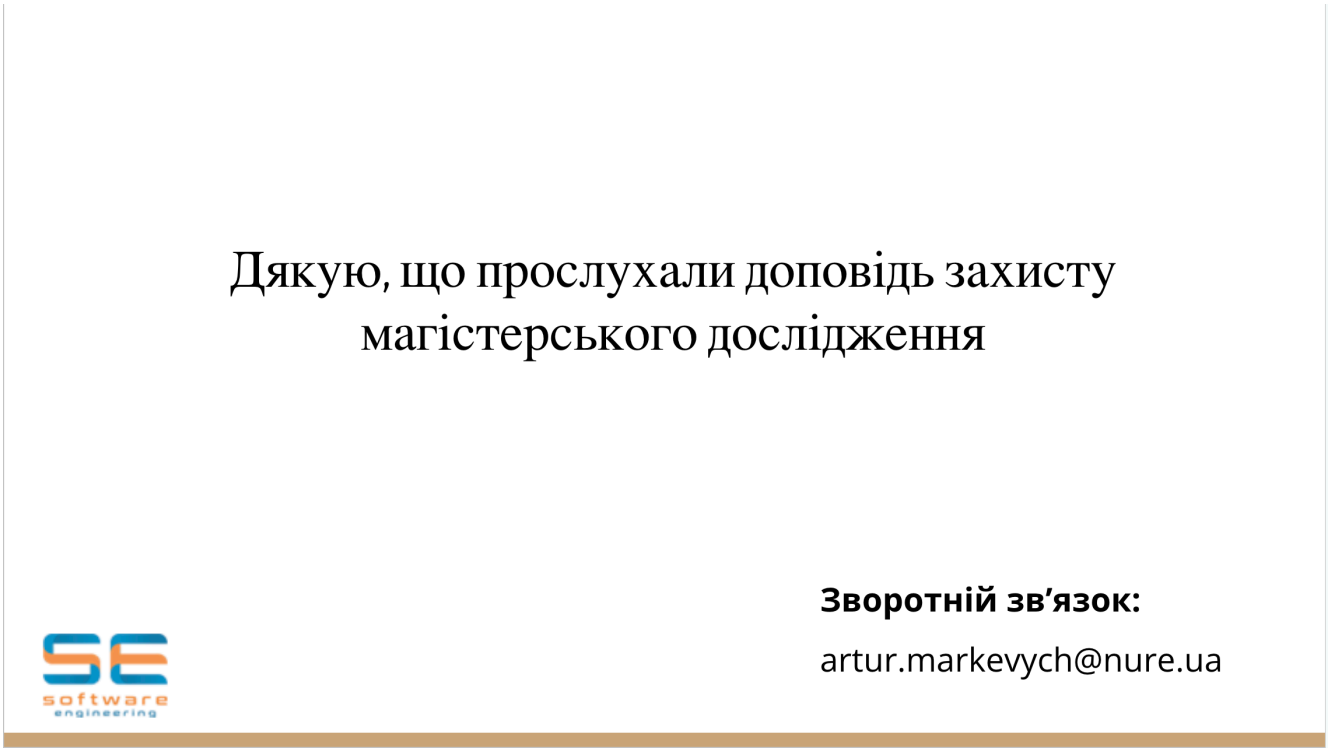



Рисунок А.15 – П’ятнадцятий слайд: «Зворотній зв’язок»

# Додаток Б

## Приклади програмного коду

```java
package ua.nure.research.performance;

import org.openjdk.jmh.annotations.*;
import org.openjdk.jmh.runner.Runner;
import org.openjdk.jmh.runner.RunnerException;
import org.openjdk.jmh.runner.options.Options;
import org.openjdk.jmh.runner.options.OptionsBuilder;

import ua.nure.research.runner.StaticCommonRunner;

import java.util.Random;
import java.util.concurrent.TimeUnit;

@BenchmarkMode(Mode.AverageTime)
@OutputTimeUnit(TimeUnit.MILLISECONDS)
@State(Scope.Benchmark)
public class LoggerBenchmark {

    private static final int ITERATIONS = 100_000;
    private static final int RANDOM_LIMIT = 10000;

    @Benchmark
    @Measurement(iterations = ITERATIONS, time = 1, timeUnit = TimeUnit.MILLISECONDS)
    public void measureLoggerPerformance() throws InterruptedException {
        StaticCommonRunner.createInfoLog(new Random().nextInt(RANDOM_LIMIT));
        Thread.sleep(1);
    }


    public static void main(String[] args) throws RunnerException {
        Options options = new OptionsBuilder()
                .include(LoggerBenchmark.class.getSimpleName())
                .forks(1)
                .build();

        new Runner(options).run();
    }

}
```

Рисунок Б.1 – Приклад коду Benchmark тестів

```java
package ua.nure.research.scaling;

import org.openjdk.jmh.annotations.Benchmark;
import org.openjdk.jmh.annotations.BenchmarkMode;
import org.openjdk.jmh.annotations.Level;
import org.openjdk.jmh.annotations.Measurement;
import org.openjdk.jmh.annotations.Mode;
import org.openjdk.jmh.annotations.OperationsPerInvocation;
import org.openjdk.jmh.annotations.OutputTimeUnit;
import org.openjdk.jmh.annotations.Param;
import org.openjdk.jmh.annotations.Scope;
import org.openjdk.jmh.annotations.State;
import org.openjdk.jmh.annotations.TearDown;
import org.openjdk.jmh.annotations.Threads;
import org.openjdk.jmh.runner.Runner;
import org.openjdk.jmh.runner.RunnerException;
import org.openjdk.jmh.runner.options.Options;
import org.openjdk.jmh.runner.options.OptionsBuilder;
import ua.nure.research.runner.StaticCommonRunner;

import java.util.Random;
import java.util.concurrent.TimeUnit;

@BenchmarkMode(Mode.AverageTime)
@OutputTimeUnit(TimeUnit.MILLISECONDS)
@State(Scope.Thread)
public class ScalingSimulatorBenchmark {

    private static final int THREAD_POOL_SIZE = 10;
    private static final int ITERATIONS = 100;
    private static final int RANDOM_LIMIT = 10000;

    @Benchmark
    @Threads(THREAD_POOL_SIZE)
    @Measurement(iterations = ITERATIONS, time = 5, timeUnit = TimeUnit.MINUTES)
    public void measureLoggerPerformance() throws InterruptedException {
        StaticCommonRunner.createInfoLog(new Random().nextInt(RANDOM_LIMIT));
        Thread.sleep(1);
    }

    public static void main(String[] args) throws RunnerException {
        Options options = new OptionsBuilder()
                .include(ScalingSimulatorBenchmark.class.getSimpleName())
                .forks(1)
                .build();
        new Runner(options).run();
    }

}

```

Рисунок Б.2 – Приклад коду багато потокових Benchmark тестів

```java
package ua.nure.research.scaling;

import org.openjdk.jmh.annotations.Benchmark;
import org.openjdk.jmh.annotations.BenchmarkMode;
import org.openjdk.jmh.annotations.Level;
import org.openjdk.jmh.annotations.Measurement;
import org.openjdk.jmh.annotations.Mode;
import org.openjdk.jmh.annotations.OperationsPerInvocation;
import org.openjdk.jmh.annotations.OutputTimeUnit;
import org.openjdk.jmh.annotations.Param;
import org.openjdk.jmh.annotations.Scope;
import org.openjdk.jmh.annotations.State;
import org.openjdk.jmh.annotations.TearDown;
import org.openjdk.jmh.annotations.Threads;
import org.openjdk.jmh.runner.Runner;
import org.openjdk.jmh.runner.RunnerException;
import org.openjdk.jmh.runner.options.Options;
import org.openjdk.jmh.runner.options.OptionsBuilder;
import ua.nure.research.runner.StaticCommonRunner;

import java.util.Random;
import java.util.concurrent.TimeUnit;

@BenchmarkMode(Mode.AverageTime)
@OutputTimeUnit(TimeUnit.MILLISECONDS)
@State(Scope.Thread)
public class ScalingSimulatorBenchmark {

    private static final int THREAD_POOL_SIZE = 10;
    private static final int ITERATIONS = 100;
    private static final int RANDOM_LIMIT = 10000;

    @Benchmark
    @Threads(THREAD_POOL_SIZE)
    @Measurement(iterations = ITERATIONS, time = 5, timeUnit = TimeUnit.MINUTES)
    public void measureLoggerPerformance() throws InterruptedException {
        StaticCommonRunner.createInfoLog(new Random().nextInt(RANDOM_LIMIT));
        Thread.sleep(1);
    }

    public static void main(String[] args) throws RunnerException {
        Options options = new OptionsBuilder()
                .include(ScalingSimulatorBenchmark.class.getSimpleName())
                .forks(1)
                .build();
        new Runner(options).run();
    }

}

```

Рисунок Б.2 – Приклад коду багато потокових Benchmark тестів

```java
package ua.nure.research.models;

import lombok.Data;
import lombok.ToString;

import java.util.Date;

@Data
@ToString
public class LogEvent {

    Date timestamp;
    String eventType;
    User user;
    Product product;

    public LogEvent(String eventType, User user, Product product) {
        this.timestamp = new Date();
        this.eventType = eventType;
        this.user = user;
        this.product = product;
    }

}

```

Рисунок Б.3 – Приклад коду моделі події журналювання

```java
package ua.nure.research.creator;

import ua.nure.research.models.LogEvent;
import ua.nure.research.models.Product;
import ua.nure.research.models.User;

public class LogCreatorService {

    public static LogEvent generateProductViewEvent(User user, Product product) {
        return new LogEvent("ProductView", user, product);
    }

    public static LogEvent generateAddToCartEvent(User user, Product product) {
        return new LogEvent("AddToCart", user, product);
    }

    public static LogEvent generatePurchaseEvent(User user, Product product) {
        return new LogEvent("Purchase", user, product);
    }

    public static LogEvent generatePaymentEvent(User user, Product product) {
        return new LogEvent("Payment", user, product);
    }

}
```

Рисунок Б.4 – Приклад коду генерації подій

```
plugins {
    id 'org.springframework.boot' version '3.3.0-SNAPSHOT'
    id 'io.spring.dependency-management' version '1.1.4'
    id 'java'
}

allprojects {
    java {
        sourceCompatibility = JavaVersion.VERSION_17
    }

    group = 'ua.nure'
    version = '0.0.1-SNAPSHOT'
}

subprojects {
    apply plugin: 'org.springframework.boot'
    apply plugin: 'io.spring.dependency-management'
    apply plugin: 'java'
}

bootJar {
    enabled = false
}

jar {
    enabled = true
}

group = 'ua.nure'
version = '0.0.1-SNAPSHOT'

repositories {
    mavenCentral()
    maven {
        url 'https://repo.spring.io/milestone'
    }
    maven {
        url 'https://repo.spring.io/snapshot'
    }
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter'
    compileOnly 'org.projectlombok:lombok'
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'

//    jmh
    implementation 'org.openjdk.jmh:jmh-core:1.37'
    annotationProcessor 'org.openjdk.jmh:jmh-generator-annprocess:1.37'
    testImplementation 'org.openjdk.jmh:jmh-runner:1.37'
}
```

Рисунок Б.5 – Приклад коду build.gradle common модуля

```
plugins {
    id 'org.springframework.boot' version '3.3.0-SNAPSHOT'
    id 'io.spring.dependency-management' version '1.1.4'
    id 'java'
}

repositories {
    mavenCentral()
    maven {
        url 'https://repo.spring.io/milestone'
    }
    maven {
        url 'https://repo.spring.io/snapshot'
    }
}

dependencies {
    implementation(project(":common"))
    implementation 'org.springframework.boot:spring-boot-starter'
    implementation group: 'org.springframework.boot', name: 'spring-boot-docker-compose', version: '3.2.3'
    compileOnly 'org.projectlombok:lombok'
    annotationProcessor 'org.projectlombok:lombok'

    // Logstash
    implementation 'net.logstash.logback:logstash-logback-encoder:6.6'

    //    jmh
    implementation 'org.openjdk.jmh:jmh-core:1.37'
    annotationProcessor 'org.openjdk.jmh:jmh-generator-annprocess:1.37'

    testImplementation 'org.springframework.boot:spring-boot-starter-test'
}

tasks.register('jmh', JavaExec) {
    group = 'benchmark'
    description = 'Runs the JMH benchmarks'

    mainClass = 'org.openjdk.jmh.Main'
    classpath = sourceSets.main.runtimeClasspath

    args = [
//            'ua.nure.research.performance.LoggerBenchmark',
            'ua.nure.research.scaling.ScalingSimulatorBenchmark',
            '-f', '1',
            '-wi', '5',
            '-jvmArgsPrepend', '-server',
            '-jvmArgs', '-Xms2G -Xmx2G'
    ]
}

```

Рисунок Б.7 – Приклад коду build.gradle elk модуля

```
plugins {
    id 'org.springframework.boot' version '3.3.0-SNAPSHOT'
    id 'io.spring.dependency-management' version '1.1.4'
    id 'java'
}

repositories {
    mavenCentral()
    maven {
        url 'https://repo.spring.io/milestone'
    }
    maven {
        url 'https://repo.spring.io/snapshot'
    }
}

dependencies {
    implementation(project(":common"))
    implementation 'org.springframework.boot:spring-boot-starter'
    implementation group: 'org.springframework.boot', name: 'spring-boot-docker-compose', version: '3.2.3'
    compileOnly 'org.projectlombok:lombok'
    annotationProcessor 'org.projectlombok:lombok'
    // Loki
    implementation group: 'com.github.loki4j', name: 'loki-logback-appender', version: '1.5.1'

    //    jmh
    implementation 'org.openjdk.jmh:jmh-core:1.37'
    annotationProcessor 'org.openjdk.jmh:jmh-generator-annprocess:1.37'

    testImplementation 'org.springframework.boot:spring-boot-starter-test'
}

tasks.register('jmh', JavaExec) {
    group = 'benchmark'
    description = 'Runs the JMH benchmarks'

    mainClass = 'org.openjdk.jmh.Main'
    classpath = sourceSets.main.runtimeClasspath

    args = [
            'ua.nure.research.performance.LoggerBenchmark',
//            'ua.nure.research.scaling.ScalingSimulatorBenchmark',
            '-f', '1',
            '-wi', '5',
            '-jvmArgsPrepend', '-server',
            '-jvmArgs', '-Xms2G -Xmx2G'
    ]
}

```

Рисунок Б.8 – Приклад коду build.gradle loki модуля

```
plugins {
    id 'org.springframework.boot' version '3.3.0-SNAPSHOT'
    id 'io.spring.dependency-management' version '1.1.4'
    id 'java'
}

repositories {
    mavenCentral()
    maven {
        url 'https://repo.spring.io/milestone'
    }
    maven {
        url 'https://repo.spring.io/snapshot'
    }
    maven {
        url 'https://splunk.jfrog.io/splunk/ext-releases-local'
        name 'Splunk Releases'
    }
}

dependencies {
    implementation(project(":common"))
    implementation 'org.springframework.boot:spring-boot-starter'
    compileOnly 'org.projectlombok:lombok'
    annotationProcessor 'org.projectlombok:lombok'
    // splunk
    implementation 'com.splunk.logging:splunk-library-javalogging:1.8.0'

//    jmh
    implementation 'org.openjdk.jmh:jmh-core:1.37'
    annotationProcessor 'org.openjdk.jmh:jmh-generator-annprocess:1.37'

    testImplementation 'org.springframework.boot:spring-boot-starter-test'
}

tasks.register('jmh', JavaExec) {
    group = 'benchmark'
    description = 'Runs the JMH benchmarks'

    mainClass = 'org.openjdk.jmh.Main'
    classpath = sourceSets.main.runtimeClasspath

    args = [
//            'ua.nure.research.performance.LoggerBenchmark',
            'ua.nure.research.scaling.ScalingSimulatorBenchmark',
            '-f', '1',
            '-jvmArgsPrepend', '-server',
            '-jvmArgs', '-Xms6G -Xmx6G'
    ]
}

```

Рисунок Б.9 – Приклад коду build.gradle splunk модуля

```java
package ua.nure.research.runner;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import ua.nure.research.creator.LogCreatorService;
import ua.nure.research.models.LogEvent;
import ua.nure.research.models.Product;
import ua.nure.research.models.User;

public class StaticCommonRunner {

    public static Logger log = LoggerFactory.getLogger(StaticCommonRunner.class);

    public static void createInfoLog(int index) {
        log.info(createLogEvent(index).toString());
    }

    private static LogEvent createLogEvent(int index) {
        if (index % 2 == 0) {
            return LogCreatorService.generateAddToCartEvent(createUser(index), createProduct(index));
        }
        if (index % 3 == 0) {
            return LogCreatorService.generatePurchaseEvent(createUser(index), createProduct(index));
        }
        if (index % 5 == 0) {
            return LogCreatorService.generatePaymentEvent(createUser(index), createProduct(index));
        }
        return LogCreatorService.generateProductViewEvent(createUser(index), createProduct(index));
    }

    private static Product createProduct(int index) {
        return Product.builder()
                .name("Product " + index)
                .price(90.15 + index)
                .build();
    }

    private static User createUser(int index) {
        return User.builder()
                .username("User [" + index + "]")
                .build();
    }
}

```

Рисунок Б.10 – Приклад коду генератору випадкових подій

Додаток В

Апробація результатів роботи

Міністерство освіти і науки України
Харківський національний автомобільно-дорожній університет
Кафедра філософії та педагогіки професійної підготовки
Секція іншомовної комунікації

**ЗБІРНИК НАУКОВИХ СТАТЕЙ**

**ПИТАННЯ СУЧАСНОЇ МОДЕРНІЗАЦІЇ НАУКИ ТА ОСВІТИ**

Харків – 2023

Рисунок В.1 – Збірник наукових статей «Питання сучасної модернізації науки та освіти»

**THE PROBLEM OF MAKING LOGS IN E-COMMERCE**

*A.V. Markevych, student*

*M. Suknov, PhD, Associate Professor*

*Kharkiv National University of Radio Electronics*

E-commerce has become an integral part of modern-day business, with more and more consumers opting to shop online. As the number of online transactions increases, so does the volume of data generated by e-commerce applications. Logging is a crucial aspect of e-commerce application development as it enables developers to record critical information, which is used for debugging, performance analysis, and security auditing. However, logging can be a challenging process, and improper logging practices can lead to severe consequences, including data breaches, system crashes, and poor system performance. In this article, the problem of making logs in e-commerce applications will be considered and provide some solutions to address this issue.



*The Problem*

E-commerce applications generate a huge amount of data, which needs to be logged to enable developers to identify issues, analyse performance, and troubleshoot.

Рисунок В.2 – Збірник наукових статей «Питання сучасної модернізації науки та освіти». Перша сторінка.

However, improper logging practices can lead to several problems, including performance degradation, security risks, difficulties in debugging, and increased storage costs.

Performance degradation is a common problem with logging, especially when logging is done excessively or inefficiently. Logging can impact the performance of the e-commerce application, causing delays in processing user requests, which can result in a poor user experience. To minimize this impact, developers should use efficient logging mechanisms, such as asynchronous logging, which enables the application to continue processing user requests while logging is performed in the background.

Security risks are also a concern when logging, especially when logs are not managed correctly. Improperly managed logs can lead to security vulnerabilities, such as the exposure of sensitive data or the ability for attackers to use logs to gain unauthorized access to the system. To prevent this, developers should implement secure logging mechanisms, such as encrypting log data and limiting access to logs to authorized personnel.

Difficulties in debugging can arise when the logging process is not well-designed. Developers may find it challenging to identify the source of problems when errors occur, leading to delays in resolving issues. To overcome this, developers should determine the appropriate level of logging required for the application and use efficient logging mechanisms.

Рисунок В.3 – Збірник наукових статей «Питання сучасної модернізації науки та освіти». Друга сторінка.

Increased storage costs can be a significant challenge for smaller e-commerce businesses. The volume of data generated by logging can be significant, leading to increased storage costs. To reduce storage costs, developers should implement log rotation, which involves archiving and deleting logs after a specified time period.

*Solutions to the Problem*

To address the problem of making logs in e-commerce applications, developers should adopt best practices, such as determining the appropriate level of logging, using



efficient and secure logging mechanisms, and implementing log rotation. These best practices help minimize the impact on system performance, prevent security vulnerabilities, and reduce storage costs.

Determining the appropriate level of logging is crucial to minimize the impact on system performance. Developers should identify the types of data that need to be logged, such as user interactions, system events, and error messages.

Рисунок В. – Збірник наукових статей «Питання сучасної модернізації науки та освіти». Третя сторінка.

This helps developers to focus on logging only the relevant information required to identify and resolve issues. Using efficient and secure logging mechanisms is also essential to prevent security vulnerabilities and minimize the impact on system performance. Developers should use efficient logging mechanisms, such as asynchronous logging, which enables the application to continue processing user requests while logging is performed in the background. Secure logging mechanisms, such as encrypting log data and limiting access to logs to authorized personnel, help to prevent unauthorized access to log data.

Implementing log rotation is crucial to minimize storage costs and ensure that only relevant data is retained. Log rotation involves archiving and deleting logs after a specified time period. This helps to prevent the accumulation of outdated data and reduces the storage requirements for log data.

*Conclusion*

Logging is a critical aspect of e-commerce application development, and developers must adopt best practices to ensure that logging is done efficiently and securely. By using efficient and secure logging mechanisms, developers can minimize the impact on system performance, prevent security vulnerabilities, and reduce storage costs. By implementing best practices, such as determining the appropriate level of logging, using efficient and secure logging mechanisms, and implementing log rotation, developers can ensure that logging is performed effectively and efficiently, helping to improve the overall quality and security of e-commerce applications.

Рисунок В.5 – Збірник наукових статей «Питання сучасної модернізації науки та освіти». Четверта сторінка.

References

1. R. Kurkovsky, "E-commerce Systems: Architecture, Design, and Implementation," Boca Raton, FL: CRC Press, 2018.

156

2. J. B. Rainsberger, "Logging Done Right," IEEE Software, vol. 31, no. 5, pp. 94-96, Sept.-Oct. 2014, DOI: 10.1109/MS.2014.118.

3. M. J. Franklin, M. J. Carey, and M. J. Zwilling, "Performance of Asynchronous Logging," ACM SIGMOD Record, vol. 28, no. 2, pp. 1-5, June 1999, DOI: 10.1145/314403.314406.

4. S. T. Chapman and S. J. Chapman, "Security Log Management: Identifying Patterns in the Chaos," Sebastopol, CA: O'Reilly Media, 2006.

Рисунок В.6 – Збірник наукових статей «Питання сучасної модернізації науки та освіти». Шоста сторінка.

Додаток Г

Звіт результатів Перевірки на унікальність тексту в базі ХНУРЕ

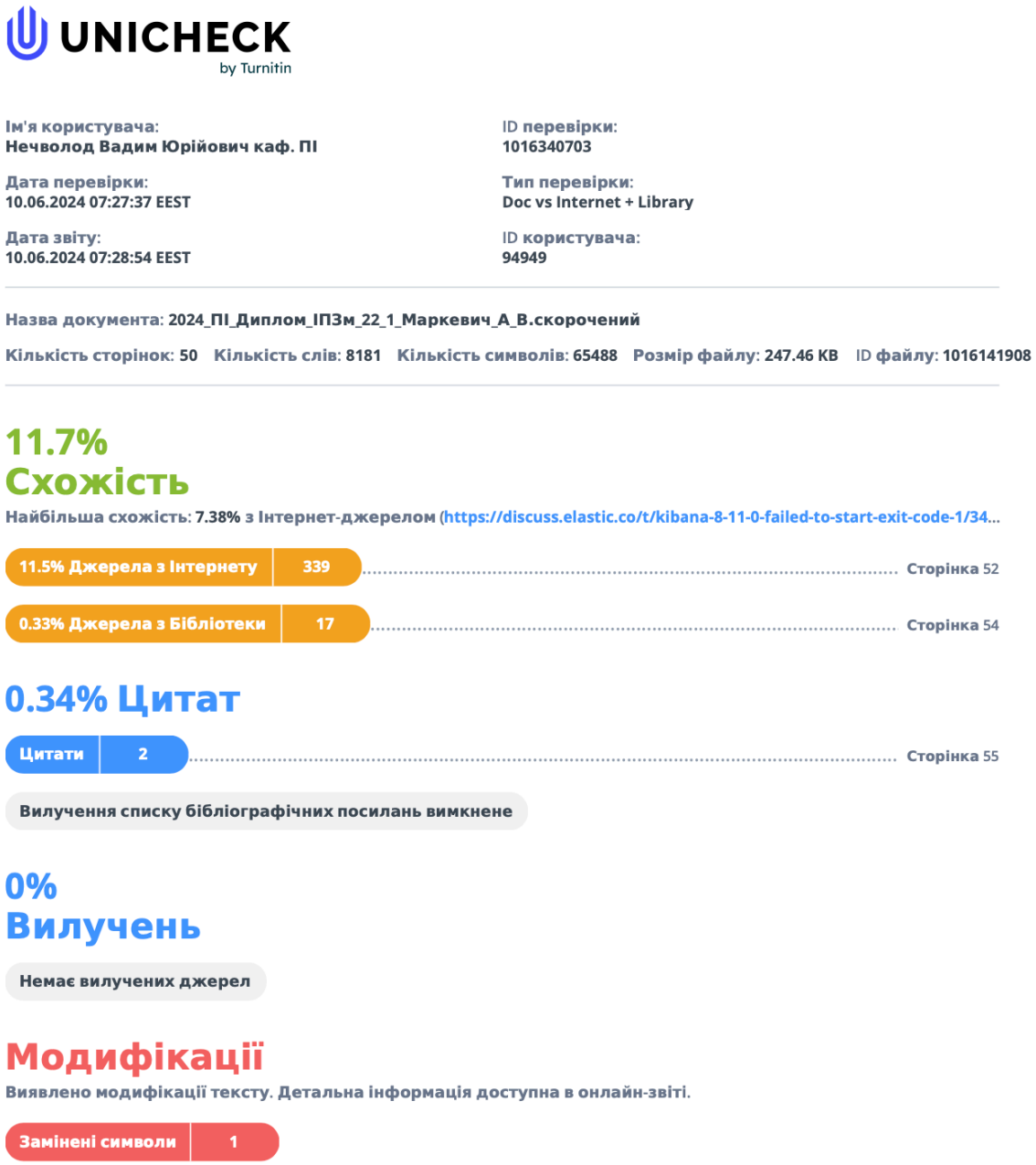



Рисунок Г.1 – Титульний аркуш звіту результатів Перевірки на унікальність тексту в базі ХНУРЕ

# Додаток Д

## Експертний Висновок результатів Перевірки кваліфікаційної роботи на відповідність оформлення Вимоги ДСТУ 3008:2015

Експертний висновок результатів перевірки кваліфікаційної роботи

студент (посада) | програмної інженерії (кафедра) | ІПЗм-22-1 (група)

Маркевич А.В.
( прізвище, ім'я, по батькові )

Зауваження

| Пункт ДСТУ 3008-2015 | Зміст пункту | Сторінка кваліфікаційної роботи |
|---|---|---|
| 1 | 2 | 3 |
| | **7.1 Загальні положення** | |
| | **7.3 Нумерація сторінок звіту** | |
| | **7.5 Рисунки** | |
| | **7.6 Таблиці** | |
| | **7.7 Переліки** | |
| | **7.8 Примітки** | |
| | **7.9 Виноски** | |
| | **7.10 Формули та рівняння** | |
| | **7.11 Посилання** | |
| | **7.13 Список авторів** | |
| | **7.14 Скорочення та умовні познаки** | |
| | **7.15 Додатки** | |

Експерт _____ (підпис) Вадим НЕЧВОЛОД (прізвище, ініціали)

18.06.2024

Рисунок Д.1 – Експертний висновок результатів перевірки кваліфікаційної роботи