Hochschule der Medien

Project Documentation of PhishCoach

Jonas Koch

41489

jk249

Steffen Schawjinski

41524

ss533

Table of Contents

Table of Figures

1. PhishCoach Introduction

PhishCoach is a web application designed to educate users on identifying phishing emails. This interactive tool was developed as part of a software project, primarily for educational use. The motivation behind PhishCoach lies in addressing one of the most common yet effective cybercrime tactics, which is phishing.

Phishing emails pose a significant threat to online security. They are deceptive messages designed to trick recipients into revealing sensitive information, such as passwords, credit card numbers or other personal data. The problem with these emails is that they often appear to come from legitimate sources, such as banks, online retailers, or even friends and colleagues, making it difficult for people to distinguish them from genuine correspondence. This can lead to devastating consequences, such as identity theft, financial loss or unauthorized access to confidential information. Phishing attacks are widespread and yield high success rates, making them a preferred method among cybercriminals (Petrosyan, 2023).

Empowering users to identify these malicious emails can drastically reduce their vulnerability to such threats.

In this document, detailed insights into the creation of PhishCoach are provided, from ideation and concept creation to the final product. The documentation begins with an overview of the thought process, decisions, and the conceptual foundation of PhishCoach. This is then followed by a deep dive into the technical aspects, including frontend and backend functionalities, icons, the Bootstrap framework and how Django has been utilized. Several specific features such as link previews, email displays, button functionalities, solutions and the process of email generation are included. This is complemented by the backend, including the Django setup and the database structure. The received feedback is shared and problems are identified during the validation phase, concluding with a review and potential use cases of PhishCoach.

While navigating through this document, an in-depth understanding of PhishCoach, its purpose, design, and workings is provided.

2. Ideation & Decision

In the early stages of the project, the type of project was first identified. Whether it be a program, an app or a website, the decision had to be unanimous. After some discussions, the available skills and experiences were exchanged and in this regard Django, a high-level Python Web framework, was chosen for developing a web application.

In the brainstorming phase a long list of potential project ideas was created. Some of these were dismissed as they had already been done by others in similar forms. Afterwards it was decided to settle on PhishCoach, a unique idea in the realm of IT security, an area that was less explored and of increasing importance.

Initially, one of the ideas was to add a feature where it would send out a phishing email with a link. This link would direct the “victim” to the PhishCoach web application trainer. However, in this case the university would need to be asked for permission to proceed with this. Therefore, this idea had been discarded because it was regarded as excessive with regards to a software project. Furthermore, this approach would force everyone to participate without giving them a choice. Additionally, real links can still be traced, which could pose a security risk if a link wasn't properly updated. After dismissing this idea, the focus lied on PhishCoach becoming a web application for the software project, focusing on educating users about phishing emails.

3. Concept

This section of the document delves into the initial stages of the PhishCoach project, focusing on the foundational concepts that shaped the direction of the software. Here, key topics such as the Minimum Viable Product (MVP), the gamified learning approach, the team communication strategies and the project setup will be discussed.

3.1. Minimum Viable Product

The project's starting point involved defining the Minimum Viable Product (MVP), the most basic version of the product that can still convey the main idea and serve its primary purpose. For PhishCoach, the MVP was identified as a user-friendly web application capable of randomly displaying various emails to the users. Its core feature should revolve around the ability for users to determine the authenticity of each email, providing them with two options: "Authentic" and "Phishing".

Following the user's selection, the application should then provide a detailed explanation as to why the presented email was classified as phishing or authentic. This feedback loop should serve to reinforce the learning process, allowing users to understand the reasoning behind each classification.

Moreover, the goal was to design the web application with scalability and adaptability in mind. Therefore, the system should offer a straightforward way of creating and adding new emails. This ensures that the application can remain up-to-date and continually offer fresh, relevant content. By focusing on these key aspects, the MVP should effectively capture the fundamental

essence of PhishCoach, laying a robust foundation for future advancements and refinements.

3.2. Gameplay Idea

The idea behind the gameplay of PhishCoach is rooted in the concept of gamification, the addition of game elements in non-game contexts. The purpose here is to transform the learning process from the simple everyday task of differentiating between authentic and phishing emails into an engaging and interactive experience. In this system, users can see their progress through a score count that reflects the number of emails they have accurately judged as either authentic or phishing. This not only keeps users informed about their progress but also adds an element of challenge and achievement, encouraging them to improve their skills. In order to speed up the game loop only 5 out of 10 emails are served at a time. This could be easily changed in the code though.

3.3. Mockups

This mockup served as a stable reference point throughout the project development. It underwent minimal alterations, with the only notable change being the location of the email solution.

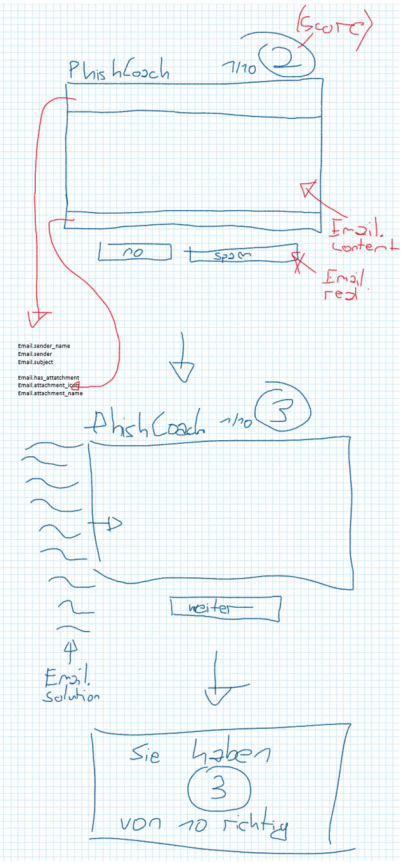


Figure 1: PhishCoach mockup

Moreover, the mockup functioned as more than just a visual guide. Django variables were incorporated into the mockup as a reminder for the communication between the frontend and the database.

3.4. Project Initialization

The set up of the PhishCoach project started with creating a Git repository in GitLab. This GitLab repository acts as a central place where all the project's files and changes are stored. After setting up the repository, the next step involved establishing the necessary files for the Django project.

Implementing Django, a high-level Python web framework, helped in developing a clean and pragmatic design for the web application. With the Django files in place, the foundation for the PhishCoach project was set, ready for further development.

The next significant step in setting up the project was integrating Bootstrap. Instead of linking to the Bootstrap library externally, which could lead to connectivity issues if the Bootstrap link fails at any point, the library was installed locally in the repository. This strategy ensured that Bootstrap was always available for use.

3.5. Communication

Successful communication is a necessity of any collaborative project. Despite being a team of two, ensuring effective communication was vital. It involved organizing and sharing all relevant information and tasks, ensuring both team members were on the same page throughout the project's development.

One of the primary tools utilized was GitLab. All the necessary files related to the project were shared via GitLab, and the repository's readme file was updated continuously with technical information.

In addition to GitLab, a shared OneDrive folder was used for storing non-code files. For instance, Photoshop files used to create the PhishCoach logo were stored here.

For sharing drawings, mock-ups, and miscellaneous notes, OneNote served as a shared notebook. This digital notebook allowed for easy sharing and editing of ideas and information in a structured format.

To stay organized and ensure that tasks were completed in a timely manner, Google Keep was employed for to-do tracking. Both technical and organizational tasks were listed here, keeping track of what needed to be done and ensuring nothing was overlooked. In the code, a "Todo" comment was also used as a reminder of elements that still required attention.

Lastly, for immediate and direct communication, the team resorted to WhatsApp and Discord. WhatsApp was primarily used for quick, less formal arrangements, while Discord served as the communication medium during active project development. These platforms allowed for easy real-time communication, ensuring both team members could discuss and solve issues as they arose.

4. Frontend

The frontend of the PhishCoach web application plays a crucial role in providing users with an engaging and easy-to-use interface. This part of the document covers the various components that contribute to the frontend's effectiveness.

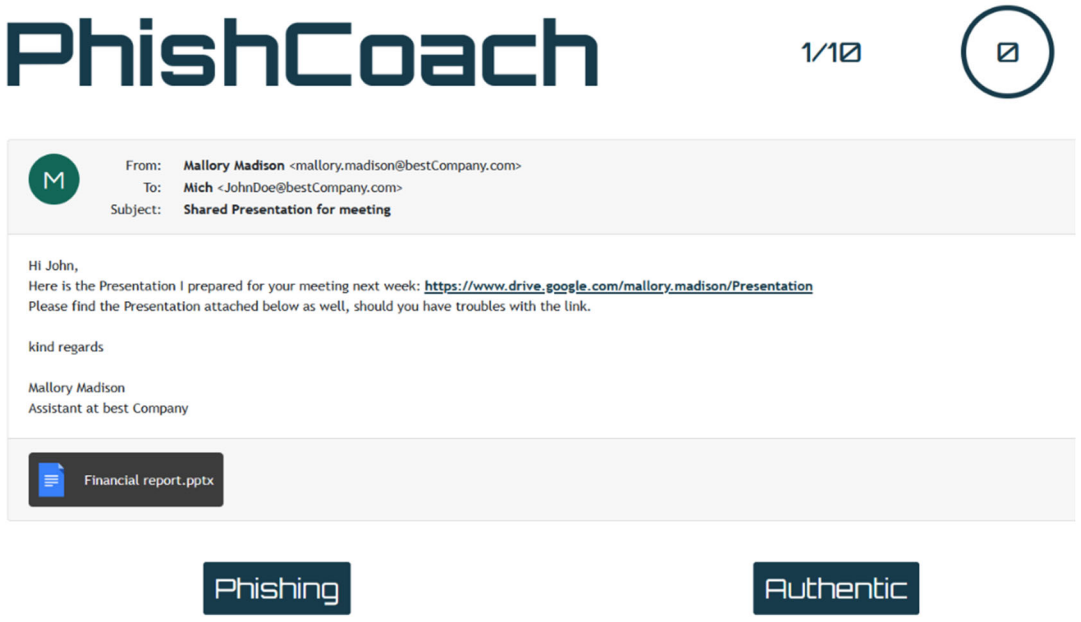


Figure 2: PhishCoach user interface

It includes information on icon creation, the use of Bootstrap, the structure of the HTML, the setup of the CSS and the logic behind the JavaScript file. Each of these components is crucial in their own way, working together to provide stable user experience.

4.1. Icons

The design and creation of the icons for PhishCoach heavily utilized the Adobe Creative Cloud, with Photoshop and Illustrator being the primary tools. The project logo, a central icon, was initially extracted from a poster created for the Media Night. However, to better align with PhishCoach's theme, modifications were made. The logo was then used as the tab icon for the web application.

The PhishCoach text in the top area of the web application was exported as a PNG icon. This was done due to PNG's ability to provide lossless compression and support of transparency.

Figure 3: PhishCoach logo

Moreover, some elements of the web application were styled to look like icons but were generated using HTML and CSS. For instance, the score display was designed as a circle using CSS.

Figure 4: Score display

A similar approach was used when creating the preview about the first letter of the sender's name.

Figure 5: First letter of sender's name

When it came to utilizing external icons, it was always ensured that the icons fell under the Creative Commons Zero (CC0) license. This license allows for unrestricted use without the need to credit the original creator. An example of this in PhishCoach is the attachment file icon.

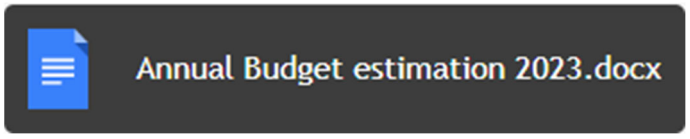


Figure 6: Attachment icon

Although there was initial consideration to create this icon dynamically, with the file ending inside the icon, it was ultimately decided to keep it simple and focus efforts on other areas of the project.

4.2. Bootstrap

Bootstrap was selected as a tool for developing the PhishCoach project due to its ability to create responsive websites. With the rising usage of various screen sizes, having a website that adapts to different devices is important. Bootstrap helps in achieving this responsiveness and speeds up the design process significantly.

A key advantage of Bootstrap is its robust grid system, which simplifies the arrangement of elements on the page. The card template from Bootstrap, for example, facilitates the quick creation of different containers. Despite some drawbacks, such as the loading of surplus code that might not be necessary, the choice to use Bootstrap was deemed beneficial. Given the lack of large elements in PhishCoach, the overhead associated with Bootstrap was not a significant concern. The benefits in terms of speed, convenience, and adaptability ultimately outweighed this issue.

4.3. Structure, Design and Logic

The HTML structure of the PhishCoach web application plays an important part in defining how its various elements are arranged and interacted with each other. The upcoming sections delve deeper into each element, discussing their design and function.

4.3.1. Django Setup

When setting up the HTML structure for PhishCoach in Django, a “base.html” file was created. This file contains all the meta data that is loaded. In a situation where the PhishCoach project is expanded in the future, this base file would be beneficial, as it can hold elements that are common to every subpage. Currently though, its use is limited to containing meta data.

Following the base file is the “index.html”. This file serves as the primary file that is displayed to the user whenever they access the web application. It functions as the main view where the core functionalities and interactions of the PhishCoach project are made accessible to the user.

Every HTML file in the PhishCoach project includes a reference to the static folder. This setup is crucial as it provides a structured way to access a variety of assets used throughout the application such as CSS files, JS files, libraries, fonts and images.

4.3.2. Responsive Design

The concept of responsive design was an important aspect in the creation of PhishCoach, especially given the popularity of smartphone use in today’s world. With most individuals now accessing websites via their vertically

oriented smartphone screens or in a small browser window on a computer, it is essential to create a website that supports various display sizes and aspect ratios. To ensure this adaptability, each new element introduced into PhishCoach was created with responsiveness in mind.

During the development process, four different PhishCoach windows were consistently kept open, each set to a different screen size, to account for a variety of potential viewing scenarios. This testing environment ensured that the site would remain accessible and user-friendly, regardless of the device being used. Bootstrap was chosen as the primary framework for PhishCoach largely due to its focus on responsive design, which further streamlined the process of creating an accessible website.

4.3.3. Link Preview

A small preview feature is implemented in the bottom left corner of the web application, designed to mirror the behaviour of a web browser link preview. This square-shaped element gives a glimpse of where a link will lead. However, because the links featured in the application should not be real, a precaution taken to avoid leading users to harmful websites, a custom link preview is introduced.

Figure 7: Link preview window

The operation of this custom link preview is driven by JavaScript interacting with HTML elements tagged with specific classes. If a class named "link" is detected in the code delivered from the backend, JavaScript assigns an event listener to this pseudo link. When hovering over this fake link, the content of another HTML element identified by the class "link-content" is displayed in the link preview square, thus providing a harmless yet realistic representation of a link's destination.

4.3.4. Logo, Progression & Score

At the top of the PhishCoach web application, there are four elements designed to provide a user-friendly experience.

Figure 8: Top row with logo, progression and score

Starting from the left, the logo text is interactive. By clicking, it refreshes the application and initiates a new quiz. Adjacent to it is a progress indicator that visually represents how much of the current quiz the user has completed. This is followed by a display that informs the user of the total number of emails included in the quiz. To the far right, a circular score counter keeps track of the number of emails the user has correctly identified up to that point.

4.3.5. Email Display

The email display in PhishCoach is set up to be as informative and intuitive as possible. The top part of each email, known as the email head, is designed to resemble a standard mail provider's layout while still offering all necessary information at a glance.

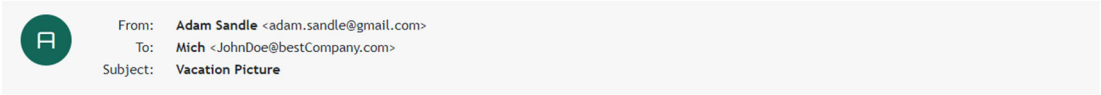


Figure 9: Email head

The head starts with the first letter of the sender's name displayed in a circle. Adjacent to it is the sender's alias and email address in brackets. Beneath this, the recipient information is shown as "Mich JohnDoe@bestCompany.com". This implies that the user is playing a character named John Doe, who is presumably an employee of a fictional company called Best Company. Lastly, the email subject is displayed, featuring custom text supplied from the backend.

Underneath the email head is the body of the email, where the content provided by the backend is displayed.

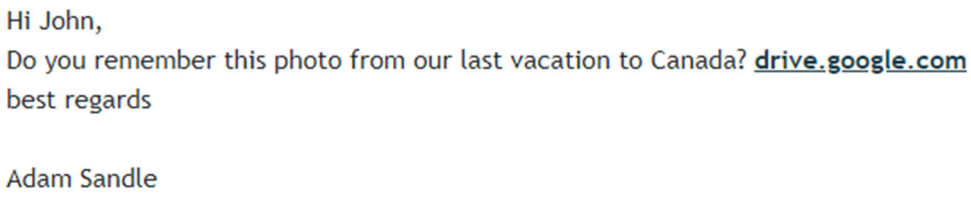


Figure 10: Email content

This section can render HTML code, which allows for a versatile range of content. However, any line breaks in the email will not be shown unless specified by HTML tags. The content area is also where any links are displayed, these links are the sources for the link preview, as discussed earlier in "4.3.3 Link Preview".

Lastly, at the very bottom of each email is an area for attachments - as can be seen in “Figure 6: Attachment icon” - if any are present for that email. In this setup, an attachment is represented by a string and doesn't contain a downloadable file. Furthermore, only one attachment can be associated with each email. A decision was made to limit emails to a single attachment, as having more than one was seen as unnecessary for assessing an email's legitimacy. The most important part of the attachment area is the ability to see the attachment extension. For instance, attachments with ".exe", “.docx” or ".html" file extensions could serve as warning signs.

4.3.6. Buttons

In an attempt to maintain a user-friendly interface, PhishCoach has been designed with minimal clicks in mind. This is achieved by providing just two options: "Phishing" and "Authentic".

Figure 11: Phishing and Authentic buttons

Both these buttons trigger JavaScript methods that reveal the solution window. While this solution window is displayed, the "Phishing" and "Authentic" buttons are hidden, and a "Next" button appears in their place. The decision to position the "Next" button between the "Phishing" and "Authentic" buttons was a conscious one, with the aim to prevent users from mindlessly clicking through the quiz.

Once the "Next" button is clicked, the previous email window is replaced by the next one, the solution window is hidden, and the "Authentic" and "Phishing" buttons reappear, ready for the next round of the quiz. This cycle continues until the quiz is completed.

4.3.7. Solution

After a user has made their selection regarding the authenticity of an email, a window is shown below the email.

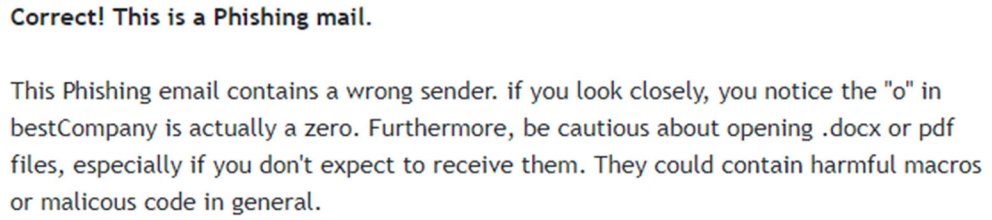


Figure 12: Solution window

As seen in “Figure 1: PhishCoach mockup” this was originally intended to be on the left side of the email. This was later changed for a better flow of the quiz, as it would have resized the email every time the solution would have been shown.

This window firstly informs the user, in bold, whether their choice was correct or not and clarifies, if the email in question was a phishing attempt. The window then displays a text that is sourced from the backend. This text typically includes (though it can contain anything) the clues that indicate whether an email was genuine or phishing. If an email is genuine, the text may elaborate on elements that could have misled the user into thinking it was suspicious. On the other hand, if the email is phishing, the text should list all the reasons, including how they can be identified and why they pose a threat. This way, users are educated by PhishCoach on how to identify potentially harmful elements in an email.

4.3.8. Email Generation

The emails in PhishCoach are structured in a way that allows all necessary information to be loaded into the application at one go. This reduces the need for frequent server communications and makes the experience smoother for the user. Although this method might require a bit more rendering time initially, it cuts down significantly on the loading time when transitioning between emails.

Every email and solution window is labelled with an identification number based on its sequence in the progression order. This number is added to their HTML id attribute. When the user interacts with the "Authentic", "Phishing", or "Next" buttons, a JavaScript code executes. This code updates the user's score, progresses through the sequence, hides the currently visible email and solution, and unveils the next set.

```
<!--Logo & Score-->
<div id="logo" class="row mb-5">…</div> flex
<!--JavaScript Reminder-->
<noscript>…</noscript>
<!--Email-->
<div id="email-1" class="card mb-5 remove">…</div>
<!--Solution-->
<div id="solution-1" class="card mb-5 remove">…</div>
<!--Email-->
<div id="email-2" class="card mb-5 remove">…</div>
<!--Solution-->
<div id="solution-2" class="card mb-5 remove">…</div>
<!--Email-->
<div id="email-3" class="card mb-5 remove">…</div>
<!--Solution-->
<div id="solution-3" class="card mb-5 remove">…</div>
<!--Email-->
<div id="email-4" class="card mb-5">…</div> flex
<!--Solution-->
<div id="solution-4" class="card mb-5">…</div> flex
<!--Email-->
<div id="email-5" class="card mb-5 remove">…</div>
<!--Solution-->
<div id="solution-5" class="card mb-5 remove">…</div>
<!--Game Buttons-->
<div id="game-buttons" class="row mb-5 remove">…</div>
<!--Next Button-->
<div id="next-button" class="row mb-5">…</div> flex
<!--Result-->
<div id="result" class="card remove mb-5" style="margin-top: 100px;">…</div>
<!--Retry Button-->
<div id="retry-button" class="row remove mb-5">…</div>
<script src="/static/js/main.js"></script>
<script>…</script>
```

Figure 13: All in one email generation

Since the user communicates with the server only once, the entire game logic is pre-loaded into a JavaScript file, eliminating the need for further server calls. This works well for PhishCoach as there is no competitive element like leaderboards involved, and hence, no risk of cheating. If a user were to cheat by inspecting the pre-loaded solutions, they'd only be undermining their own learning experience. Besides, doing so would require a conscious effort to seek out and examine the relevant elements within the code.

4.3.9. Final Score Display

Upon completion of the quiz, the top row, the email window, buttons and solution panel are replaced by a final score display.

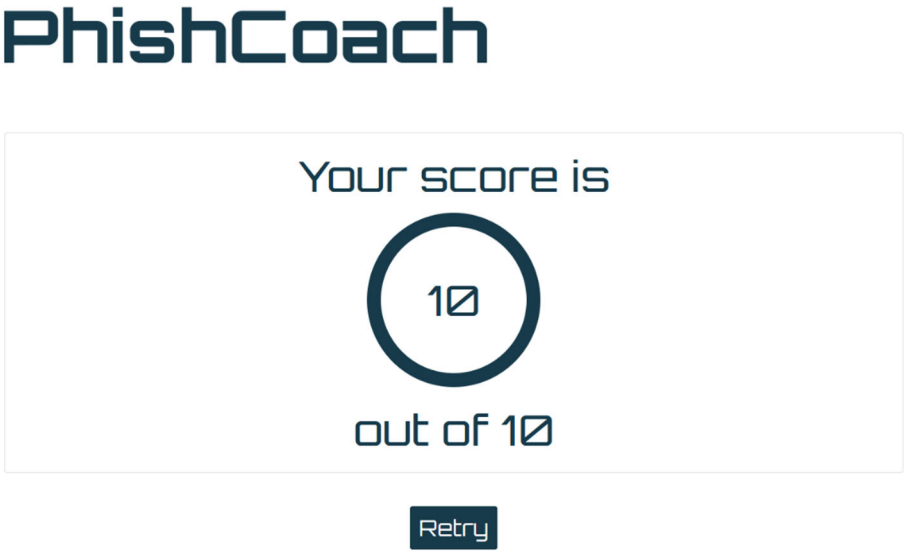


Figure 14: Score display after quiz

This new window showcases the player's final score, providing immediate feedback on their performance. Alongside the score, a retry button is offered. This allows for immediate replay, enabling users to challenge themselves again.

4.3.10. JavaScript Reminder

If a user has JavaScript turned off, it's important that they understand the PhishCoach website won't function as intended. To ensure this, a clear warning message is set to display at the top of the site when JavaScript is detected as deactivated.

Figure 15: JavaScript reminder in the user interface

This alert requests the user to enable JavaScript, making it clear that the website's functionality depends on it. By doing so, any potential confusion or issues stemming from the site not working should be avoidable.

5. Backend

This section discusses the technologies employed in the backend, providing reasons for each choice, and offering a detailed peek into the backend logic of the program.

5.1. Python

The backend of the web application, PhishCoach, is coded in Python using the Django framework. The choice to use Python was influenced by the team's previous experience with this programming language. Moreover, the team had worked with Java for their last three software projects and wanted to diversify their experience.

Python, being one of the most popular programming languages, has an easy-to-understand syntax. This simplicity makes it a great choice because if any issues were encountered, solutions could be found quickly due to the widespread use of the language. The straightforward syntax allowed the team to focus more on programming, minimizing the time wasted on debugging syntax-related issues.

5.2. Django

Django, a robust Python framework, was the top choice for the PhishCoach project. What sets Django apart is that it comes with a lot of built-in features. This means less manual setup compared to other frameworks like Flask. It's like a full toolset that offers everything you need to build a web application right out of the box.

For example, Django takes care of database connections for you, which can be a time-consuming task if done manually. Also, Django's inbuilt templates make the program more dynamic and easier to extend, which is a great plus. A big help was Django's large and supportive community, along with its clear, detailed documentation. This makes it easier to find answers when stuck. The fact that the team had prior experience with Django gave a head start, but even if this would not have been the case, the benefits of Django would still be hard to ignore.

5.3. Database

PhishCoach relies on a database to keep emails stored for the long term. Django's built-in sqlite3 database was used for the project. To manage emails - adding new ones, editing existing ones, or deleting them - Django's standard admin panel has been used. The database itself is kept simple, built around a single table that holds all the necessary details for an email.

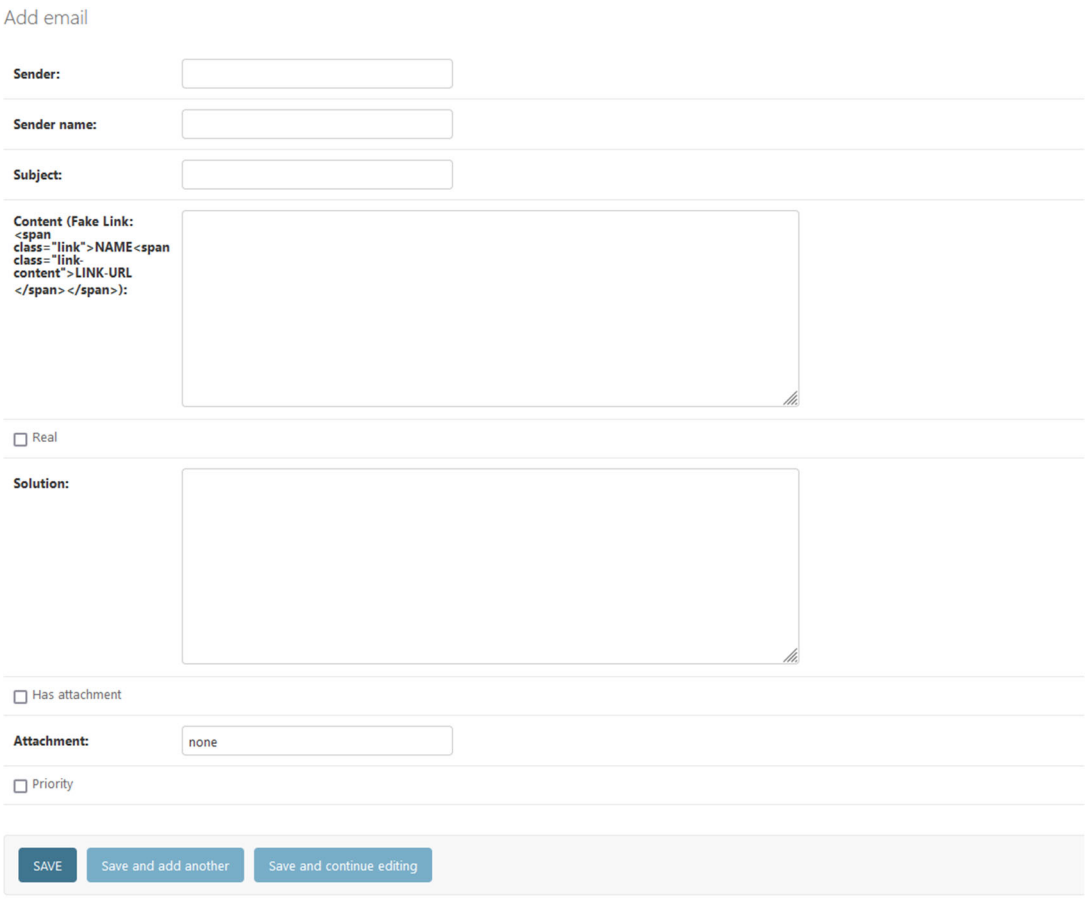


Figure 16: Backend configuration panel

This table consists of the following columns:

- Sender – Email address of the sender
- Sender name – Name which is displayed in the email header
- Subject – Text field with the subject of the email
- Content – The email body that allows usage of HTML Tags
- Real – Boolean value marking an email as real or as phishing
- Solution – Explanation of why a email is real or phishing
- Has attachment – Determines if the attachment windows should be displayed
- Attachment – Text that is displayed as the attachment name
- Priority – Ensures that an email will be at the start of the quiz

Apart from custom attachments, this table enables the creator of the emails full email customization.

5.4. Email

Emails form the core of the PhishCoach application. The aim was to include a wide range of possible phishing scenarios to provide users with a

comprehensive understanding of the threats they could encounter. These scenarios varied from seemingly harmless co-worker emails and supplier phishing attempts to more severe CEO fraud phishing situations. In these latter scenarios, the attackers had previously compromised the CEO's account. The effort was made to always try to implement more than one clue in a phishing email to cover as many different dangers as possible.

To make the situations more relatable, a fictional setting was created. A character named John Doe was introduced, an employee at a company called Best Company. John works in the finance department and receives various emails during his workday. However, not all these emails are as genuine as they first appear to be.

5.4.1. Phishing emails in detail

In one of the emails, the Unicode characters were swapped with characters that look identical. Instead of an “e” the unicode character U+0435 was used. Because this character is not distinguishable by the human eye, a fake link was also put into the email.

A fake link can be created by leveraging the popular URL shortening service, TinyURL. It's worth noting that services like TinyURL can mask the true destination of a link, making it an ideal tool for phishing attempts. In this scenario, the true URL would likely be concealed within a TinyURL link, adding a layer of deception.

In other emails look-alike email addresses were used instead of look-alike URLs, meaning that either a character was left out or swapped with a similar one (e.g. swapping “O” and “0”).

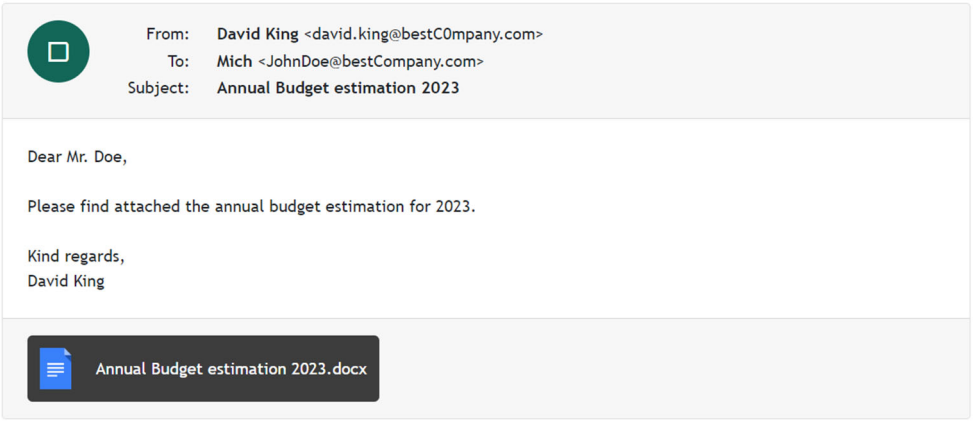
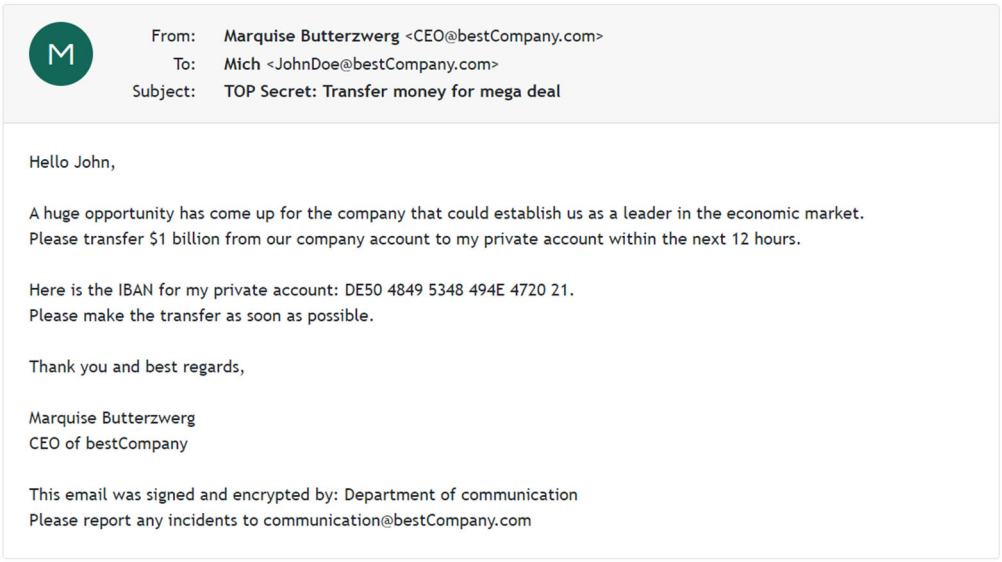


Figure 17: Example phishing email with swapped “O” and “0”

These were also the emails which had the highest rates of going unnoticed as phishing emails.

As mentioned above a scenario was implemented in which CEO fraud was attempted. In this email the classic signs of phishing emails were used, which usually involve time pressure and the message not to tell anyone about it.



From: **Marquise Butterzwerg** <CEO@bestCompany.com>
To: **Mich** <JohnDoe@bestCompany.com>
Subject: **TOP Secret: Transfer money for mega deal**

Hello John,

A huge opportunity has come up for the company that could establish us as a leader in the economic market.
Please transfer $1 billion from our company account to my private account within the next 12 hours.

Here is the IBAN for my private account: DE50 4849 5348 494E 4720 21.
Please make the transfer as soon as possible.

Thank you and best regards,

Marquise Butterzwerg
CEO of bestCompany

This email was signed and encrypted by: Department of communication
Please report any incidents to communication@bestCompany.com

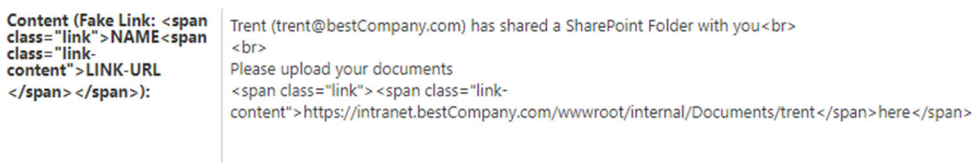
Figure 18: Example phishing email with CEO fraud

Something similar was used for an email scenario in which a different company was imitated that the company of John Doe owed money to. In it the standard approach to put time pressure on the victim and a fake link in combination were used.

In the standard emails of the project exists a sender with the name Mallory. This name was chosen to throw off students that visited IT-Security lessons because it is often implied that Mallory is an acronym for a malicious attacker. However, the goal was to confuse people with this and check whether they get careless when reading this name.

5.5. HTML Content

In PhishCoach, creating emails includes some unique features. Notably, the content field can process HTML code, which means you can directly paste real phishing email content into this field and it will display as expected.



Content (Fake Link: NAMELINK-URL):

```
Trent (trent@bestCompany.com) has shared a SharePoint Folder with you<br>
<br>
Please upload your documents
<span class="link"><span class="link-
content">https://intranet.bestCompany.com/wwwroot/internal/Documents/trent</span>here</span>
```

Figure 19: HTML in backend

However, this also implies that any links in the HTML will be active. To manage this, a mechanism was developed using CSS and JavaScript to simulate the links.

When a user moves their cursor over a text with the "link" class, an event listener is triggered. This action removes a CSS attribute and as a result, the URL is shown in the link preview in the bottom left corner of the screen, similar to how it would appear when hovering over a standard hyperlink. Hence, links need to be formatted in this way:

```
<span class="link">

     <span class="link-content">URL</span>

     DISPLAYED TEXT

</span>.
```

5.6. Priority

PhishCoach has a feature that makes it possible to choose which emails should be seen first. Each email has a "priority" option. This is a simple yes or no choice. When the priority is set to "yes", these emails are always shown before other emails and are shuffled in their own list. This way, the most important emails come first.

If there are more than five priority emails, then only five of those marked as priority emails will be shown in the list. This feature ensures that the most important or relevant emails are always shown to the user first. This way, it can be ensured that users focus on the most critical emails first in their training.

6. Project Setup

To initialize the project on a new machine, the ReadMe, Initialization Script, and Startup Script serve as important tools.

6.1. ReadMe

The "ReadMe" file serves as a comprehensive guide to set up and utilize the project. It provides detailed instructions that should be followed from beginning to end, skipping steps only when explicitly stated. This file outlines the system requirements necessary for the project and provides both automated and manual setup processes. It is designed to be user-friendly, aiming to ensure a smooth installation and setup. In the latter part, it covers the most important use cases for the admin panel, making it a valuable resource for the project's operation.

6.2. Initialization Script

The "PhishCoach_Initialization.ps1" script serves as a helpful tool to set up the project on a new system. To employ this script, PowerShell is used according to the steps specified in the readme file. The first action performed by this script is to trigger a debug output. This allows the user to monitor progress as it carries out various operations. Without this, the script would appear to not be working for a period of time. After this, the script stores various paths for future use.

The second aspect of this script is the creation of a shortcut to the "PhishCoach_Startup.ps1" script. The paths previously saved are employed here. Elements such as a target path, arguments, a working directory and an icon are set. If any of these elements are absent, with the exception of the icon, the shortcut would fail to function if relocated.

The third part of the Initialization Script handles the creation of a virtual environment and its subsequent activation. It's worth noting that all the following steps are performed within this environment. To execute this step, a script created by Python is used. However, this requires that script execution be allowed in the console beforehand, a process that is detailed in the ReadMe file. Following this, Django is installed, Django database migrations are made and an admin account is created. Due to security reasons, scripts are not permitted to insert password details. This step is therefore accomplished by simulating user input.

6.3. Startup Script

The " PhishCoach_Startup.ps1" is designed to simplify the process of launching PhishCoach. Typically, to launch the script, it would be necessary to navigate to the appropriate folder, right-click the file and select "Run using PowerShell". The creation of a shortcut and corresponding arguments, however, streamlines this process. This shortcut can be placed anywhere and is immediately identifiable thanks to its assigned icon. A simple double-click is required to launch the application. This is made possible by linking the shortcut to PowerShell, which in turn is set to execute the script. Additionally, the executionPolicy of the current user must be set to at least RemoteSigned as Restricted and Undefined do not suffice to use the Shortcut.

When the Startup Script is initiated, it launches a separate PowerShell within the existing one. This second PowerShell navigates to the current folder, activates the virtual environment, performs migrations and then launches the server. This nested process ensures that all required steps are completed in the correct environment and order.

Once these actions are completed, the initial PowerShell window takes care of opening two browser tabs. One tab contains the admin panel, while the other showcases the main PhishCoach application. With these two tabs open, the original PowerShell window can then be closed. This process allows for a smooth transition into using PhishCoach, with all necessary windows and tools immediately available.

7. Validation

In the process of developing a software project, validation plays an essential role. This section delves into the different strategies applied to validate the work done on the PhishCoach project. It addresses the key areas testing, identifying problems, and gathering feedback. In creating the program, continuous testing was carried out to ensure its proper functioning. Inevitably, some problems emerged that were either unsolvable or impractical to address, given the project's scope and resources. To make the application as effective and user-friendly as possible, feedback was continually sought and incorporated.

7.1. Testing

When developing software applications, testing is a crucial process to ensure reliability and performance. For PhishCoach, even though specific unit tests were not conducted, various other testing approaches were adopted to ensure minimal bugs and maximum efficiency.

For the front end, testing was immediately carried out while a new component was programmed. Different windows were filled with various contents to verify the responsiveness of the website. When JavaScript code was created, html testing elements were established to check if the code was functioning correctly. This way, any issues with the display, layout, or behaviour of the website could be spotted more easily and fixed.

In contrast, testing the backend involved assessing the database by sending data to the frontend and checking if the content was accurate. Django was also immediately tested whenever any changes were implemented. This check was done to confirm if Django was correctly manipulating the database. Changes in Django were verified by creating test database content via the Django admin portal and observing it on the frontend.

7.2. Identified Problems

In the development process, several issues surfaced that proved challenging to resolve completely. One of these is related to the incorporation of CSS code into emails. Any CSS included within an email would influence the entire web page. There were attempts to mitigate this by renaming every

element on the webpage to obscure names or by trying to encapsulate the webpage within an iframe. However, the iframe solution did not work as anticipated. The trouble lay in the fact that the email was rendered as HTML content by Django into the frontend HTML file. When an iframe was used, the content had to be inserted as a string into the “srcdoc” attribute, making it impossible to dynamically generate an email via Django. The current workaround is to be careful about the content of emails created in the backend. This includes ensuring all HTML tags are correctly formatted. If a closing tag is forgotten, it can disrupt the entire frontend page.

Another unresolved issue persists to the working mode of Django. At present, only the debug mode is operational. When attempting to switch Django into release mode or setting debug to false, problems arise. Libraries fail to load, emails are not displayed correctly, and the formatting of the web application is compromised. The root of this problem appears to lie with the static folder. When Django is set to release mode, the static folder, which holds all content for Bootstrap, JavaScript files, CSS files and images, isn't accessed as expected.

One thing to keep in mind when using real emails is the potential issue with external objects like images. There's a chance they might not load properly. It's likely that the servers hosting these external objects could be blocking the request. However, emails that link directly to external objects (for instance, using the “src” tag in HTML code) worked fine in the application.

7.3. Feedback

Getting feedback from outside sources is an important part of any development project. This approach was implemented in the creation of PhishCoach. After each component of the application was developed, it was presented to an individual not involved in the project. This fresh perspective offered an unbiased opinion on the project's progress, helping to identify areas that might need improvement or further clarification.

These feedback sessions were tremendously beneficial. While the team may have found certain aspects of the project good and understandable, having external viewpoints helped to reveal areas that might not have been as clear to an outside observer. An example for this would be the bold text in the email head highlighting the sender, recipient and subject. This continuous cycle of development, presentation and feedback ensured that many issues were quickly identified and rectified, ultimately leading to an application that should be a lot more user-friendly.

7.4. Presentation

In preparation for Presentation Day, a series of slides were created to outline the PhishCoach project. Knowing that the time limit for the presentation was three minutes, a full day was set aside specifically for practicing the pitch. The rehearsal allowed for a smooth delivery without exceeding the allotted time or the need for physical notes during the actual presentation. Nonetheless, some notes were brought on stage as a safety for potential stage fright scenarios.

For the Media Night event, a more interactive setup was crafted. A small stand with a tablet was set up, which was held up by a 3D-printed vertical stand. This allowed visitors to easily notice and interact with the PhishCoach application.

Throughout the Media Night, numerous visitors engaged with PhishCoach. Initially, the setup included a quiz of ten emails, but it was soon adjusted to five emails, as it became evident that ten was too lengthy and exceeded the attention span of most visitors. An observation made during these interactions was that users had an average success rate of around 80% in identifying phishing emails from the presented examples. This suggested that there was a 20% chance of people not recognizing a phishing email in a similar real-life scenario, thus underlining the relevance and potential usefulness of PhishCoach as a tool for raising awareness about email phishing.

8. Discussion

This section offers an opportunity to reflect on the overall PhishCoach project, shedding light on its potential applications, areas for improvement, and a general review of the development process.

8.1. Potential Use Cases

PhishCoach holds potential as a practical teaching tool, particularly in an IT-security lecture setting. Given that it allows for the creation of custom emails, a professor could design a series of phishing examples that align with the curriculum. This would keep students actively engaged while reinforcing key course concepts in a real-world context. It provides an interactive way to highlight the signs of phishing emails, thereby enhancing students' awareness and understanding of this critical aspect of cybersecurity.

Similarly, PhishCoach could be beneficial in a corporate environment. The IT security department within a company could leverage the application to conduct hands-on phishing awareness training. By creating emails that are specific to the company's context, employees can gain a better sense of the

kind of phishing attempts they are most likely to encounter. This could improve their ability to recognize and avoid falling victim to real phishing attempts. However, it's worth noting that the successful implementation of PhishCoach in these settings hinges on the email creator's understanding of realistic phishing email constructs.

Additionally, PhishCoach can provide significant value for older users who may not be as familiar with technology. While this demographic may face challenges in navigating new applications, they stand to benefit greatly from the phishing awareness training provided by PhishCoach. Its simple, user-friendly interface may prove particularly beneficial for these individuals, promoting ease of use while enhancing their understanding and resilience against phishing emails.

8.2. Improvement Capabilities

While the PhishCoach program fulfilled the Minimum Viable Product (MVP), several areas could benefit from future improvements.

One such improvement relates to handling links. Currently, any customized links needs manual formatting to fit the specific structure. This could be streamlined with automation, either through a dedicated button that inserts this link structure into the email content field in the admin panel or by text parsing functionality that automatically replaces traditional links and enables a link preview for PhishCoach.

Incorporating a means to suspend the current session and continue it another time represents another potential upgrade. By allowing users to save their sessions via cookies or a login setup, they could easily resume where they left off.

A login feature would also pave the way for the inclusion of a high score list, showcasing top scores and encouraging competition. Alternatively, the system could provide statistics on performance rates across various questions and the difficulty level of different emails.

A start page where users could define their names might also be a beneficial addition. However, this could compromise the simplicity of the site, which currently showcases only a maximum of two buttons.

Addressing database backup is another critical aspect. In the event of a server failure, it's essential not to lose the entire database. An already working but inelegant solution is to regularly copy the "db.sqlite3" file to a separate location.

Also worth considering is a preview feature for the admin panel, which could parse the text in the content field and display the rendered result. This would need to synchronize the images, CSS, JavaScript and most importantly the template files from the frontend to the backend. A current workaround for this would be to mark the desired email as priority to make it appear first in the application.

8.3. PhishCoach Review

The development of PhishCoach offered a great learning experience, particularly in expanding understanding of Django, Python, HTML, CSS, JavaScript and Bootstrap. Additionally, the experience with Visual Studio Code improved, especially when it comes to coding in a shared environment like GIT.

While not all additional features were able to be implemented, the minimal viable product was achieved with some additional features. Creating the emails was a particular challenge, as the perspective had to shift from detecting phishing emails to crafting realistic ones. It was necessary to conceal the obvious signs of phishing emails.

The presentation and Media Night went well. Most visitors walked away with new insights. One observation was that concentrated reading reduces the chances of a successful phishing attack significantly. Nevertheless, despite the heightened alertness of users knowing that the application is a phishing trainer, the project was successful in simulating as close to a real phishing scenario as possible.

The workload division was justly managed. This was facilitated by the consistent collaboration on the project, ensuring a balanced time investment from all members with Jonas Koch focusing on the frontend and Steffen Schawjinski on the backend. While the backend tasks were completed before the frontend, this member could shift focus to researching and creating phishing emails for the tool. A clear to-do list guided the team in distributing tasks fairly throughout the project's duration.

Conclusively, PhishCoach's development proved both enjoyable and highly educational. The end product provides a sense of achievement and the team hopes that the tool will help prevent phishing attempts in the future.

List of References

Petrosyan, A. (2023, 07 19). *statista*. Retrieved from https://www.statista.com/statistics/184083/commonly-reported-types-of-cyber-crime-global/

Ponemon Institute. (2021). The Impact of the New Normal on Workplace Privacy: A Study of Business & IT and IT Security Managers. 8-12.