



LLM-QA

Yazılım Kalite Güvencesi ve Testi



Literatür Taraması Görev Ataması

Ekip üyeleri kendisine bir hedef test seçti ve bu testler için güncel teknikleri öğrenmek adına literatür taraması gerçekleştirdi.

MrAliBulut opened last month · edited by MrAliBulut

Edits ⋮

Task Description

This week's objective is to conduct a formal literature review on your chosen test topic. This research is the first step for both your own learning and our project's "Related Work" section.

Action Items

- 1. Conduct Review: Find and deeply read a minimum of **5 high-quality sources** (academic papers, key blog posts, book chapters) related to this topic.
- 2. Document Analysis: Use the [report_template.md](#) to log your analysis for all 5+ sources. Pay close attention to the "Personal Analysis & Relevance to LLM-QA" section.
- 3. Submit Report:
 - Name your completed file *exactly* as specified in the template: `your-test_your-name_literature-review.md` (e.g., `Performance_Kubra_LR.md`).
 - Upload your file to the `Reports` folder within the `Firat-Kalite-Güvencesi/Management` repository.

Task1 Github Issue Ataması Kesiti



Literatür Taraması Rapor Şablonu

- Her üye seçtiği konu için 5 kaynak araştırdı.
- Yaptıkları işleri raporlamaları adına kendilerine bir rapor şablonu verildi.
- Şablonda raporun nasıl hazırlanacağı, nereye ne halde yükleneceği açıklandı ve özellikle buldukları kaynakları alıntılamaları istendi.

3. Appendix: How to Cite (A Brief Guide)

You must cite your resources in your work. This is non-negotiable for academic work.

What is Citing and Why Do We Do It?

Citation is the practice of giving credit to the sources you use. In research, failing to cite is considered **plagiarism**, which is the theft of intellectual property and the most severe academic offense.

We cite for two reasons:

1. **To give credit:** To acknowledge the work of the researchers who came before us.
2. **To provide a trail:** To allow your reader (me, or the reviewers of our paper) to find the exact source you used and verify your claims.

Two Types of Citation:

We are using both in articles.

1. **In-Text Citation:** A brief note *inside* your sentence to show where the idea came from.

- **Example:** The "Defect-to-Test" model is superior because it provides a "permanent safety net" (Your Team's Abstract, 2025).
- **Example:** Smith (2021) argues that mutation testing is too slow for CI/CD pipelines.

2. **Full Citation (Bibliography/Reference List):** The full details of the source, listed at the end of your document. This is what allows the reader to find the source.

How to Format a Full Citation (APA 7th Edition)

Use these templates based on the source type.

Template 1: Journal/Conference Article Author, A. A., Author, B. B., & Author, C. C. (Year). Title of the article: Subtitle if any. *Title of the Journal or Conference Proceedings, Volume(Issue), pages. DOI (Digital Object Identifier)*

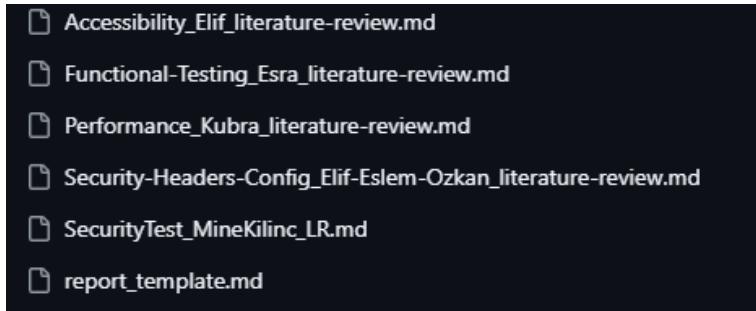
Example: De Millo, R. A., Lipton, R. J., & Perlis, A. J. (1979). Social processes and proofs of theorems and programs. *Communications of the ACM*, 22(5), 271–280. <https://doi.org/10.1145/359104.359106>

Task1 Rapor Şablonu Alıntı Yönlendirme Kısmı



Literatür Taraması Sonuç

- Üyeler raporlarını şablonda belirtildiği üzere github repomuza yüklediler.
- Üyelerin raporları incelendi ve şablonu izleme oranları not edildi.



Raporlar GitHub Repomuzda

2. Adherence to Task Instructions

- **Assigned Topic:** Security Testing
- **Submitted Topic:** Literature Review for LLM in security testing
- **File Structure:** Perfect but be more careful about headers.
- **Citings:** Perfect Citing
- **File Naming Correct?** Yes
- **File Location Correct?** Yes
- Instructions are followed perfectly.

Rapor İnceleme Notları (Mine)



Literatür Taraması Sonuç

- Üyelerin araştırma konuları hakkında geleceğe dair işe yararlılığı hakkında not düşüldü.
- Her üye için verilmesi gereken feedback not alındı ve verildi. Geliştirilmesi gereken noktalar vurgulandı.

4. Review

- **Extractable Insights (for Group Synthesis Report):**
 - First 3 articles are great insights about llms in performance testing. Last 2 are more focused on LLMs in testing.
- **Private Feedback (for Individual):**
 - Be carefull with the structure. Everything is in order but appendix needed to be removed.
 - Excellent job about the research. The topics you've provided are very strong. First 3 articles are the exact output I expected when i gave you the task. Splendid work. Though article 4 and article 5 strays of your topics they are still usefull and will be used in the future of our project.

Rapor İnceleme Notları (Kübra)



Manuel Uygulama Görev Ataması

Üyeler halihazırda üzerine literatür taraması yaptıkları test konularını manuel olarak denedi.

- Böylece ürettikleri sistemin neden gerektiğini deneyimlediler.
- Ayrıca dersimizin gereksinimi olan test konularında hands-on deneyim elde ettiler.

MrAliBulut opened last month · edited by MrAliBulut

1. Objective: From Theory to Practice

In [Literature Review 1](#), we built the *theoretical* foundation for our assigned testing methods. We documented what they are, why they matter, and what the research says about them.

Now, we move from theory to *practice*. The objective of this task is to *experience the implementation process firsthand*.

This "process data"—your challenges, frustrations, and "a-ha" moments—is the primary evidence we will use to argue *why* a tool like LLM-QA is necessary.

2. Why This Documentation Is Critical

Your final report will not be about "success" or "failure" but about documenting the *process* itself.

Your [Implementation Log](#) is the *raw data* for our paper's "Methodology" section. When we claim that existing methods are [e.g., "confusing," "time-consuming," "difficult for non-experts"], your logs will be the *evidence* to support that claim.

3. Deliverables

- **Primary Deliverable:** A completed [Implementation Log](#) for your task.
 - **Template Location:** [Implementation Report TEMPLATE](#)
 - **Submission:** Upload your file to the [Reports/Experimental-Implementation](#) folder within the [Firat-Kalite-Givencesi/Management](#) repository.
- **Secondary Deliverable:** Any code you wrote to complete the task (provide link in your report).

Task2 GitHub Issue Atması Kesiti



Manuel Uygulama Görev Ataması

- Her üyenin yapacağı testin gerekliliğini görebileceği uygun bir hedef ortam belirlendi ve üyeler özel olarak görev atandı.
- Üyeler çalışmalarını kendi github hesaplarına bir repo olarak açıklayıcı bir biçimde depoladılar.

3. Code & Artifact Publication:

Your "defect-proving test" code is a critical artifact. Please create a new, public **GitHub repository**. This way you also keep a proof of your work.

You will link to this repository in the `## 4. Final Artifacts` section of your report. This ensures we have a clean, verifiable copy of the "golden data" you generated.

Task2 Kişisel Repo Talebi

MrAliBulut opened last month · edited by MrAliBulut

Edits · ...

1. Security Headers Config Test Experiment

The Target Project:

You will apply your test to the [OWASP Juice Shop](#), our standardized project for these experiments. It is a large application *intentionally* full of security misconfigurations.

- Repository: <https://github.com/owasp/juice-shop>
- Action: You will need to clone, install, and run this project locally to perform your tests. You will be testing the HTTP responses from the local server.

Your Goal:

Your goal is to write an automated test that *proves* a specific security header misconfiguration exists in the Juice Shop application.

The process is up to you, but we recommend these steps:

1. Explore the application's network traffic (using browser DevTools) or use an online scanner to identify a clear security header violation (e.g., missing `Content-Security-Policy`, weak `X-Frame-Options`, or missing `Strict-Transport-Security`).
2. Write an automated test script that makes an HTTP request to the application's homepage (or any page).
3. Inspect the response headers and assert that your test *fails* because a required header is missing or has an insecure value, based on security best practices.

This failing test is your "proof of error" and will serve as a "golden standard" for what we want our LLM to be able to generate.

2. 📚 Resources

Use these resources to guide your implementation. You are not expected to be an expert; you are expected to be a beginner *documenting your journey*.

- Key Standard: [OWASP Secure Headers Project](#) (This is the official guide on what security headers are and what their correct values should be. Use this to identify a violation).
- Reference: [Your own literature review](#).

3. 📈 Reporting

- Deliverable: As you work, your primary task is to fill out the [Implementation Log](#). This is the *only* deliverable for this issue.
 - Template: [Implementation Report TEMPLATE](#)
- Support: This task is *designed* to be challenging. If you struggle, Document this 'struggles' in detail. This data is *more valuable* than a quick success.

MrAliBulut opened last month · edited by MrAliBulut

Edits · ...

1. Performance Test Experiment

The Target Project:

You will apply your test to "The Internet" by Herokuapp, our standardized project for this experiment. It is a stable, publicly-hosted application designed for automation practice.

- Application URL: <https://the-internet.herokuapp.com>
- Action: You do not need to install anything. Your test script will run directly against the live URL.

Your Goal:

Your goal is to write an automated test that *proves* a specific performance profile for a page on "The Internet" application.

The process is up to you, but we recommend these steps:

1. Explore the application and identify a clear performance-intensive endpoint. A good candidate is the [Large & Deep DOM](#) page, which is *intentionally* large.
2. Write an automated test script using a load testing tool that runs against the URL you selected.
3. Define a performance budget (e.g., "p95 response time must be under 1.5s" or "error rate must be 0%") and assert that your test *fails* this budget under a specific load (e.g., 5 virtual users for 30 seconds).

This failing test (or a test that *proves* a specific, slow response) is your "proof of error" and will serve as a "golden standard" for what we want our LLM to be able to generate.

2. 📚 Resources

Use these resources to guide your implementation. You are not expected to be an expert; you are expected to be a beginner *documenting your journey*.

- Key Tool: [k6 \(by Grafana\)](#) (This is a modern, developer-friendly load testing tool. Their "Get Started" guide is excellent for this task).
- Reference: [Your own literature review](#).

3. 📈 Reporting

- Deliverable: As you work, your primary task is to fill out the [Implementation Log](#). This is the *only* deliverable for this issue.
 - Template: [Implementation Report TEMPLATE](#)
- Support: This task is *designed* to be challenging. If you struggle, Document this 'struggles' in detail. This data is *more valuable* than a quick success.

Task2 Özel GitHub Issue Ataması (Eslem)

Task2 Özel GitHub Issue Ataması (Kübra)



Manuel Uygulama Görev Ataması

Ayrıca üyelerden test süreçleri boyunca yaşadıkları sıkıntıları not almaları istendi.

- Böylece inşa edeceğimiz sistemin neden gerektiği sorusuna cevaplar toplandı.

Biggest "Pain Point" / "Struggle": [Describe the single most frustrating or confusing part of this task. Example: "(Domain Knowledge): I didn't know what a 'good' security header was. I had to spend 40 minutes on the OWASP site *before* I could even start to write the test."]

Task2 Zorlanma Noktaları Not Tutma Talebi

Biggest "Pain Point" / "Struggle": Deciding which tool (such as Axe-core or Cypress) to use to generate error evidence with an automated test and carrying out the integration of these tools into the Node.js project is the most challenging part with my current knowledge. The phase of writing the automated test code is the most expertise-demanding and challenging part of this task.

Örnek Zorlanma Noktası Notu (Elif)

3. Nihai Başarısızlık (FAIL) Kanıtı

Testin 30 saniye sonunda FAIL olarak sonuçlandı, p95 bütçesinin aşıldığı ve hata mesajının göründüğü kritik terminal çıktıları.

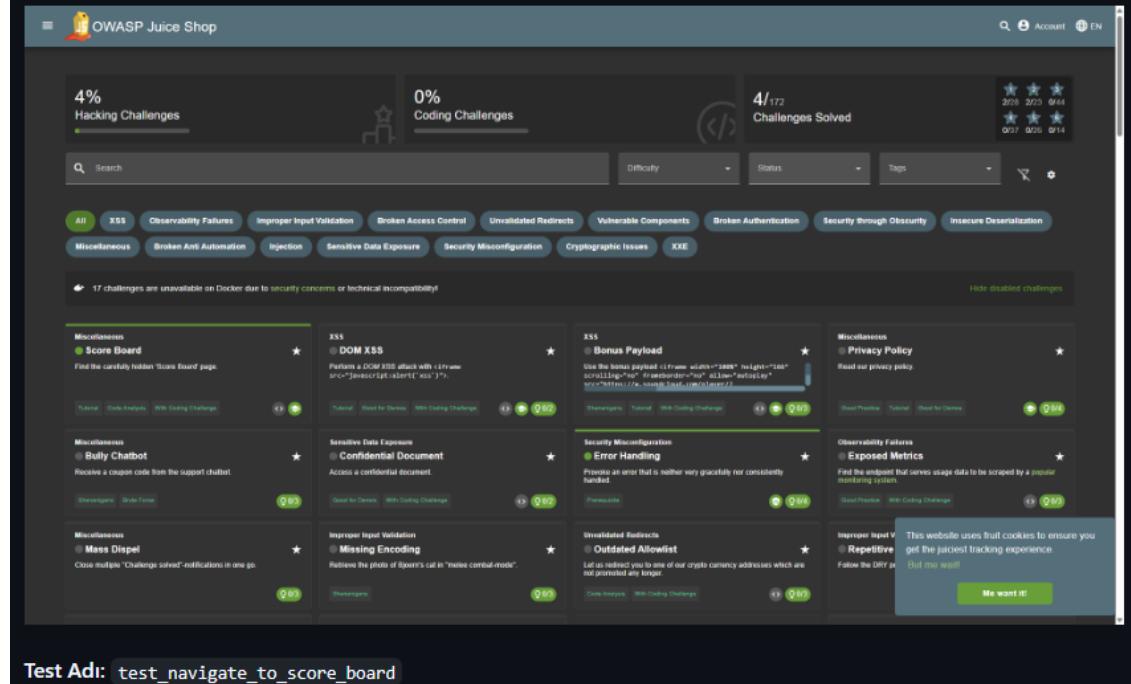
```
PS C:\Users\msi> cd C:\Users\msi\Desktop\k6Test
PS C:\Users\msi\Desktop\k6Test> k6 run performance_test.js

  Grafana
    execution: local
      script: performance_test.js
      output: -  
  
scenarios: (100.00%) 1 scenario, 5 max VUs, 1m0s max duration (incl. graceful stop):
      * default: 5 looping VUs for 30s (gracefulStop: 30s)  
  
| running (0m15.3s), 5/5 VUs, 43 complete and 0 interrupted iterations
default [=====>-----] 5 VUs 15.3s/30s  
  
THRESHOLDS
  http_req_duration
  X 'p(95)<500' p(95)=667.13ms
  http_req_failed
  ✓ 'rate<0.01' rate=0.00%
```

Task2 Sonuç Reposu README Kesiti (Kübra)

Navigation Testi - Score Board Sayfasına Yönlendirme

Aşağıdaki resim, "Score Board sayfasına yönlendirme" testinin çalıştırıldığı bir örneği göstermektedir:



The screenshot shows the OWASP Juice Shop application's main interface. At the top, there are navigation links for 'Home', 'About', 'Contact', 'Logout', and 'EN'. Below the header, there are two progress bars: 'Hacking Challenges' at 4% and 'Coding Challenges' at 0%. A central search bar has 'Search' placeholder text. To the right, there are filters for 'Difficulty' (dropdown set to 'All'), 'Status' (dropdown set to 'All'), and 'Tags' (checkboxes for 'XSS', 'SQL Injection', etc.). Below these filters, there is a note: '17 challenges are unavailable on Docker due to security concerns or technical incompatibility.' The main content area displays a grid of challenges categorized by type (Miscellaneous, XSS, DOM XSS, etc.) and sub-type (Score Board, DOM XSS, Bonus Payload, etc.). Each challenge card includes a title, a brief description, a difficulty rating (star icon), and a 'Solve' button with a progress bar indicating completion status (e.g., 98%, 99%, 100%). Some cards also show a 'Report' button. On the far right, there is a sidebar with a message about cookies and a 'Me want it!' button.

Test Adı: test_navigate_to_score_board

Task2 Sonuç Reposu README Kesiti (Esra)



Makale Durumu

Üyelerin ilk görevlerinde gerçekleştirdikleri literatür taramasının senteziyle makalemizin literatür taraması metni oluşturulmuş, Özeti ve Giriş metinleri değişikliğe uğramıştır.

Üyelerin ikinci görev ile ilgilendikleri sırada gerçekleşen işlem sonucu makalemizin metinleri metodoloji kısmına kadar hazır durumdadır ve metodoloji kısmının yazılmasıyla ufak değişikliklere uğrayabilir.

Ekibimizin güncel odağı projeyi tam anlamıyla gerçekleştirmektir.

<https://github.com/Firat-Kalite-Guvencesi/Management/blob/main/Paper/Paper.md>

Ekip Arkadaşlarım: **ve** **Dinleyicilerime** **Teşekkür Ederim.**

Mine KILINÇ
Elif ATAŞ
Kübra SOYSAL
Elif Eslem ÖZKAN
Esra Nur GÜMÜŞ