Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій

Кафедра програмного забезпечення



**ЗВІТ**

**Про виконання лабораторної роботи № 5**

**«Створення звіту виконання автотестів для швидкого аналізу якості програмного забезпечення»**

**з дисципліни «Управління якістю ПЗ»**

**Лектор:**

асист. каф. ПЗ

Ваврук І.Є.

**Виконав:**

студ. групи ПЗІП-11

Бойчук Т.В.

**Прийняла:**

асист. каф. ПЗ

Ваврук І.Є.

«\_\_\_» \_\_\_\_\_\_ 2020 р.

∑ = \_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

Львів – 2020

**ТЕМА РОБОТИ**: Створення звіту виконання автотестів для швидкого аналізу якості програмного забезпечення.

**МЕТА РОБОТИ**: Навчитися створювати звіти про виконання автотестів для швидкого аналізу якості програмного забезпечення.

**ЗАВДАННЯ**

1. За допомогою Allure report згенерувати звіти автоматизованого тестування для тестування REST-сервісу (лабораторна №5).
2. Побудувати статистику отриманих результатів.
3. Обчислити метрики автоматизації.
4. Сформувати звіт.

### РЕЗУЛЬТАТИ ВИКОНАННЯ

Для встановлення та використання Allure Report потрібно у файлі pom.xml додати наступні конфігурації:

*<?*xml version="1.0" encoding="UTF-8"*?>*<project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
  
 <groupId>com.miller.dev</groupId>  
 <artifactId>QA\_lab3</artifactId>  
 <version>1.0-SNAPSHOT</version>  
  
 <properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 <aspectj.version>1.9.6</aspectj.version>  
 </properties>  
  
 <dependencies>  
 *<!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->* <dependency>  
 <groupId>org.projectlombok</groupId>  
 <artifactId>lombok</artifactId>  
 <version>1.18.10</version>  
 <scope>provided</scope>  
 </dependency>  
  
 <dependency>  
 <groupId>org.seleniumhq.selenium</groupId>  
 <artifactId>selenium-java</artifactId>  
 <version>3.141.59</version>  
 </dependency>  
 <dependency>  
 <groupId>org.testng</groupId>  
 <artifactId>testng</artifactId>  
 <version>7.0.0</version>  
 <scope>test</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.jacoco</groupId>  
 <artifactId>jacoco-maven-plugin</artifactId>  
 <version>0.8.4</version>  
 </dependency>  
 <dependency>  
 <groupId>org.aspectj</groupId>  
 <artifactId>aspectjweaver</artifactId>  
 <version>1.9.6</version>  
 <scope>runtime</scope>  
 </dependency>  
 <dependency>  
 <groupId>io.github.bonigarcia</groupId>  
 <artifactId>webdrivermanager</artifactId>  
 <version>3.8.1</version>  
 <scope>test</scope>  
 </dependency>  
 *<!-- https://mvnrepository.com/artifact/io.qameta.allure/allure-testng -->* <dependency>  
 <groupId>io.qameta.allure</groupId>  
 <artifactId>allure-testng</artifactId>  
 <version>2.13.6</version>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <configuration>  
 <source>1.8</source>  
 <target>1.8</target>  
 </configuration>  
 </plugin>  
  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-surefire-plugin</artifactId>  
 <version>2.20</version>  
 <configuration>  
 <suiteXmlFiles>  
 <suiteXmlFile>suites/tests.xml</suiteXmlFile>  
 </suiteXmlFiles>  
 <argLine>  
 -javaagent:"${settings.localRepository}/org/aspectj/aspectjweaver/${aspectj.version}/aspectjweaver-${aspectj.version}.jar"  
 </argLine>  
 </configuration>  
 <dependencies>  
 <dependency>  
 <groupId>org.aspectj</groupId>  
 <artifactId>aspectjweaver</artifactId>  
 <version>${aspectj.version}</version>  
 </dependency>  
 </dependencies>  
 </plugin>  
  
 <plugin>  
 <groupId>org.jacoco</groupId>  
 <artifactId>jacoco-maven-plugin</artifactId>  
 <version>0.8.4</version>  
 <executions>  
 <execution>  
 <goals>  
 <goal>prepare-agent</goal>  
 </goals>  
 </execution>  
 <execution>  
 <id>report</id>  
 <phase>test</phase>  
 <goals>  
 <goal>report</goal>  
 </goals>  
 </execution>  
 </executions>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

Тепер ми можемо використовувати анотації для специфікації тестів та їхніх кроків. Я використовував в своїй роботі наступні анотації:

@Description

@Epic

@Story

@Feature

Також я створив клас Allurelistener який імплементує клас ITestListener для того, щоб виконувати певні сценарії при різних результатах тестування, зокрема для здійснення скріншотів тестів які не аройшли.

**Allurelistener.java:**

public class Allurelistener implements ITestListener {  
 private static String getTestMethodName (ITestResult iTestResunt) {  
 return iTestResunt.getMethod().getConstructorOrMethod().getName();  
 }  
  
 @Attachment  
 public byte[] saveFailureScreenShot(WebDriver driver){  
 return ((TakesScreenshot)driver).getScreenshotAs(OutputType.*BYTES*);  
 }  
  
 @Attachment(value="{0}", type = "text/plain")  
 public static String saveTextLog(String message) {  
 return message;  
 }  
  
*/\* @Override  
 public void onStart(ITestContext iTestContext) {  
 System.out.println("I am in onStart method "+iTestContext.getName());  
 iTestContext.setAttribute("WebDriver", BaseClass.);  
 }\*/* @Override  
 public void onStart(ITestContext iTestContext) {  
 System.*out*.println("I am in onStart method "+iTestContext.getName());  
 iTestContext.setAttribute("WebDriver", BaseClass.*getDriver*());  
 }  
  
 @Override  
 public void onTestSuccess(ITestResult iTestResult) {  
 System.*out*.println("I am in onTestSuccess method "+*getTestMethodName*(iTestResult)+" success");  
 }  
  
 @Override  
 public void onTestFailure(ITestResult iTestResult) {  
 System.*out*.println("I am in onTestFailure method "+*getTestMethodName*(iTestResult)+" faild");  
  
 Object testClass = iTestResult.getInstance();  
 WebDriver driver = BaseClass.*getDriver*();  
  
 if (driver instanceof WebDriver) {  
 System.*out*.println("Screenshot captured for test case:"+*getTestMethodName*((iTestResult)));  
 saveFailureScreenShot(driver);  
 }  
 *saveTextLog*(*getTestMethodName*(iTestResult)+" faild and screenshot taken!");  
 }  
}

За допомогою анотації @Listeners ({Allurelistener.class}) я підключаю створений клас до класу тестування.

Був створений BaseClass який міститиме сутність класу WebDriver і дасть можливість використовувати її класам Allurelistener та TestGmail.

**BaseClass.java**

public class BaseClass {  
 public WebDriver driver;  
 public static ThreadLocal<WebDriver> *tdriver* = new ThreadLocal<WebDriver>();  
  
 public WebDriver initialize\_driver() {  
 WebDriverManager.*chromedriver*().setup();  
 driver = new ChromeDriver();  
 driver.manage().timeouts().pageLoadTimeout(20, TimeUnit.*SECONDS*);  
 *tdriver*.set(driver);  
 return *getDriver*();  
 }  
  
 public static synchronized WebDriver getDriver () {  
 return *tdriver*.get();  
 }  
}

Після виконання тестів в кориневій папці проекту з’явиться папка ./allure-results, після чого ми можемо викликати команду для відображення репорту - **allure serve allure-results.**

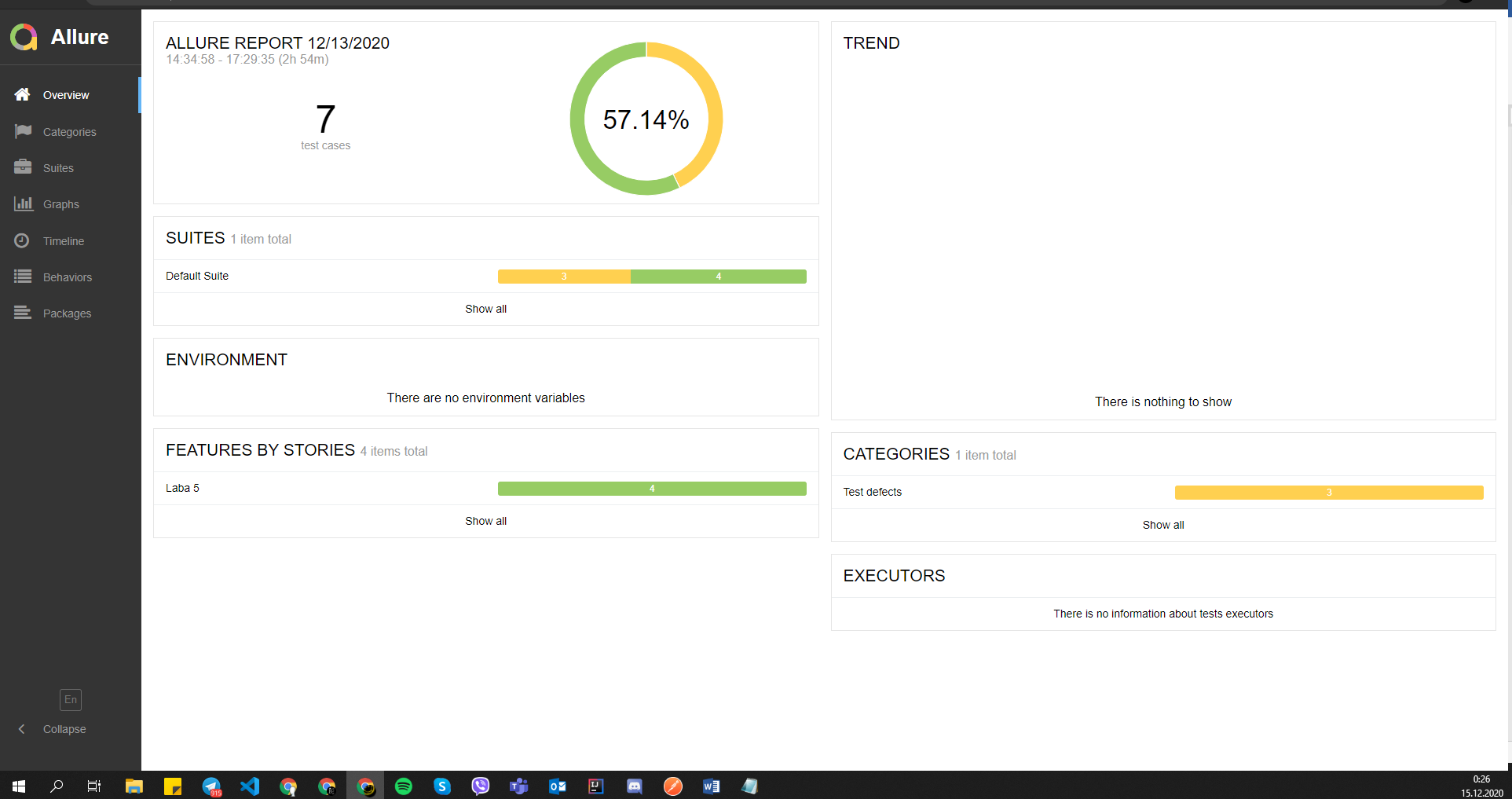


Рис. 1. Головна сторінка звіту про тестування

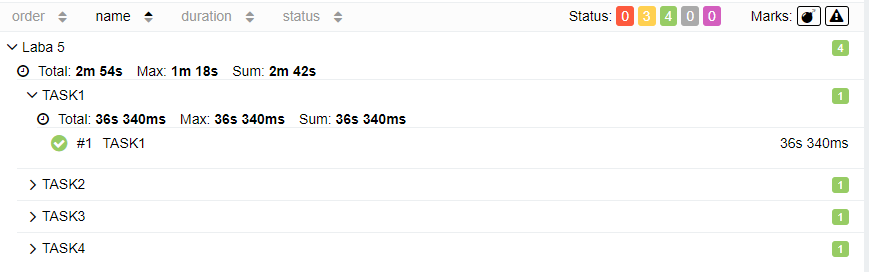


Рис. 2. Сторінка інформації про тестові випадки та їх результати

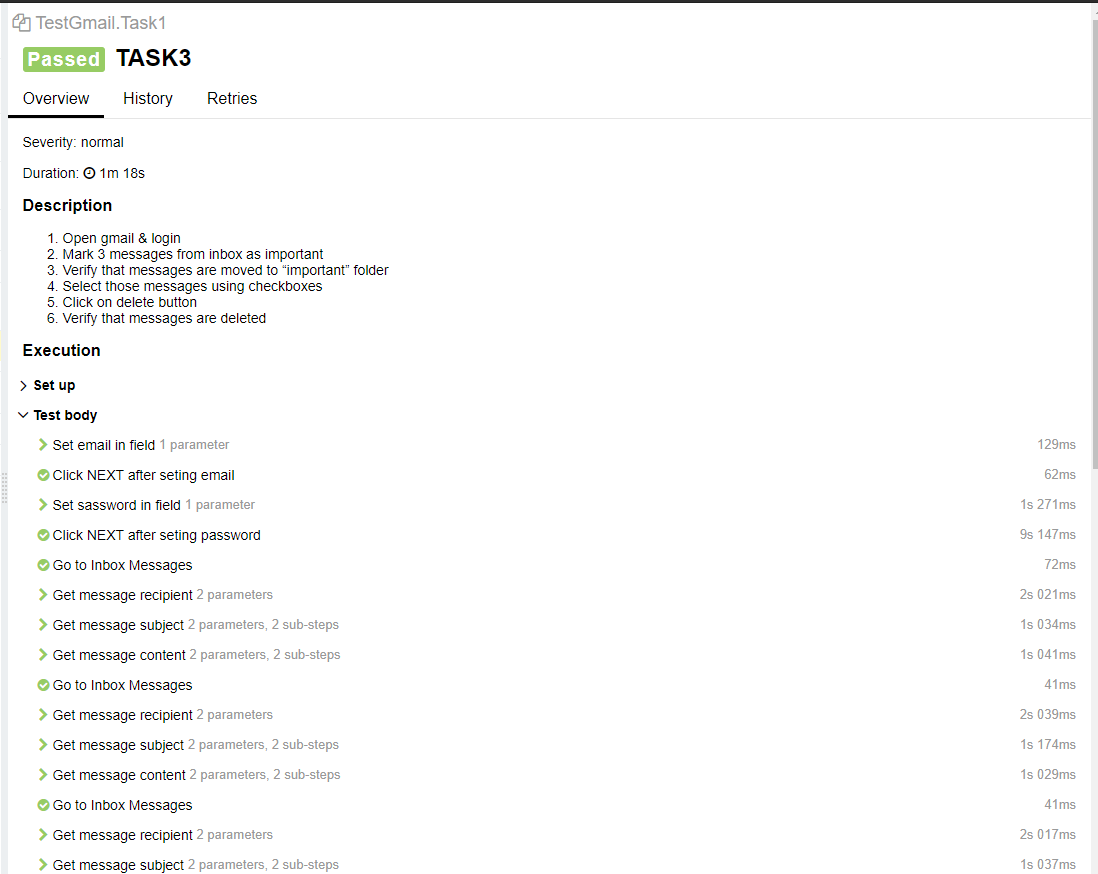


Рис. 3. Детальна інформація про тест

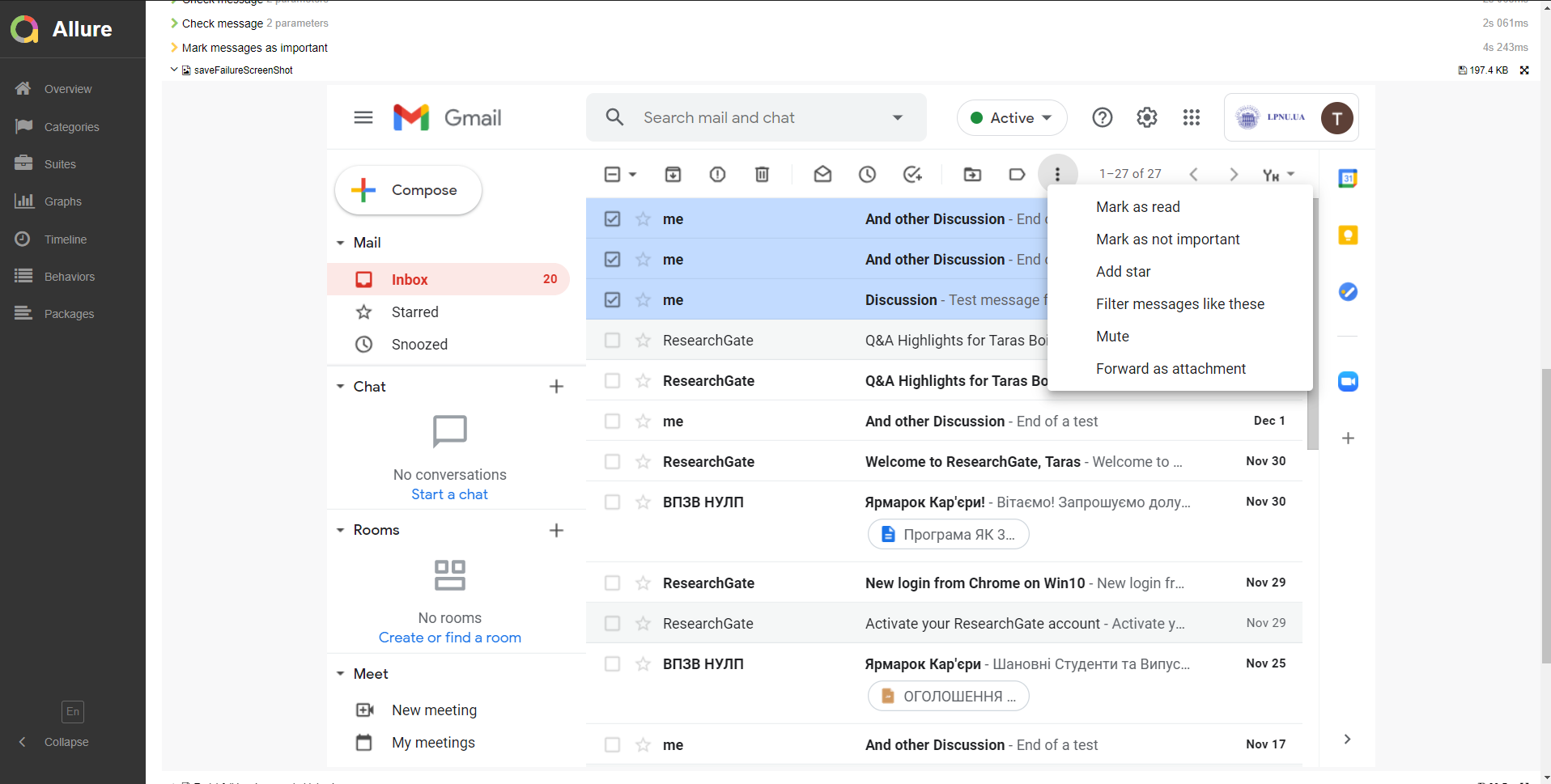


Рис. 4. Скріншот зроблений під час невдалого виконання тесту

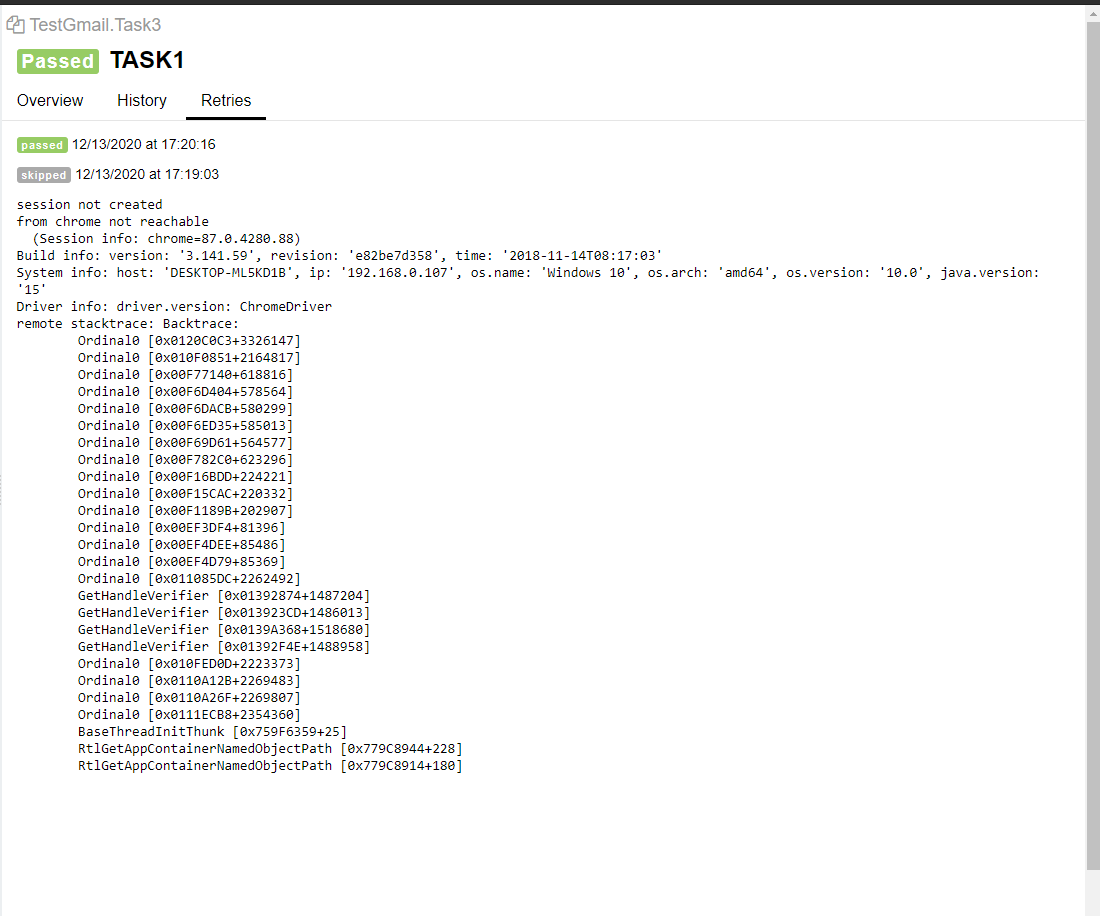


Рис. 4. Історія виконання тестів

### ВИСНОВКИ

На даній лабораторній роботі я навчився створювати звіти про виконання автотестів для швидкого аналізу якості програмного забезпечення, з використанням Allure Report.