Verification & validation - PhishGuard

Authors: Roko Mladinić, Daniella Namuli, Simeon Markov & Zed Minabowan

Table of contents

Introduction

In this document, we'll be talking about the verification & validation of our PhishGuard project. Before we get into the document, we want to explain what our definition of verification & validation is.

- **Verification** = us testing whether what we built works the way we planned it in the analysis, planning and design.
- **Validation** = others testing our website and telling us whether they experience it the way we intended it to be built.

We'll start off by talking about what we think our website will do and what we found out during our personal tests (verification). After that we'll summarize what others thought after testing our website (validation).

Verification

The goal of our verification is to check whether the 'must have' user stories from our analysis document have been implemented correctly. Only the user stories that were marked as MUST HAVE in the MoSCoW table are included in this section. The verification is done by us testing the application and checking if the functionality behaves as we described it in our user stories.

User role	User story	Verification method	Passed
EMP-01	As an employee, I want to practice identifying phishing emails in a simulated inbox, so I can learn to spot red flags without risking actual company data.	Open the game and play through multiple email scenarios	YES
EMP-03	As a new hire, I want to receive instant feedback when I make a mistake, so I can immediately understand which visual cues I missed.	Make correct and incorrect choices during gameplay	YES
US-01	As a user, I want to be able to log-in, in order to track my score and win-streak.	Log in, play a round, log out and log in again	YES

Validation of functional requirements

This section focuses on validating whether the functional requirements that we made in our analysis document have been achieved. We'll also find out whether these requirements meet the goals of the project and the needs of our users. The validation is done from a user perspective and checks if our project actually helps with learning how to recognize phishing emails. The goal is not to test whether the system works technically, but to see whether our final product is useful and meaningful for our target audience.

Practicing

- Requirement:
    - Users should be able to actively practice spotting phishing emails, links and other red flags.
- Validation:
    - This requirement has been met. PhishGuard allows users to make use of a simulated inbox where they will be shown different email scenarios. Once a game starts, users get 30 seconds to decide whether an email is safe or not. This setup allows users to practice recognizing phishing indicators in a safe environment without any real risk. The practicing aspect directly supports the learning goal of this project.

Feedback system

- Requirement:
    - Users should receive feedback on their actions, including what they did wrong and which visual clues they missed.
- Validation:
    - The feedback system requirement has been met. After each decision, users receive immediate feedback explaining whether their choice was correct or incorrect. Feedback messages are short, focus on only a few issues at a time and are written in simple & language that's easy to understand. This helps users understand their mistakes without feeling overwhelmed or demotivated. The feedback also explains what users should look out for next time, which supports learning and improvement.

Result

- Requirement:
    - Users should be able to view their results, such as overall score and win-streaks.
- Validation:
    - This requirement has been met. After completing a game session, users can view their performance through results like their total score and current win-streak. These results give users insight into how well they perform and help to improve them during future sessions.

Administration

- Requirement:
    - Administrators should be able to view data analytics, such as which tasks users struggle with the most.
- Validation:
    - This requirement has been met. PhishGuard offers an admin dashboard that shows analytical data about user performance (Different users struggle with different attack types). This allows administrators to see patterns in user mistakes, identify easy/difficult phishing topics (attack types/categories) and admin has insight into which areas need more attention. These analytics support the educational goal of the project by making user behavior measurable. This allows the admins to make changes based on real user performance. To visualize this data, we have used the Pareto chart.

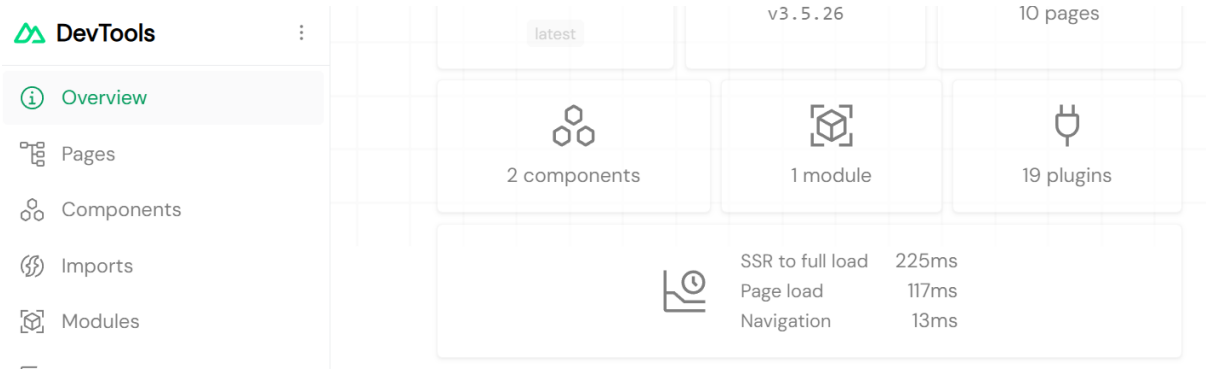
Validation of non-functional requirements

This section focuses on validating the non-functional requirements of PhishGuard. These requirements were also made in our analysis document and they are about the quality of our final product. We'll be talking about the project's usability, performance, clarity and security. The validation checks whether the application provides a smooth and understandable experience for non-technical users within the scope of this project.

Performance

- Requirement:
    - The game should respond quickly to user actions without noticeable delays.

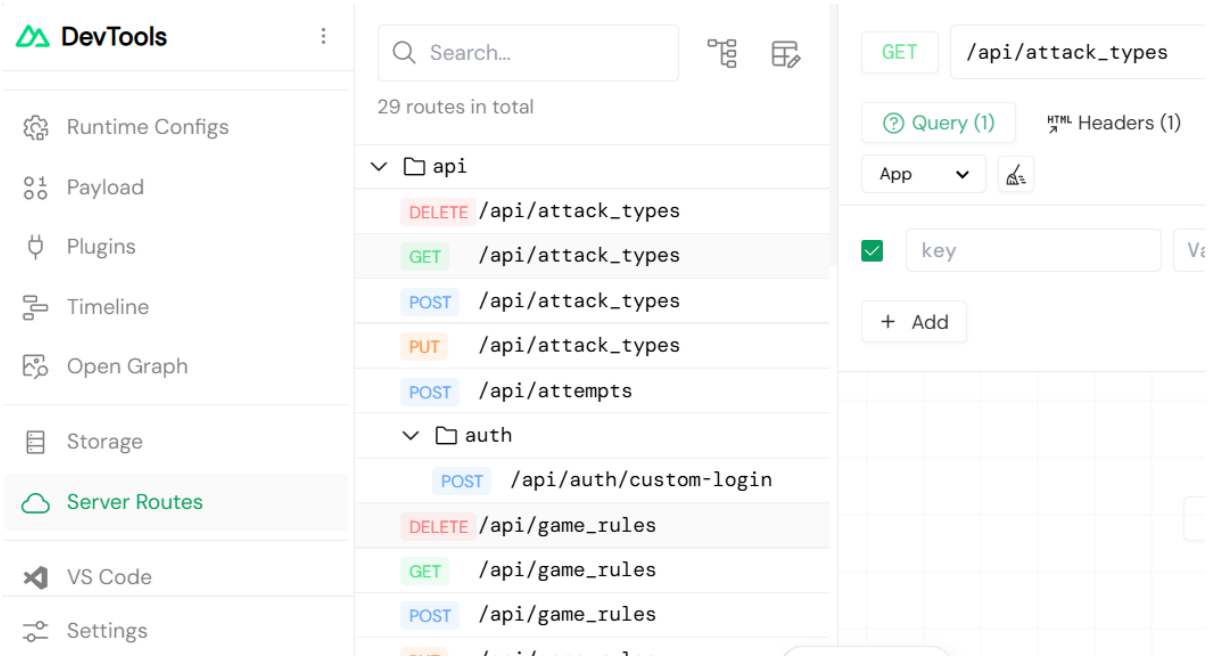
- Validation:
    - The performance requirement has been met in this project. Actions like clicking buttons, receiving feedback and starting games happen instantly during testing. The current setup provides smooth user interaction.



The image above is a snapshot of the dev panel, which serves as monitoring across the app. On the bottom right corner an overview of the performance can be seen, where it is apparent that each page loads relatively fast (avg. of 117 milliseconds).

Scalability

- Requirement:
    - The project should allow future expansion without major rework.
- Validation:
    - The project follows a file-based structure that comes with the framework, which means as the structure grows, the framework adjusts to that structure. It makes use of reusable components, middleware for guarding (authenticate) routes.



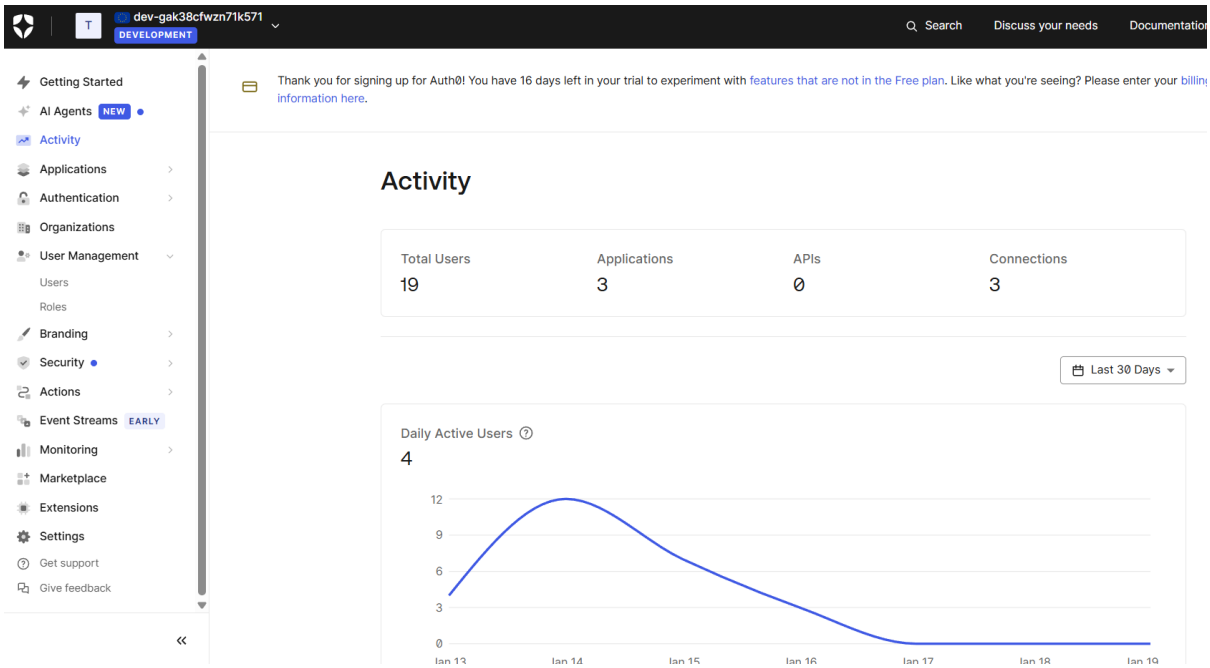
The image above is a screenshot of Devtools. The DevTool monitoring tool helps with tracking how the application grows over time by providing insights into performance, configured routes, modules imported, assets preview, http requests history, etc.

Usability

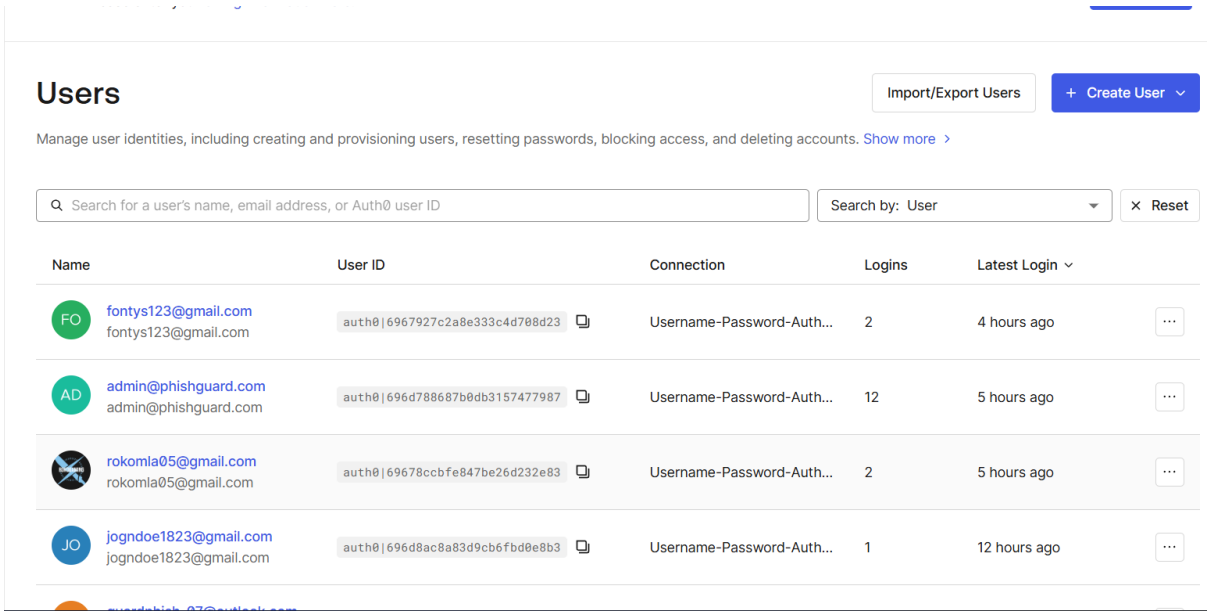
- Requirement:
    - The application should be easy to use for non-technical users.
- Validation:
    - The usability requirement has been met. Our interface is simple and focuses on essential elements of our project. Buttons are very clearly labeled, screens are not packed with unnecessary things and the game flow is easy to understand. Users can start playing without needing technical knowledge and that fits our target audience. Styling libraries like TailwindCSS & HeadlessUI keep the UI responsive to different screen sizes.

Security

- Requirement:
    - Validation for common vulnerability attacks, password hashed, CSRF tokenization.
- Validation:
    - The authentication is being handled by an external authentication provider, which comes with all the security like prevention of writing SQL queries and abusing http requests.



The image above is a screenshot of the activity tab of Auth0 (authentication provider), where an overview of total registered users, active users etc. could be seen.



The external provider also comes with user management like previewing registered users from the application, creating test users and a log (when a user registered/logged in).

Conclusion

The verification and validation of PhishGuard show that the core features of our project have been implemented correctly and that they meet the requirements we made in our analysis document. The verification confirms that all must have user stories function as intended, while the validation shows that both functional and non-functional requirements support the learning goal of the project. Overall, we believe that PhishGuard is a usable and effective way for non-technical users to practice recognizing phishing emails.