



Abertay University®

SCHOOL OF DESIGN AND INFOMATICS

Scam Shield: Combating Phishing Emails in the Browser

Ryan Tyler Denholm

2104018@uad.ac.uk

A dissertation submitted as partial fulfilment of the requirements for the award of the degree Bachelor of Science (Honours) in Ethical Hacking, 2025.

Supervised by

Marc Kydd

7TH May 2025

List of Figures	3
List of Tables	4
Acknowledgements	5
Abstract	5
List of Acronyms	6
1. Introduction	7
1.1 Research Question	7
1.2 Aims	8
2. Literature Review	8
2.1 Phishing: Definition and Evolving Scope of the Threat	8
2.2 Static vs Dynamic Detection Approaches	9
2.3 Machine Learning Techniques for Phishing Detection	10
2.4 Conclusion	11
3. Methodology	11
3.1 Model Development	12
3.2 Extension	17
3.3 User-Testing	21
3.4 Survey	24
3.5 Statistical Analysis	26
4. Results	27
4.1 Model Results	27
4.1.2 Confusion Matrix	28
4.2 Participant Results	29
5. Discussion	32
5.1 Model Decisions	32
5.2 Dataset limitations and Practical User Classification Results	33
5.3 Practical User Confidence	34
5.4 Extension Implementation and Limitations	35
5.5 Privacy	36
5.6 Testing website implementation and limitations	36
5.7 Summary	36
6. Future Work	37
7. Conclusion	39
References	41

Appendices	43
Appendix A: Manifest.json	43
Appendix B: Background.js.....	44
Appendix C: GDPR Research Data Management, Data Sign Off Form	45
Appendix D: Statistical Analyses Results.....	46
Appendix E: Full SUS Data.....	49

List of Figures

Figure 1: Methodology flowchart, showing each integral stage of development performed in this section.	12
Figure 2: Sample of CEAS_08.csv dataset.	13
Figure 3: Model Development Flowchart, displaying the methodical approach to development.....	14
Figure 4: Scam Shield Model Architecture	15
Figure 5: Extension Flowchart, illustrating the steps taken when an email is scanned	17
Figure 6: DOM Observation, Green Box is showing "aria-label", Red Box shows the email body.	20
Figure 7: Extension displaying "scanning this email" banner.	20
Figure 8: Extension's Popup when probability is above 70%.	21
Figure 9: Extension displaying "Safe" banner within the email client	21
Figure 10: Practical 1 Showing the simulated inbox, buttons and clarification on "scam" or "genuine".	23
Figure 11: Consent Popup informing users of Inspectlet.	23
Figure 12: Sample of the predictions JSON used for the Scam Shield predictions on the website	24
Figure 13: Practical Two modal, with Scam Shield Probability Banner.	24
Figure 14: Survey Flowchart	25
Figure 15: Validation metrics across three, five and ten epochs, plotted as scattered bar-graphs, showing no statistical significance between different training lengths.	27
Figure 16: This confusion matrix displays the number of correct and incorrect classifications made by the Scam Shield model on the test dataset. The top-left and bottom-right cells represent correct predictions, the bottom-left and top-right reflect misclassifications.	28
Figure 17: Practical Participant Results, Individual Scores (N=14), pair-matched score pre-Scam Shield use and with Scam Shield, represented with mean +- SD, as determined by Wilcoxon Test, P = 0.0059.	29
Figure 18: User Confidence Results, Individual Scores (N=14), pair-matched score pre-Scam Shield use and with Scam Shield, represented with mean +- SD, as determined by Wilcoxon Test, P = 0.0078.	30

Figure 19: Individual SUS item responses across 14 participants (N = 14). Bars represent mean ratings, with error bars = standard deviation. Overlaid symbols show individual participant responses.	31
Figure 20: Bar graph showing the number of different misclassifications per practical, N=14.	34

List of Tables

<i>Table 1: Preprocessing Steps for Email Text Prior to Phishing Detection</i>	19
<i>Table 2: Stages of model utilisation in background.js</i>	44

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor, Marc Kydd. Without his continuous guidance, expertise, and support, I would not have been able to develop the tool this work presents. His thoughtful feedback and suggestions helped refine my initial ideas and introduced me to alternative approaches I had not previously considered. Marc played a vital role in ensuring that my work remained aligned with the project's development milestones and deadlines, and I am deeply appreciative of his mentorship throughout this process.

I would also like to extend my heartfelt thanks to my friends, family and partner. Their unwavering encouragement, patience, and belief in me helped me stay grounded during times of stress and uncertainty. Their support was invaluable in helping me maintain perspective and motivation throughout the duration of this project.

Lastly, I am grateful to the wider Ethical Hacking faculty at Abertay University for fostering an environment that allowed me to explore my interests and grow both technically and academically.

Abstract

Phishing, exploiting human vulnerabilities to bypass technical defences, continues to represent a significant cybersecurity threat. This project aimed to design, implement and evaluate Scam Shield, a lightweight, browser-based machine learning tool for real-time phishing detection. This tool ensures user privacy through exclusive client-side operation with minimal external dependencies.

Leveraging a Long Short-Term Memory (LSTM) neural network architecture, the model is trained on a publicly available dataset comprised of both phishing and legitimate emails. The model was optimised for efficient phishing detection within a Firefox extension via TensorFlow.js. Model performance was evaluated through standard classification metrics, while user-centred efficacy was assessed via user research wherein 14 participants performed phishing detection tasks with and without Scam Shield.

This research shows that browser-based, real-time phishing detection powered by client-side machine learning is both viable and effective, achieving an accuracy of 98%, helping users detect more phishing attacks, and be more confident when identifying such attempts. Scam Shield offers a promising foundation for the development of privacy-preserving, user-centric security tools that augment human judgement, protecting users against phishing threats.

List of Acronyms

ML Machine Learning

AI Artificial Intelligence

LSTM Long Short-Term Memory

GDPR General Data Protection Regulation

URL Uniform Resource Locator

HTTPS Hypertext Transfer Protocol Secure

SMS Short Message Service

HTML Hyper Text Markup Language

CNN Convolutional Neural Network

BERT Bidirectional Encoded Representations from Transformers

NLP Natural Language Processing

TF TensorFlow

TF.JS TensorFlow.js

OOV Out-of-Vocabulary

ReLU Rectified Linear Unit

DOM Document Object Model

API Application Programming Interface

CSS Cascading Style Sheets

UI User Interface

JSON JavaScript Object Notation

SUS Software Usability Scale

FP False Positive

FN False Negative

TP True Positive

TN True Negative

1. Introduction

Phishing remains one of the most persistent and damaging forms of cybercrime, with attackers continuously adapting their strategies to evade traditional security systems. These deceptive emails are designed to harvest user credentials, install malware, manipulate behaviour, or carry out other forms of digital exploitation. Despite increased public awareness and widespread use of spam filters, phishing continues to succeed in compromising people's security. This is often achieved by exploiting user trust at the interface level, where individuals must decide the legitimacy of an email based on minimal technical cues and visual indicators.

As phishing attacks evolve in complexity, traditional detection methods, such as blacklists and rule-based filters, struggle to keep pace. Machine Learning (ML) offers a promising alternative, providing the ability to learn and generalise from patterns in email content and structure. ML-based approaches can detect subtle indicators of phishing that static methods often miss. This makes them increasingly valuable in cybersecurity; by enabling models to learn from real-world data, ML can provide more adaptive and intelligent threat detection mechanisms.

Conventional phishing defences typically rely on static detection methods, such as known URL databases or signature-based filtering. While these are effective for known threats, they fail to detect zero-day attacks or sophisticated phishing campaigns that manipulate language and formatting. Dynamic detection, in contrast, involves analysing the content and structure of emails in real-time, thus identifying potential threats based on learned behaviour and context.

This project adopts a dynamic approach to phishing detection, moving away from centralised, server-side models towards a browser-based, client-side implementation. This shift offers not only speed and adaptability but also significant advantages in data privacy.

Server-based phishing detection tools require users to transmit their email content to third-party servers for analysis, raising concerns about data privacy and regulatory compliance. By embedding the detection model directly within the browser environment, this project eliminates the need to send user data externally. The model runs entirely on the client side, enabling a robust solution for phishing detection while maintaining user privacy and autonomy. This design aligns with current principles of privacy-preserving machine learning and "data minimisation" as promoted by the GDPR and similar frameworks.

1.1 Research Question

The core research question guiding this dissertation is:

Can a browser-based machine learning model reliably detect and flag phishing emails in real-time, helping users identify phishing attempts?

This research question underpins an interdisciplinary inquiry into practical cybersecurity: combining machine learning deployment constraints, real-time user interaction, and the ethical imperative of privacy by design. To address this research question, this project aims to contribute a scalable, user-friendly, and ethically aligned solution to phishing mitigation.

1.2 Aims

The primary aims of this project are:

1. Train an AI model to detect phishing attempts.
2. Implement the model into the browser via a Firefox extension.
3. Evaluate the model's performance and user experience.

The successful achievement of these aims is anticipated to contribute to the development of an effective tool for enhancing users' email security.

2. Literature Review

2.1 Phishing: Definition and Evolving Scope of the Threat

Phishing is a form of cyber-attack that combines technical methods with social engineering. It typically involves an attacker masquerading as a trustworthy entity to deceive victims into divulging sensitive information or installing malware. Classic phishing attacks have often been associated with fraudulent emails containing links to fake login pages, aiming to steal credentials. However, the scope of phishing has continually evolved. According to Lim et al. (2025), modern phishing campaigns may employ multiple attack vectors, including email, fraudulent websites, instant messages, or phone calls which increasingly serve as entry points for malware like ransomware. For example, a 2024 report by Cofense found a 67% increase in credential phishing compared to 2022, leading to devastating ransomware attacks and data breaches, underscoring how phishing is not only about credentials but also a delivery mechanism for malware (Cofense, 2024).

Crucially, phishing leverages human trust. Attackers exploit psychology through social engineering tactics such as urgency, fear or impersonating authoritative figures to trick users into action. A recent comprehensive study emphasises phishing as a “socio-technical” process targeting human and technical vulnerabilities in tandem (Alkhalil, et al., 2021). Attackers often follow a multi-phase approach consisting of planning (gathering information and choosing a lure), preparation (setting up fake sites or exploits), attack (sending the phishing email or link), and valuable acquisition (harvesting credentials or installing malware) (Alkhalil, et al., 2021). Each phase can leverage evolving techniques: for instance, today's phishers might spoof domains, use HTTPS to appear legitimate, or craft highly personalised spear-phishing emails. Additionally, phishing now extends beyond generic mass emails to targeted attacks on organisations and other mediums like SMS and voice

calls, reflecting an expanding threat landscape. This evolution necessitates defences that address both the technical indicators of phishing and the human factors that make phishing effective.

2.2 Static vs Dynamic Detection Approaches

Early anti-phishing solutions largely relied on static detection techniques, such as rule-based heuristics and blacklists of known malicious URLs. While these static defences provided some protection, they suffer notable limitations. Blacklist-based filters and rigid rules are reactive and struggle to catch new (“zero-day”) phishing attacks (Abuadbba, et al., 2022). Attackers can easily evade static rules by slightly altering URLs or email content. As a result, static techniques often produce high false negatives (missed attacks) and can yield false positives if legitimate content happens to match a pattern. Abuadbba et al. (2022) found that phishing detection methods notes that blacklists fail to detect novel phishing sites until after victims have been exposed. Similarly, simple rule-based filters (e.g. flagging emails with certain keywords or HTML features) may not adapt to evolving tactics of phishers, who continually tweak their strategies to bypass known rules. Indeed, static detection methods cannot evolve with new threats, leading to both missed attacks and potentially high false alarm rates over time, Lim et al. (2025).

These shortcomings have led to the rise of dynamic, adaptive detection systems, particularly those based on machine learning. Instead of using only fixed signatures, dynamic approaches learn to recognise phishing through patterns in data. In recent years, researchers such as Lim et al. (2025) and industry practitioners have increasingly adopted machine learning (ML) models to automatically classify emails or URLs as phishing or legitimate. ML-based systems can analyse a wide range of features from, from email header metadata and textual content to URL string properties and domain reputation and find subtle correlations that static rules might miss. Notably, the first ML-driven phishing email filter (PILFER) used a Random Forest classifier on features like IP-based URLs, number of links, and mismatched URLs in emails, achieving high accuracy even back in 2007 (Fette, et al., 2007). This demonstrated early on that data-driven classifiers could outperform static heuristics by combining many indicators.

Modern dynamic systems often leverage statistical models and AI to continuously improve. For example, they may retrain on newly reported phishing samples to recognise emerging attack patterns. The shift to dynamic detection has significantly improved coverage of new phishing attacks, but it also introduces new challenges, such as the need for large training datasets and concerns about model transparency (addressed later in this review). Overall, the literature strongly suggests that rule-based detection alone is insufficient, and intelligent, adaptive techniques are now central to phishing defences Lim et al. (2025). Scam Shield embraces this trend by using learning-based methods to stay effective against evolving attacks, rather than relying on rules.

2.3 Machine Learning Techniques for Phishing Detection

Machine learning has become a cornerstone of phishing detection in the past decade, with both traditional classifiers (like Random Forest or Support Vector Machine) and deep learning models showing promise. Deep learning approaches have gained traction for their ability to automatically learn features from raw data such as email text or URLs. Recurrent neural networks, especially LSTM networks, have been extensively explored due to their strength in modelling sequential data. Phishing URLs and email bodies can be viewed as sequences (of characters or words), and LSTMs can capture sequential patterns and context that might indicate a phishing attempt. For instance, an LSTM can learn that certain sequences of characters in a URL (like “paypal-login...id=...”) or certain wordings in an email are strong predictors of phishing (Alshingiti, et al., 2023). Alshingiti, et al. research further confirms the appropriateness of implementing LSTM models to classify phishing URLs by learning character-level patterns, as well as in analysing email language for phishing cues.

Alongside LSTMs, researchers have applied Convolutional Neural Networks (CNNs) and hybrid models. CNNs can learn spatial or positional features and have been used on textual data (e.g., by treating a URL string or email as a “sequence image” or 1-D signal). Interestingly, there are mixed findings on which approach is superior. Alshingiti et al. (2023) compared a pure LSTM, a pure CNN, and a combined CNN-LSTM model for phishing URL detection. Their results showed that the CNN-based model achieved the highest accuracy (about 99.2%), slightly outperforming the hybrid CNN-LSTM (97.6%) and the LSTM alone (96.8%). This suggests that for certain URL datasets, CNNs were particularly effective, due to their ability to capture local character patterns, whereas the sequential memory of LSTM added less value. In general, hybrid models (e.g., CNN+LSTM, or deep neural networks combined with recurrent layers) can leverage the strength of multiple architectures and have frequently reported high performance in phishing detection.

Beyond URLs, phishing email detection has also leveraged deep learning on textual content. Recent work by Atawneh & Aljehani (2023) evaluated deep NLP models including CNNs, LSTMs and the latest Transformer-based model BERT on a large collection of phishing and legitimate emails. They found that the best performance was achieved with a BERT-based model, with LSTM being the next best, reaching detection accuracy up to 99.6%. This underscores that while LSTMs are powerful, newer language models (BERT) can further improve detection by understanding context and semantics in emails. Nonetheless, LSTMs remain a popular choice due to their relative simplicity and impressive performance on sequence data without requiring the extensive training data that Transformers often need.

It is important to critically note that extremely high accuracies reported in research (often over 90%) may not directly translate to real-world performance. Many experiments evaluate models on curated datasets in controlled settings. Issues like

training data quality and distribution can affect a model's true efficacy. Still, the trend in literature is clear, machine learning (including deep learning) has vastly improved phishing detection capabilities compared to earlier heuristic systems.

2.4 Conclusion

Considering the literature discussed, Scam Shield adopts an LSTM architecture for phishing email detection due to a balanced combination of efficacy, efficiency, and deployability within a browser environment. While newer models such as BERT or hybrid CNN-LSTM architectures have demonstrated slightly higher accuracy in controlled experiments, their practical use in real-time, browser-based phishing detection tools remains limited due to their significant computational overhead. These models often require powerful hardware and large memory allocations, making them unsuitable for deployment in client-side environments such as browser extensions. This project plans to leverage an LSTM-based classifier for detecting phishing emails in-browser, building on these successful techniques. However, it will do so with the understanding of their limitations: for example, a black-box neural model might be accurate but hard to interpret, and it must be continually updated to recognise new phishing tactics (Lim, et al., 2025). This insight will influence Scam Shield to consider explainable outputs or user feedback loops despite using advanced ML, thereby making the tool not only effective but also trustworthy.

The decision to run phishing detection on the client-side, rather than outsourcing computations to a server, is equally critical. Server-side approaches may offer access to more powerful models, but they introduce serious data privacy concerns, particularly in the context of emails, which often contain personal or confidential information. Performing classification locally ensures that sensitive user data does not need to be transmitted over the internet, mitigating risks related to data breaches or non-compliance with privacy regulations such as GDPR.

Moreover, keeping detection local also circumvents latency and connectivity issues. Server-based detection may introduce delays or rely on a consistent internet connection, both of which degrade the user experience. A lightweight, in-browser model offers faster inference times and uninterrupted phishing analysis regardless of network status.

3. Methodology

The following methodology describes the comprehensive steps taken in the development, implementation, and validation of the Scam Shield project. The goal of which is to clearly outline the processes and procedures to facilitate replication and critical evaluation.

The Scam Shield project was developed within the Agile development methodology, emphasising iterative and incremental progress. This process was chosen due to its flexibility, allowing continuous feedback integration, regular testing, and modular adjustments based on results.

This methodology covers the complete workflow of the Scam Shield project. The process begins with data collection and preprocessing, continues through model development and training using TensorFlow/Keras, and concludes with integration into a browser extension using TensorFlow.js. Each phase is described in detail to support reproducibility and clarity.

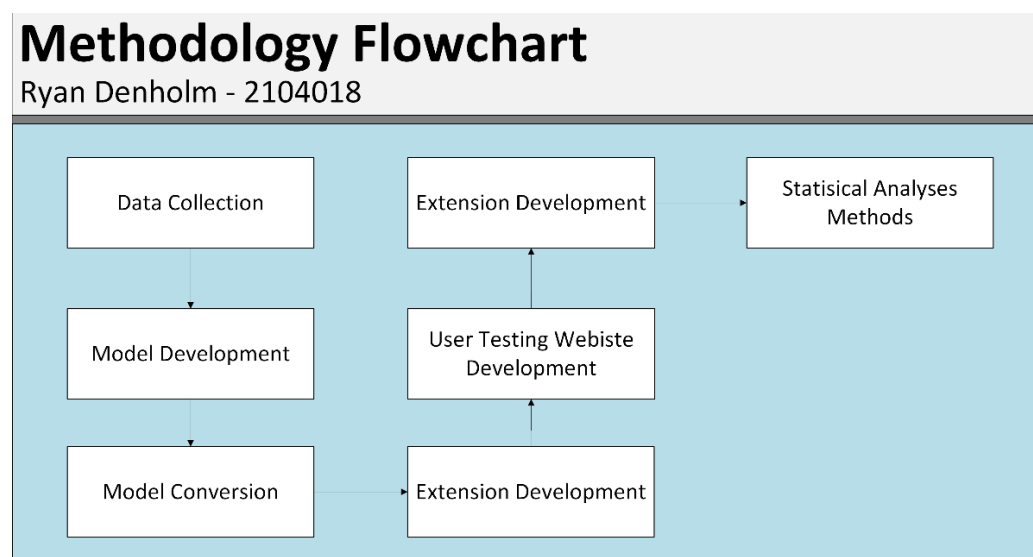


Figure 1: Methodology flowchart, showing each integral stage of development performed in this section.

3.1 Model Development

Developing a fundamentally sound and reliable machine learning model is essential to ensuring the effectiveness of any phishing detection system. The model's ability to accurately classify emails as either phishing or legitimate forms the basis upon which further implementation can function as intended. This section details the comprehensive development of the Scam Shield model, beginning with the sourcing and preparation of suitable datasets, followed by model architecture design, training, and evaluation. The process concludes with the conversion of the trained model into a format suitable for real-time, browser-based deployment.

3.1.1 Data Collection and Preparation

The dataset used for training this model was found publicly online, this provided sufficient data for the model to train from and saved time with pre-labelled data. CEAS_08.csv (Alam & Khandakar, 2024) was used for training the model, due to the dataset providing broader context for the emails such as: sender and recipient emails, date of email, body of email, and Spam/Legitimate labelling.

sender	receiver	date	subject	body	label	urls
Young Esposito <Young@world.de>	user4@gvc.ceas-challenge.cc	Tue, 05 Au	Never agree to be a loser	Buck up, your troubles caused by small dimension will soon be over!	1	1
Mak <pinet'sB63@cabte.ph>	user2.2@gvc.ceas-challenge.cc	Tue, 05 Au	Bel friend Jenna Jamieson		1	1
Daily Top 10 <Karnandee-opangvi@univer>	user2.3@gvc.ceas-challenge.cc	Tue, 05 Au	CNN.com Daily Top 10	>THE DAILY TOP 10 >from CNN	1	1
Michael Parker <ivornal@pobox.com>	SpamAssassin Dev cer@spamassassin.apache.org	Tue, 05 Au	Re: sun commit: r615793 - in /spar	Would anyone object to removing .so from this list? The .so TLD is	0	1
Gretchen Suggs <extema1aap1@loa.noffi>	certcc user2.2@gvc.ceas-challenge.cc	Tue, 05 Au	Special Prices Pharm Moreinfo		1	1
Caroline Aragon <dathaidomainnamesm@th>	user7-ext3@gvc.ceas-challenge.cc	Wed, 06 Ai	From Caroline Aragon		1	0
Replica Watches <jhorton@thebakercompany>	user2.10@gvc.ceas-challenge.cc	Tue, 05 Au	Replica Watches	We have fake Swiss Men's and Ladie's Replica	1	0
Daily Top 10 <acidirev_1572@tccwpq.com>	user2.3@gvc.ceas-challenge.cc	Tue, 05 Au	CNN.com Daily Top 10	>THE DAILY TOP 10 >from CNN	1	1

Figure 2: Sample of CEAS_08.csv dataset.

The emails were loaded and encoded in the UTF-8 Format to ensure consistent handling of diverse text data, including special characters and symbols, thereby reducing encoding-related errors or data loss that could impact the quality of text processing. Irregular entries or corrupted lines were skipped (“on_bad_lines=”skip”). “on_bad_lines” is a parameter in pandas.read.csv() that instructs the parser to ignore problematic rows (e.g. rows with too few/many columns), reducing potential loading errors.

The dataset was then split into training and validation sets, 80% allocated to the former and 20% to the latter. This is to ensure unbiased evaluation of model performance on unseen data.

Due to the removal of missing values, the training dataset had become unbalanced with 37,349 total emails remaining, 21,807 being identified as “Scam” and 15,542 emails being identified as “Legit”. This led to a 28.8% imbalance leaning towards Scam emails, oversampling was conducted to resolve this imbalance using “RandomOverSampler” from imblearn (developers, 2024), a python library used for handling class imbalance for machine learning datasets. In this case, used to enhance the model performance and reduce bias towards the majority class. It is important to note that this was implemented after the training and validation split, as to not introduce data leakage, ensuring that oversampling was conducted exclusively on the training dataset. Oversampling the validation dataset would lead to an overly optimistic and unreliable performance metrics, thus compromising the validity of the evaluation.

A tokenizer was employed to convert email text into numerical sequences, limited to the 20,000 most frequent words to optimise performance and manage vocabulary size efficiently. Unknown or rare words were handled using an Out-of-Vocabulary (OOV) token labelled “UNKNOWN_WORD” and sequences were padded or truncated to a fixed length (500 words per email), enabling consistent model input dimensions.

The fitted tokenizer was saved using Python’s pickle module, ensuring reproducibility and consistent data processing during development.

To further handle class imbalance effectively, “compute_class_weight” from Scikit-Learn was implemented to assign higher weights to minority classes, enabling the neural network to give proper attention to underrepresented examples during training.

These preprocessing steps, notably the handling of class imbalance, tokenizer consistency, and controlled splitting of the dataset, significantly impact the model

performance. Appropriate preprocessing directly enhances the model's robustness, and ability to identify phishing content accurately in varied real-world scenarios.

3.1.2 Model Development and Training

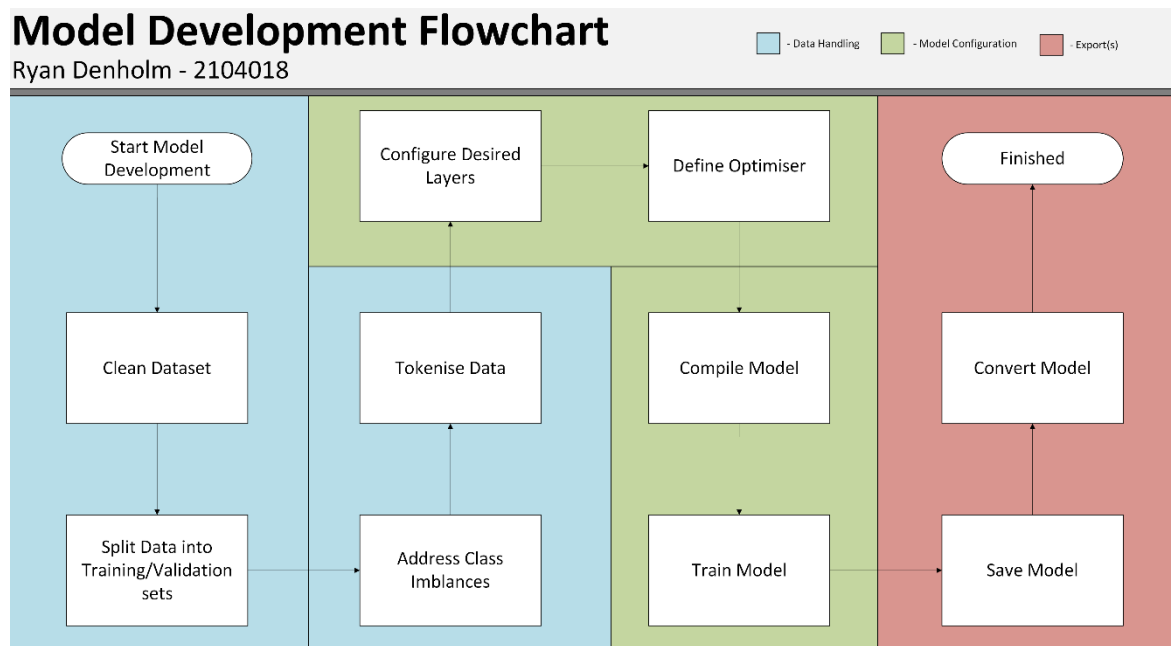


Figure 3: Model Development Flowchart, displaying the methodical approach to development

TensorFlow/Keras (TensorFlow, 2024) was chosen as the primary framework for this project due to its efficiency, extensive documentation, and strong community support.

One of the main strengths of TF is its extensive support for deep learning models, including advanced architectures like RNNs, LSTM, and attention mechanisms (Abadi, et al., 2016). The framework also offers efficient GPU acceleration and optimised computational performance, which significantly enhances the speed and scalability of training large datasets. Additionally, TF provides numerous pre-built models and layers, simplifying model construction and allowing developers to focus more on design and experimentation rather than low-level implementation. Furthermore, TF supports direct and streamlined export to TensorFlow.js, which was crucial for integrating the model with the browser-based Scam Shield extension, as seen in a later section (Smilkov, et al., 2019).

However, TF is not without drawbacks. Models developed using this framework can be resource-intensive, particularly when deployed in the browser, requiring careful optimisation to ensure smooth and efficient operation. Additionally, the vast number of features and options available can present a steep learning curve, especially for those unfamiliar with the framework.

An alternative considered during development was PyTorch (PyTorch, 2024), a framework known for its flexibility and intuitive interface. PyTorch's dynamic computational graphs allow for quick and built-in experimentation with model architectures, making it a powerful tool for prototyping and working with custom

components such as attention mechanisms (Paszke, et al., 2019). It also offers superior debugging and transparency, allowing for clearer error tracing and easier model inspection. PyTorch enjoys strong support from the research community and is frequently used in experimental deep learning applications.

Despite these advantages, PyTorch was ultimately decided against. While it could have handled the project’s architectural requirements effectively, TF was more suitable due to its easier and more reliable model export capabilities, specifically tailored for deployment in browser environments via TensorFlow.js (TensorFlow.js, 2024).

3.1.2.1 Model Architecture

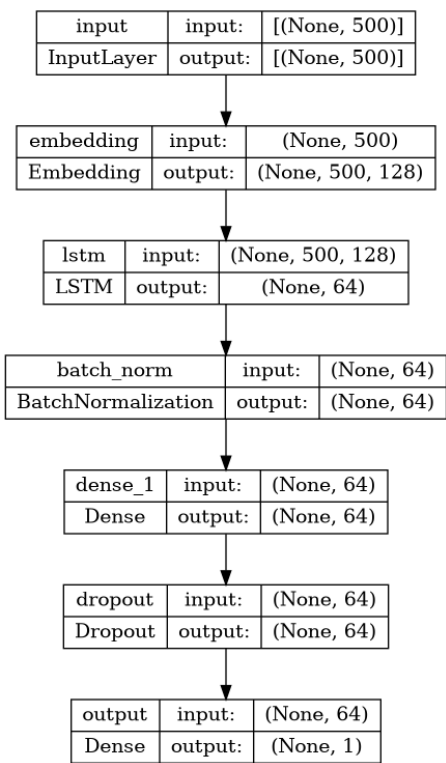


Figure 4: Scam Shield Model Architecture

The architecture of the Scam Shield model begins with an input layer designed to handle sequences of uniform length, specifically 500 tokens. This padding ensures that all email inputs, regardless of their original length, are processed consistently by the model. Following the input layer, an embedding layer with a dimensionality of 128 is applied. This layer transforms each token into a dense vector representation that capture semantic relationships and contextual nuance. This is particularly important for phishing detection, as subtle variations in language can differentiate legitimate emails from malicious ones.

Central to the model is the LSTM layer, which contains 64 units. This component enables the network to retrain and process sequential information, making it well-suited to understanding the structure and flow of language as found in emails. To

prevent overfitting (where the model learns patterns too specific to the training data and fails to perform well on new, unseen data), dropout (set to 0.5) and recurrent dropout (set to 0.3) were implemented. Additionally, L2 regularisation with a weight of 0.01 was applied to increase generalisation and avoid excessive model complexity.

To refine the features extracted by the LSTM layer, the model incorporates a sequence of dense (fully connected) layers. A dense layer ensures that each neuron receives input from all neurons in the previous layer, allowing the model to synthesise complex feature interactions. The process begins with batch normalisation, which standardises the inputs (Rounds to 0 or 1) to the dense layer to stabilise and accelerate training by maintaining consistent data distributions. This is followed by a dense layer with 64 units and a ReLU activation, which introduces non-linearity and enables the model to learn complex patterns. A dropout layer, set to 0.5, is applied next to reduce the risk of overfitting by randomly deactivating half of the neurons during each training iteration. Finally, the output layer uses a sigmoid activation function to produce a binary classification, clearly distinguishing between phishing and legitimate emails.

Model optimisation was carried out using the Adam optimiser (Kingma & Ba, 2015), an algorithm that combines the advantages of two popular methods “AdaGrad” and “RMSProp” (Brownlee, 2021) by computing adaptive learning rates for each parameter. It was selected for its adaptive learning rate capabilities and efficient convergence, particularly in complex neural networks such as those handling natural language data. A learning rate of 0.005 was applied, alongside a gradient clipping (clipnorm=1.0) to prevent the occurrence of exploding gradients during backpropagation, which can occur when the model encounters unexpected inputs (such as rare or unknown words), causing excessively large weight updates and destabilising training.

Compilation of the model used binary cross-entropy loss, optimal for binary classification tasks, and a range of common evaluation metrics including accuracy, precision, recall and F1-Score to comprehensively measure model performance.

Training was conducted over three epochs (refer to Section 5.1 Model Decisions), using a batch size of 128, sufficient for stable convergence while maintaining computational efficiency. An early stopping callback monitored validation accuracy with patience set at one epoch, automatically restoring the best-performing model weights to avoid overfitting and enhance model generalisation capabilities.

Class weighting was implemented to address class imbalance, providing balanced importance to each class, and ensuring fair model performance across both classes.

Evaluation of the model was systematically carried out using a reserved validation split of the dataset, analysing metrics such as accuracy, precision, recall and F1-score. A confusion matrix was computed and visualised, aiding the identification of potential areas for improvement.

3.1.3 Conversion to TensorFlow.js

Following model training and evaluation, the TF model was converted to TF.js format to ensure compatibility with browser deployment, specifically utilised in the two other practical aspects later discussed.

Conversion utilised the official TensorFlow.js converter tool “tensorflowjs_converter,” allowing for consistent integration into the web-based environments. This conversion is a crucial step in Scam Shield’s development, enabling the real-time analysis of emails within users’ browsers, significantly enhancing the practical usability and accessibility of Scam Shield.

3.2 Extension

The following section provides a detailed breakdown of the Scam Shield browser extension, focusing on its architecture, components, and operational flow. It explains how background and content scripts work together to facilitate real-time phishing detection within an email client. The background logic handles the loading and execution of the machine learning model using TensorFlow.js, while the content script integrates directly with email interfaces like Outlook to monitor, extract, and analyse email content as the user interacts with their inbox. Key processes including model loading, vocabulary handling, input processing, DOM observation, and phishing probability scoring are described, offering insight into how Scam Shield delivers its core functionality of phishing detection within the browser.

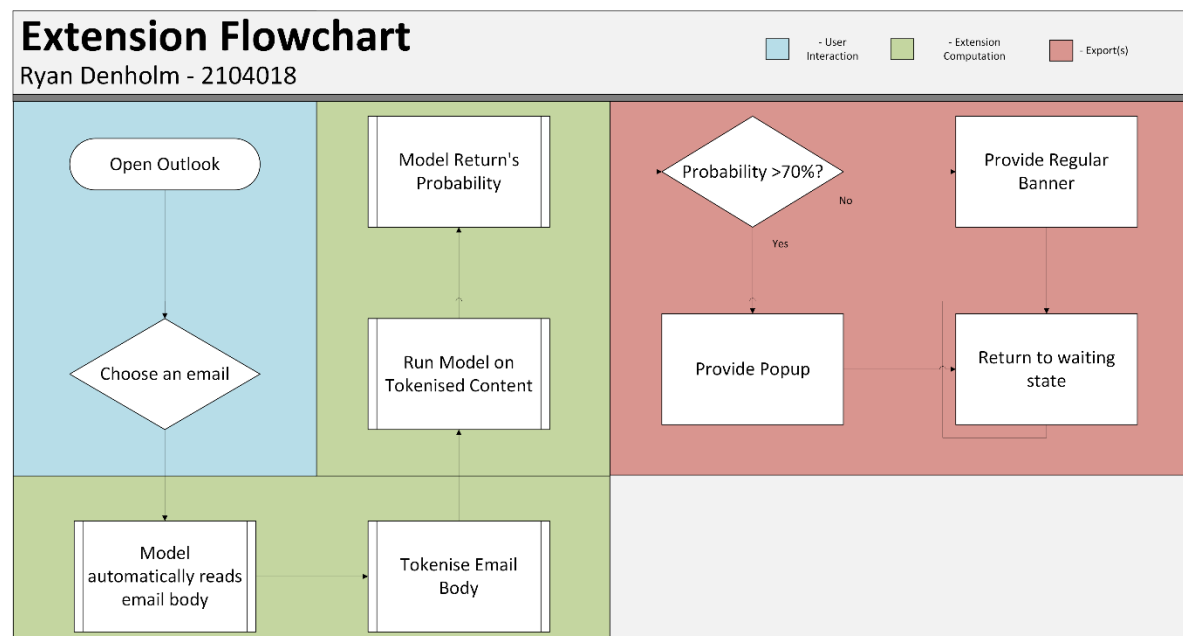


Figure 5: Extension Flowchart, illustrating the steps taken when an email is scanned

3.2.1 background.js and script handling

The background.js section defines the persistent scripts essential for loading and continuously running the core logic of the extension, including machine learning models and libraries (See Appendix B).

The “scripts” section specifies critical scripts required to initialise and operate the machine learning model and back-end functionalities. Tf.min.js and tf-layers.min.js are required for the Scam Shield model to be built within the extension. The “persistent” tag is used to ensure that all scripts necessary for the background functionality always remain active and accessible.

Content scripts are integrated directly into web pages visited by users, executing the front-end scanning logic automatically upon loading or interaction. The current iteration of the extension is active on all URLs, dictated by the “matches” tag. The content.js script is passed here, allowing for direct access to the open web page for real-time phishing analysis, and “run_at” controls when the script is loaded, in this case it is set to “document_idle”, meaning the script will load once the web page content is fully loaded.

The vocabulary, created during the model training phase, is loaded asynchronously into the extension to convert email text into numerical sequences consistently. This step is essential for accurate predictions to be produced, as mishandling of potential “unknown words” can cause errors in prediction, leading to a false negative or false positive result.

This vocabulary variable is saved globally, allowing for optimised performance by negating multiple loads of the vocabulary.

The pretrained Scam Shield Model, converted to TensorFlow.js format, is loaded into the extension, making the predictive capability accessible within the browser context.

The presence of Scam Shield is verified by checking the “scamShieldModel” variable, if it is not null, the model is already loaded. If the variable is null however, the TensorFlow function “loadLayersModel” is used in combination with the model URL (the file location of the saved model). Once the model is loaded, the model is then built with the specific input shape that the model is expecting “null, 500”. If the model is successfully loaded, a console log will be printed confirming the model has loaded, otherwise a console error will occur.

Raw email texts are converted to numerical sequences matching the input format expected by the model. The preprocessing involves tokenization, numerical mapping, and padding to maintain consistent input shapes.

Process	Reason	Implementation
Standardising Inputs	Machine learning models can only process	The function converts email content into lowercase and removes

	numerical data, not raw textual content.	special characters, ensuring consistent preprocessing irrespective of original formatting.
Tokenization & Numerical Mapping	The model was trained in numerical sequences representing words. To accurately detect phishing, the input for prediction must match the data format used in training.	Each word from the text is matched to its corresponding numerical value from the previously loaded vocabulary. Words not seen during training are assigned to default numerical value to prevent errors.
Sequence Padding and Truncation	Recurrent Neural Networks require inputs of a uniform length, without uniformity, TensorFlow.js would raise shape mismatch errors, leading to failed predictions.	The numerical sequence representing email text is padded with zeros if the body is shorter than 500 tokens, or truncated if above 500 tokens, enforcing a strict input length.

Table 1: Preprocessing Steps for Email Text Prior to Phishing Detection

This approach is vital to achieving the desired implementation of the Scam Shield model, ensuring consistency with training conditions and avoiding prediction errors due to mismatching input shapes.

The pre-processed email text is passed through the loaded machine learning model to predict the likelihood of phishing. The result, a probability score, quantifies the risk associated with the email content.

Finally, the script is set up to listen for requests from the extension's front-end (content.js). The function first waits to ensure that both the model and tokenizer are loaded, before then adding a listener for an action request of "scanEmail". Once this request is received, the runPrediction function is called, to analyse the email's body for elements of phishing, ultimately returning a probability of that email being a scam.

3.2.2 Content.js

This script is responsible for the front-end user interaction and real-time email scanning functionality within the Scam Shield browser extension. It interfaces with the background logic to provide the model with the email content, resulting in phishing predictions directly within the user's email client interface.

Once the user browses to their email client (currently coded for Outlook), the content.js script utilises a MutationObserver to monitor changes within the Document Object Model (DOM). This observer ensures that new emails loaded dynamically into the user interface are automatically detected and scanned without manual user interaction.

Let lastScannedEmail = "".

This variable stores the last scanned email text, ensuring the scan function only activates when new or different email content appears, optimising performance and reducing redundant scanning operations.

The MutationObserver continuously monitors changes in the email content area, identified specifically by the selector '[aria-label="Message body"]'. When a change is detected, indicating a new or updated email, the scanEmail() function is triggered automatically to analyse the email content for potential phishing threats.

It is important to note that the selection of '[aria-label="Message body"]' is specifically looking for the font used by Outlook in their email body., this was found inspecting the element of the webpage and allowed for a simple lookup, whereas for Gmail, "tbody" would have to be used to find the body of the email.

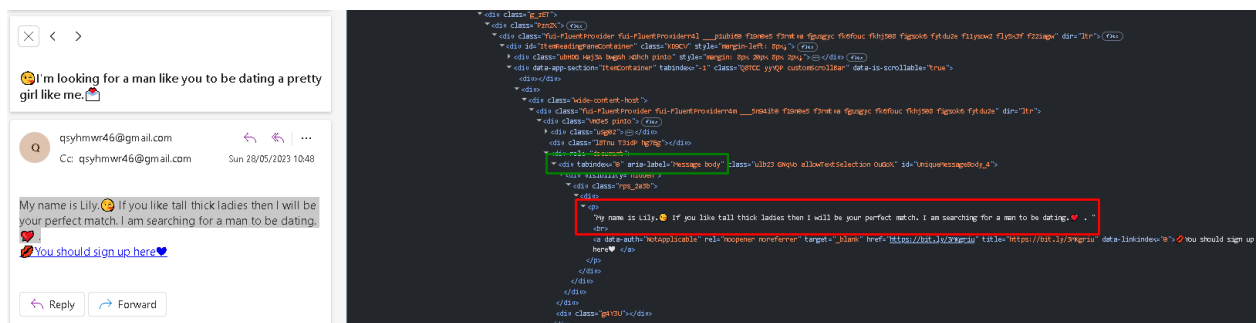


Figure 6: DOM Observation, Green Box is showing "aria-label", Red Box shows the email body.

The extension requires clear communication with users about the current state of email scanning; this is handled by the functions "showLoadingMessage()" and "showResult()". They create or update a visible banner in the user's interface, informing users about ongoing scans and displaying results clearly.

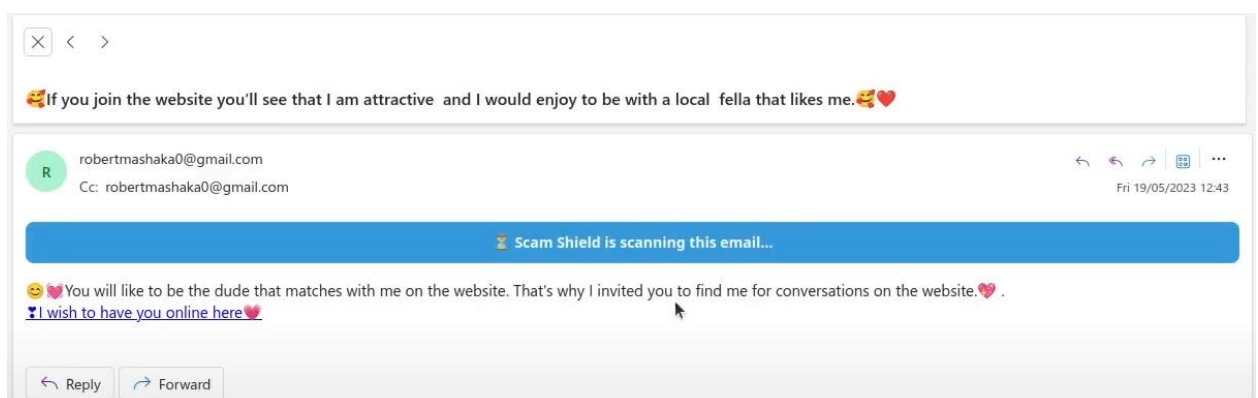


Figure 7: Extension displaying "scanning this email" banner.

Extracted email content is sent to the background script, initiating phishing prediction analysis. The "scanEmail()" function transmits the email content to the background script through Chrome's messaging API, receiving back a phishing probability score.

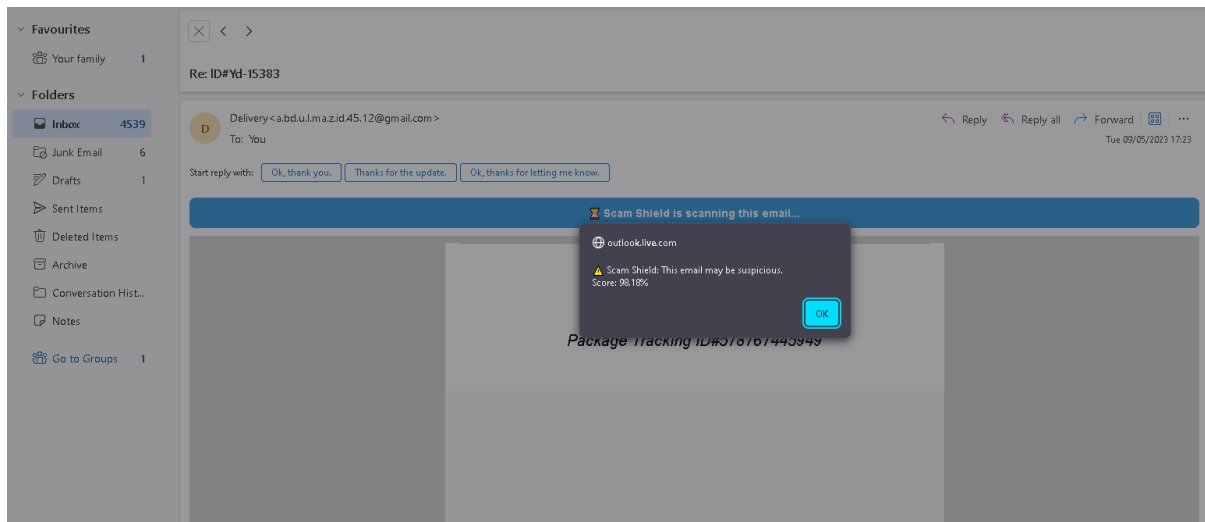


Figure 8: Extension's Popup when probability is above 70%.

The extension provides immediate and clear feedback to users based on the prediction analysis, showResult() evaluates the phishing probability and dynamically displays and intuitive, visual warning or confirmation message directly within the email client interface.

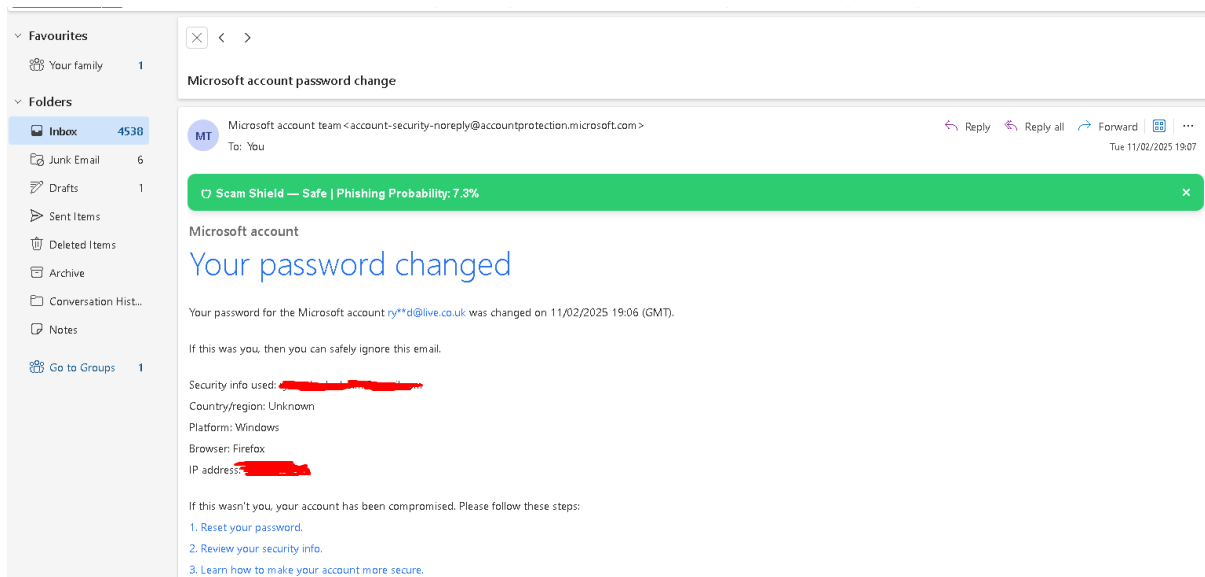


Figure 9: Extension displaying "Safe" banner within the email client

For more information about the manifest configuration and background.js scripts used in Scam Shield's extension, please see Appendix A and Appendix B respectively.

3.3 User-Testing

The Scam Shield testing environment was developed to evaluate the tool's real-world effectiveness in helping users detect phishing emails. The primary aim of this user testing phase was to observe whether the presence of Scam Shield influenced participants accuracy and confidence when determining the legitimacy of emails. The

testing environment was hosted on a static HTML page (scamshield.html), using GitHub Pages (GitHub, n.d), simulating an inbox-like interface.

The study was structured around Practical One and Practical Two. Practical One, where participants evaluated ten randomised emails (five phishing and five legitimate) without any Scam Shield assistance. Practical Two, had participants evaluating a different set of randomised emails (again five phishing, five legitimate) this time with Scam Shield predictions visible.

This two-part format enabled a repeated-measure design, helping isolate the impact of the tool on the same participant across both conditions. The decision to use paired tasks rather than separate test groups was made to minimise variability and highlight individual changes in performance.

3.3.1 Formatting and Design

The basic HTML and CSS defines document structure, sets viewport for responsiveness, includes external resources, Bootstrap and TensorFlow.js. This is done to ensure compatibility, provide a framework for the UI styling and machine learning model execution.

Inline CSS styling is used to define visual aspects of the website, provides the animation for the site, colour scheme and responsive layout rules. This creates visually separate panels (modals) clearly and provides consistent, professional interface.

Bootstrap is employed to enable the webpage to dynamically scale to user's devices, making the website more accessible while also providing the modals that contain both Practical One and Practical Two. This is designed to have the participant focus on the task at hand, making the content of both practicals be the focal point of the webpage, while blurring the background. JavaScript is used to load the emails from the provided dataset and select 10 random emails (five phishing and five legitimate) for each practical. Once the emails are selected, they are passed to the webpage to be displayed to the participant, with two buttons, "Scam" and "Genuine".

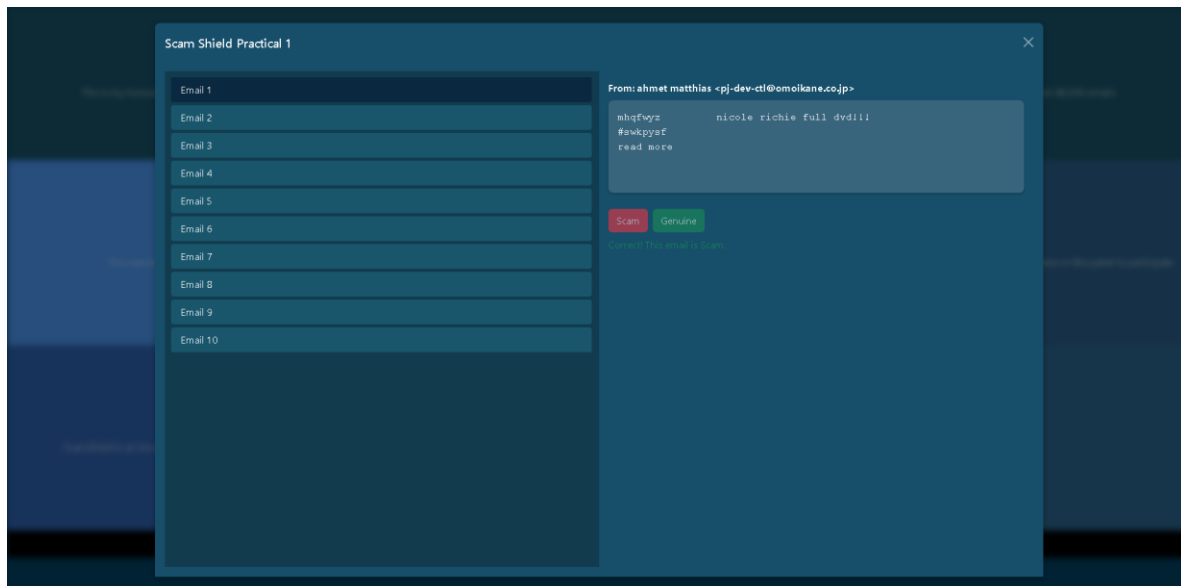


Figure 10: Practical 1 Showing the simulated inbox, buttons and clarification on "scam" or "genuine".

3.3.2 Data Collection and Observation

To evaluate participants' interaction with the simulated email interface, Inspectlet (Inspectlet, n.d) was used as a non-intrusive analytics tool. Inspectlet recorded session data including mouse movements, click paths, and button choices. This allowed for the reconstruction of each participants' actions, which were then manually reviewed to determine accuracy.

Given the remote data collection, a clear and mandatory consent popup was implemented on page load, outlining Inspectlet's role and ensuring ethical transparency.

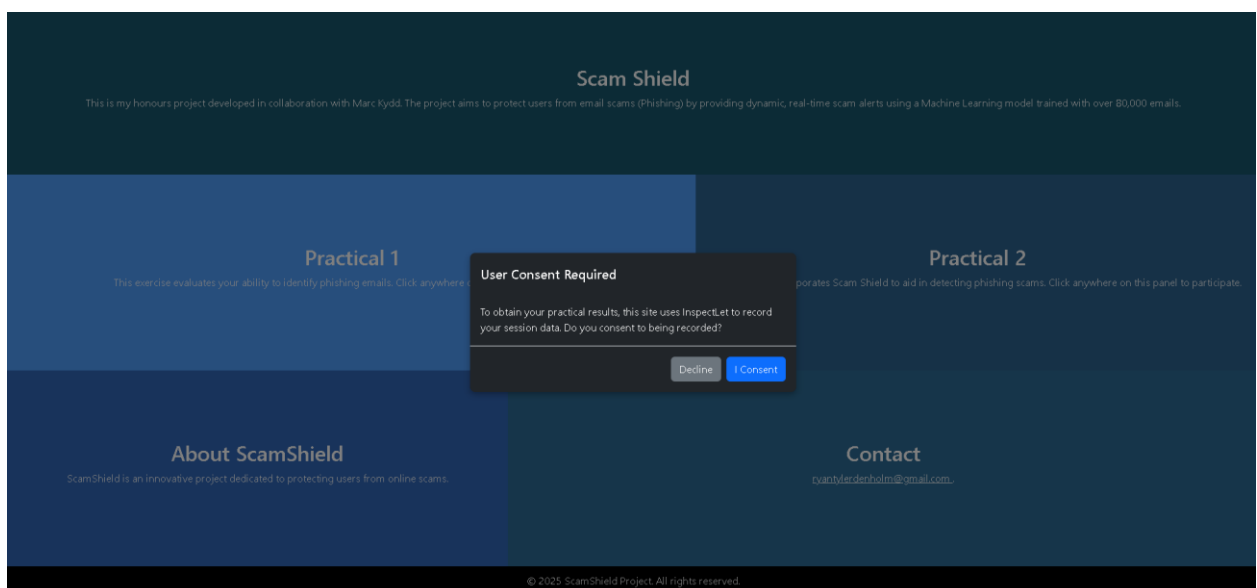


Figure 11: Consent Popup informing users of Inspectlet.

The results were then collected from Inspectlet's dashboard, with those results being

The predictions that are present within Practical Two are provided by a JSON file, which was created by the model, running locally and saving the required data for the practical to take place. This JSON provides the email body, sender email address, the prediction label (scam or genuine) and the associated probability of that prediction.

```
{
  "sender": "Pritesh Damani <gxegmxv@macrovision.com>",
  "body": "hey,\nwhen you click on the last picture button, it is does not go\nback to the first one if i then click on the first one.\njust play around a little bit to find the bug.\n\nthanks",
  "label": 0,
  "predicted_probability": 0.49495044350624054,
  "predicted_label": 0
},
{
  "sender": "Worth Lassiter <limelightp6@federrealty.com>",
  "body": "it breaks your heart to see the one you love is happy with someone else, but it's more painful to know that the one you love is unhappy with you. url",
  "label": 1,
  "predicted_probability": 0.5947404503922327,
  "predicted_label": 1
},
{
  "sender": "Fredrick Hardy <ArturowhitneySchultz@politicalcompass.org>",
  "body": "they look and feel exactly like the real thing.\n\nhigh quality rolex replica watches\n\nladies and gents watches from only num . num inc. delivery ... \n\n url",
  "label": 1,
  "predicted_probability": 0.6034402251243591,
  "predicted_label": 1
},
}
```

Figure 12: Sample of the predictions JSON used for the Scam Shield predictions on the website

While the structure of Practical 1 and Practical 2 are similar, there is one key difference:

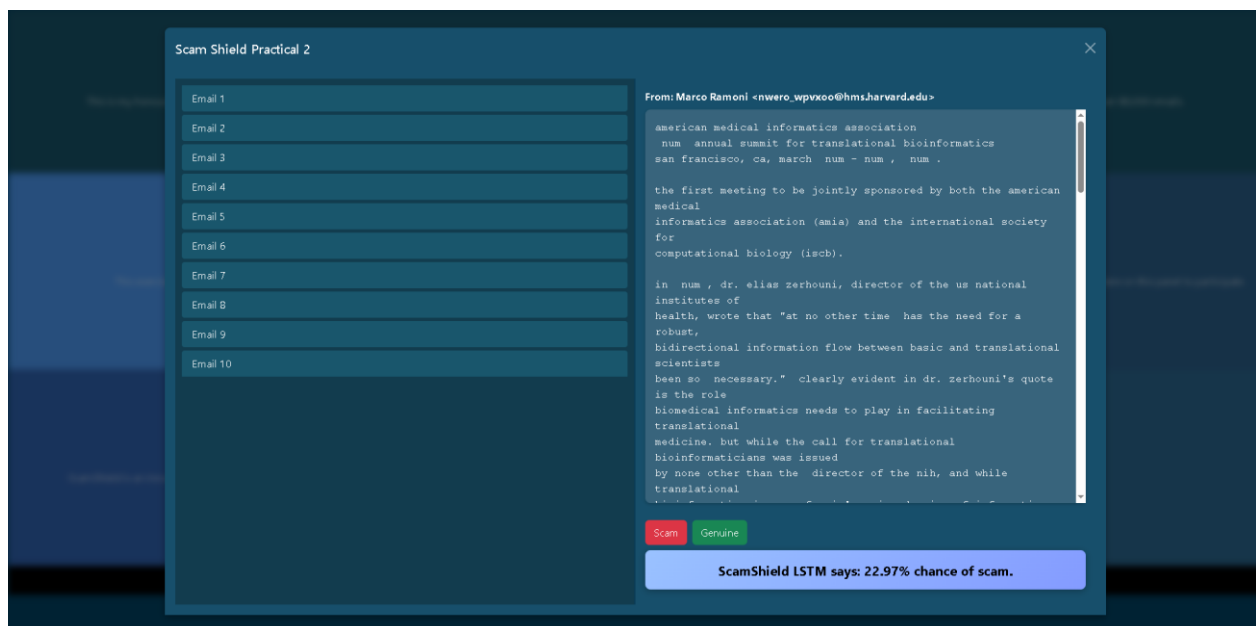


Figure 13: Practical Two modal, with Scam Shield Probability Banner.

The participant will now have ten emails, five labelled as phishing, five labelled as genuine, each of those have their associated Scam Shield probabilities, and labels provided by the JSON. It is important to note that the predictions provided to the website are identical to what would have been produced if the extension was present in the practical.

3.4 Survey

To evaluate the impact and usability of the Scam Shield tool, a survey was conducted involving 14 participants. The study followed ethical guidelines provided by Abertay University for participant consent and non-biased data collection.

3.4.1 Participant Consent and Ethical Considerations

Participants were informed of the nature and purpose of the study prior to participation. Informed consent was obtained digitally via an introductory Microsoft Forms page, which outlined the voluntary nature of the participation, anonymity of responses, and the option to withdraw at any time. No personally identifiable data was collected, and the study was approved in accordance with university ethics procedures.

3.4.2 Survey Design and Structure

The survey was designed to avoid priming effects and bias. To achieve this, participants were not initially told that the study focused on phishing detection (Herr, *et al.*, 1983). To align with Herr's findings, participants were informed that the purpose was to evaluate an AI powered cyber security tool.

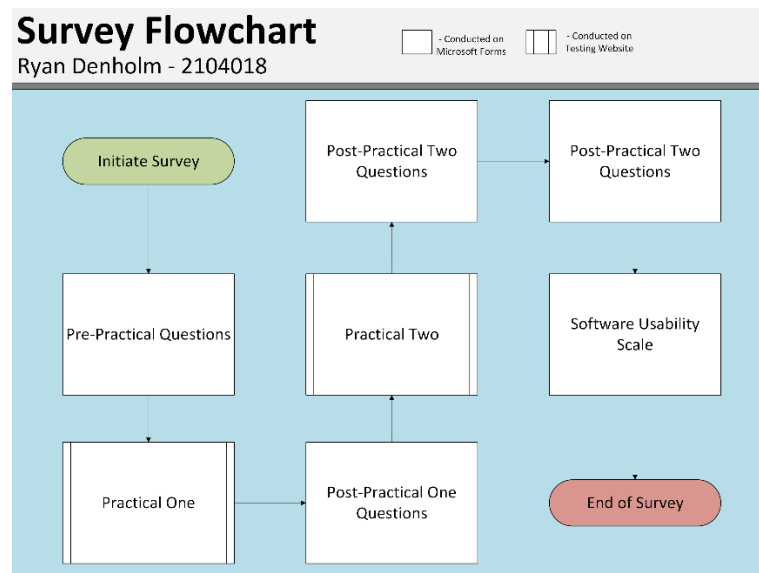


Figure 14: Survey Flowchart

The full survey process (Fig. 14) was split across two platforms to separate questionnaire-based responses from practical exercises. Microsoft Forms was used to collect participant responses for the pre-practical and post-practical questionnaires, which included, phishing awareness and confidence self-assessment, and feedback on the usability of the Scam Shield tool.

Both practical exercises were hosted on a dedicated testing website, where participants interacted with simulated email inboxes. The participant was tasked with predicting if email was legitimate or not, clicking the “Scam” or “Genuine” button, depending on their perception of the email content.

During Practical One, the Scam Shield tool was not present to establish a baseline performance, while during Practical Two, the tool was active, allowing comparison of user's detection accuracy across both conditions.

As part of the post-practical questionnaire, the System Usability Scale (SUS) was used to quantitatively evaluate the user experience and interface design of the Scam Shield tool. SUS is a validated and widely used instrument for assessing usability across a broad range of systems and interfaces (Brooke, 1996). The scale consists of ten items alternating between positive and negative phrasing, rated on a five-point Likert scale from “Strongly Disagree” to “Strongly Agree”. The SUS was chosen due to its simplicity, reliability, and ability to benchmark usability against known industry standards (Bangor, et al., 2008). Results from this scale were used to interpret the perceived usability and overall user satisfaction of Scam Shield following the phishing detection tasks. Please see Appendix E for a list of the questions presented to participants.

3.4.2 Evaluation Strategy

Survey responses were analysed using a repeated-measure design to compare pre- and post-intervention confidence levels for the same participants. The Wilcoxon signed-rank test was applied due to non-normal data distribution (see 3.5.2).

3.5 Statistical Analysis

The statistical analyses of the data produced by these methods were separated into two groups, Model Results and Practical Participant Results.

3.5.1 Model Results

The same model was used to test proficiency across three different training epoch intervals: three, five and ten respectively, these intervals were trained ten times, producing ten replicates for Precision, Recall and F1 Score (for training and validation) per epoch interval.

The Friedman test was performed on all epoch interval validation metrics: Precision, Recall and F1 Score. This rank-based non-parametric test was chosen over a standard one-way ANOVA as the data did not display a Gaussian distribution, as determined by D’Agostino-Pearson normality test (Appendix D.3). Subsequent post hoc multiple comparison tests were run to compare each epoch interval to each other. Dunn’s multiple comparison test was also performed to compare each epoch’s metric against each other (e.g., three epochs, precision compared to five epochs precision). No outliers were excluded; this was determined by a ROUT method (Q = 1%) outlier test. All statistical analyses were performed using Prism 8.0.2 software.

3.5.2 Practical Participant Results

Participants’ Practical One and Two results were obtained through visual observation of participants’ screen recording, noting their responses. Participants’ confidence was self-assessed post Practical One, and again post Practical Two in the survey.

Participants' practical results and confidence data were assessed for normality using the D'Agostino-Pearson normality test. As they did not follow a Gaussian distribution, a non-parametric, rank based Wilcoxon test was conducted.

4. Results

4.1 Model Results

4.1.1 Precision, Recall and F1 Scores Across Epochs (three, five and ten)

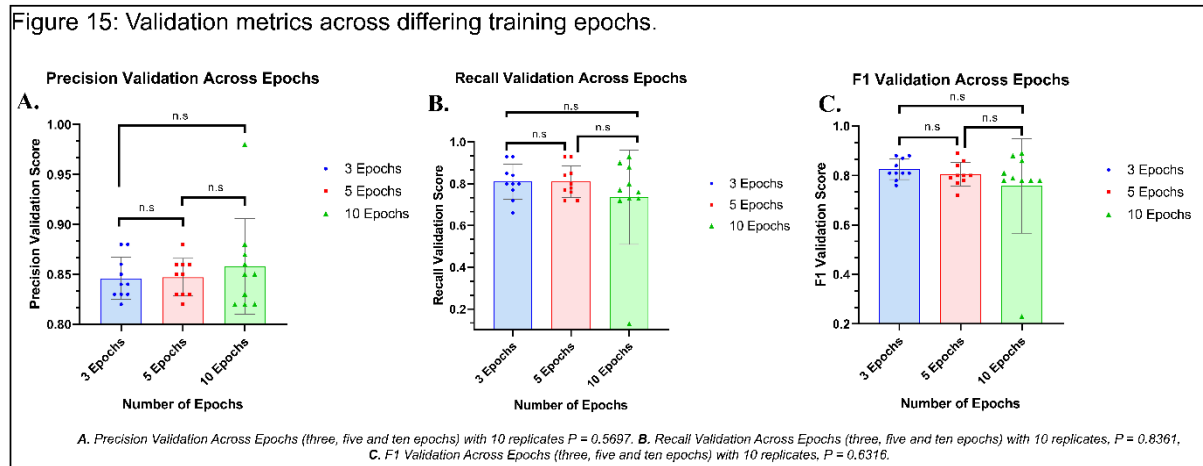


Figure 15: Validation metrics across three, five and ten epochs, plotted as scattered bar-graphs, showing no statistical significance between different training lengths.

Precision, recall and F1 validation metrics were assessed using a Friedman test for repeated measures across three, five and ten epochs. Precision results ($\chi^2(2) = 6.783$, $p = 0.6597$), recall ($\chi^2(2) = 0.4828$, $p = 0.8361$) and F1-Score ($\chi^2(2) = 1.086$, $p = 0.6316$) all indicated no statistically significant differences. Dunn's multiple comparisons test further confirmed no significant differences between any group pairs for all three metrics ($P > 0.9999$).

4.1.2 Confusion Matrix

Scam Shield Model - Confusion Matrix

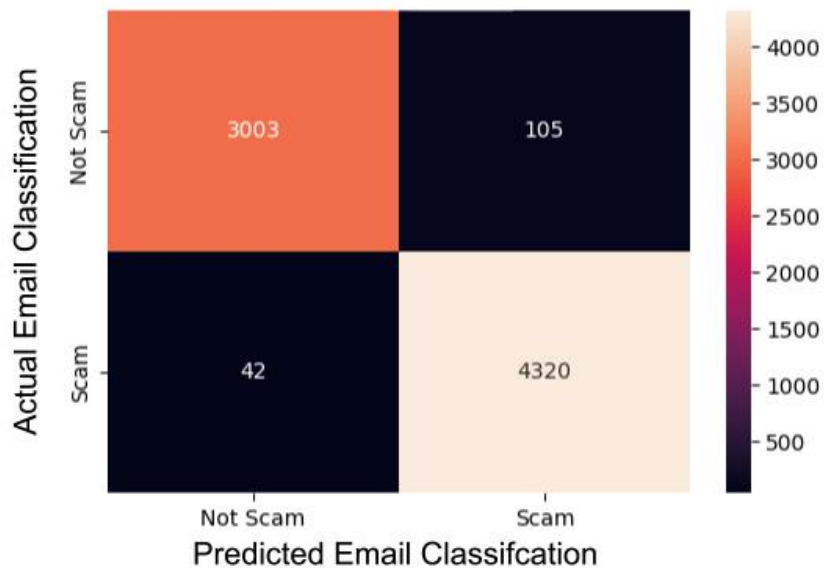


Figure 16: This confusion matrix displays the number of correct and incorrect classifications made by the Scam Shield model on the test dataset. The top-left and bottom-right cells represent correct predictions, the bottom-left and top-right reflect misclassifications.

A confusion matrix was generated to provide a granular view of the model's classification outcomes. As shown in Figure 16, the model correctly identified 3003 legitimate emails and 4320 phishing emails. Misclassifications included 105 false positives, and 42 false negatives.

4.2 Participant Results

4.2.1 Practical Participant Results

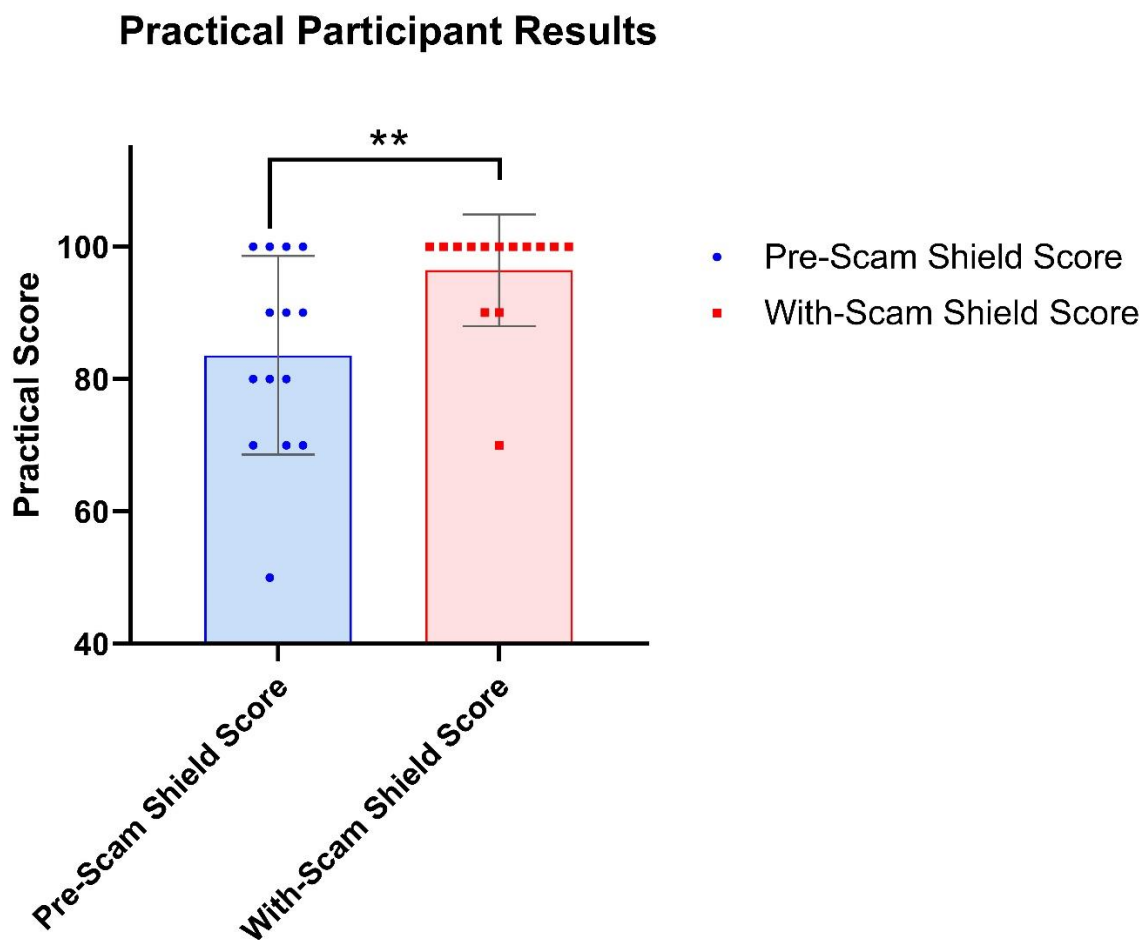


Figure 17: Practical Participant Results, Individual Scores ($N=14$), pair-matched score pre-Scam Shield use and with Scam Shield, represented with mean \pm SD, as determined by Wilcoxon Test, $P = 0.0059$.

The Wilcoxon test ($W = 87$, $n = 14$, $P = 0.0059$) results indicate a statistically significant ($P < 0.05$) increase in scores when using Scam Shield ($P = 0.0059$), with a 10% difference in average participant score pre and with the use of Scam Shield.

4.2.2 User Confidence Results

Participant Confidence Results

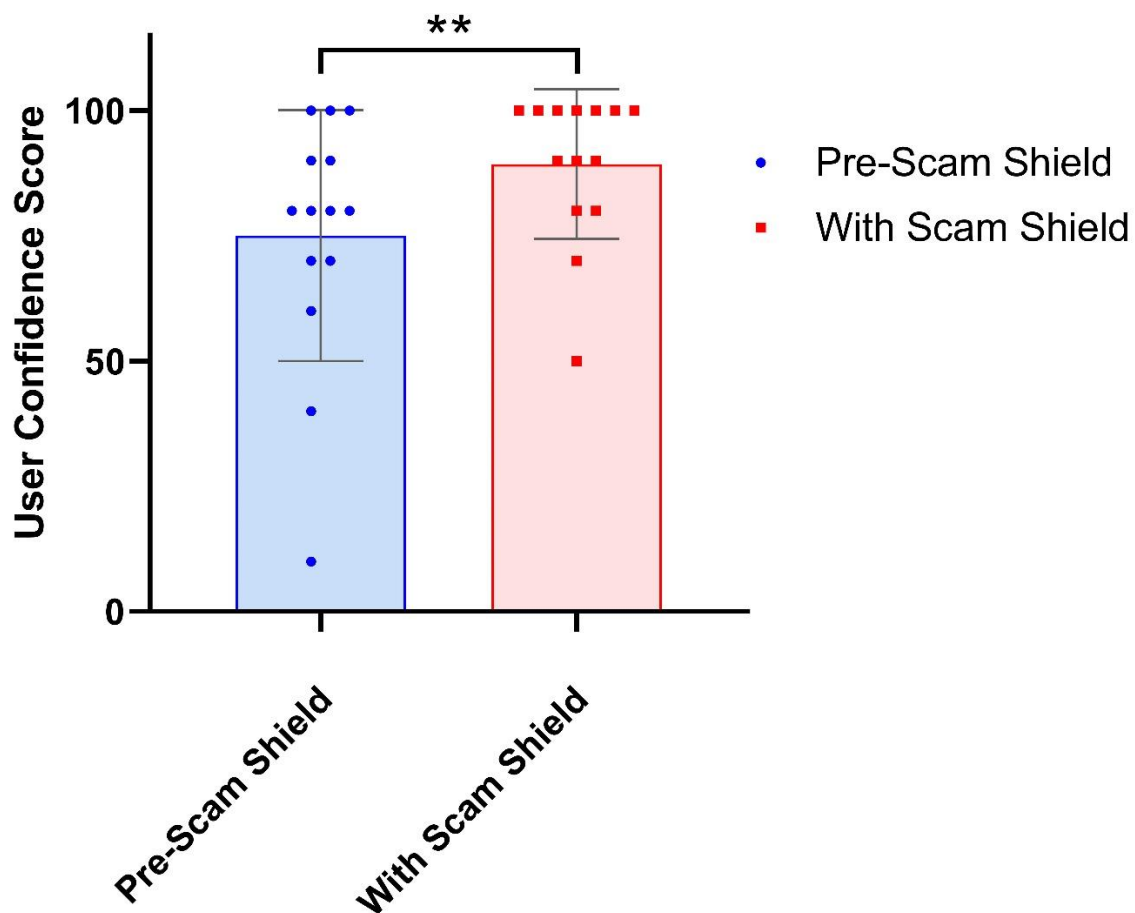


Figure 18: User Confidence Results, Individual Scores ($N=14$), pair-matched score pre-Scam Shield use and with Scam Shield, represented with mean \pm SD, as determined by Wilcoxon Test, $P = 0.0078$.

The Wilcoxon test ($W = 83$, $n = 14$, $P = 0.0078$) revealed a statistically significant ($P < 0.05$) increase in user confidence after using Scam Shield ($P = 0.0078$), with a 15% difference in average participant confidence pre and with the use of Scam Shield.

4.2.3 System Usability Scale (SUS) Results

System Usability Scale (SUS) Item Scores

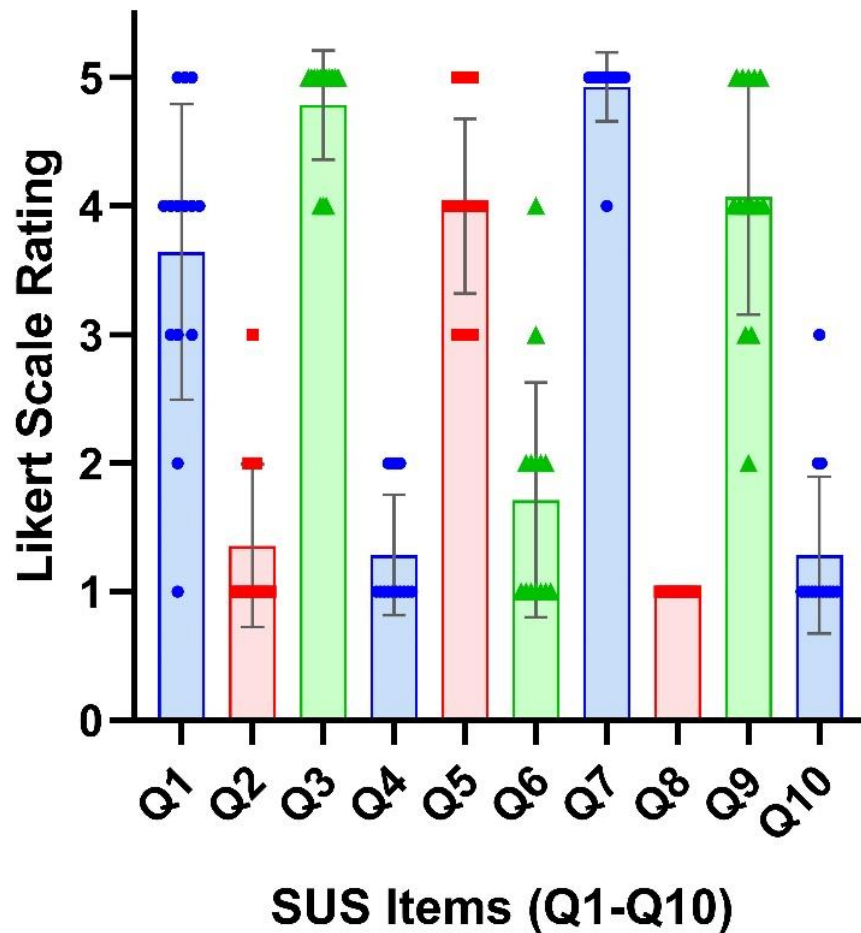


Figure 19: Individual SUS item responses across 14 participants ($N = 14$). Bars represent mean ratings, with error bars = standard deviation. Overlaid symbols show individual participant responses.

Figure 19 shows that participants strongly agreed with positive usability statements (Q3, Q5, Q7) while negative items such as Q2, Q4, Q8 received very low agreement. See Appendix E for total SUS scores per participant.

5. Discussion

The primary objective of this project was to investigate whether a browser-based machine learning model could reliably detect phishing emails in real-time, ultimately helping users identify phishing threats effectively. Phishing remains a persistent cybersecurity challenge due to its reliance on both technical deception and psychological manipulation. The choice of an LSTM architecture for the machine learning model was carefully informed by literature (Alshingiti, et al., 2023). The LSTM's capability to retain contextual information across text sequences makes it well-suited for the nuanced task of phishing detection, where subtle language patterns and structural cues significantly differentiate legitimate emails from deceptive ones. Additionally, its relatively low computational overhead compared to other advanced neural network models, such as Transformer-based architectures, allows for real-time execution within resource-constrained environments like browser extensions, effectively balancing performance accuracy with practical deployability.

5.1 Model Decisions

The decision to train the model using three epochs emerged from empirical testing, explicitly comparing performance metrics from the validation portion of the dataset, such as Precision, Recall, and F1 Score across epochs (three, five and ten). Friedman tests indicated no statistically significant differences in these metrics when epochs were increased beyond three (Precision $P = 0.9999$, Recall $P = 0.8361$, F1 Score $P = 0.6316$). Figure 15 from Results 4.1.1, clearly illustrates these comparisons, highlighting consistent performance across varying epoch lengths. This suggests the model reached convergence early in the training process, effectively capturing the relevant patterns from the dataset within fewer training iterations. Consequently, limiting training to three epochs optimised computational resources, reduced the risk of overfitting, and ensured responsiveness suitable for real-time phishing detection within the constrained environment of a browser-based extension. This increased responsiveness is a direct result of the model's reduced complexity due to fewer training epochs, which in turn leads to faster inference times. Lighter models demand less memory and computational power, making them ideal for implementing through a Firefox extension.

5.1.1 Confusion Matrix Interpretation

The confusion matrix, as shown in Figure 16, further validates the model's performance and complements the F1 score analysis. With a total of 7,470 emails in the validation test set, the model achieved an overall accuracy of 98% ($(3003 + 4320) / [3003 + 4320 + 42 + 105]$). Its precision for identifying phishing emails was 97.6% ($4320 / [4320 + 105]$), and the recall was 99.0% ($4320 / [4320 + 42]$), indicating a high true positive rate. This is particularly important in phishing detection, where false negatives pose significant risks by allowing malicious content to bypass filters.

The false negative rate, though relatively low, still resulted in 42 phishing emails being reported as legitimate. While this may slightly impact user trust, the rate remains acceptable given the tool's role as an assistive aid rather than an automated blocker. These figures confirm that the model effectively balances security sensitivity and practical deployability within a real-time browser context.

5.2 Dataset limitations and Practical User Classification Results

An essential consideration during development was the quality and currency of the training dataset. While the dataset CEAS_08 significantly accelerated development, its dated nature (2008) may limit the model's effectiveness against modern phishing tactics, which increasingly involve advanced social engineering and dynamic content variations. Despite this limitation, the practical evaluation of Scam Shield demonstrated a measurable improvement in phishing detection, a statistically significant increase was observed in mean practical scores from Practical One (without Scam Shield) and Practical Two (with Scam Shield) ($P = 0.0059$), suggesting that the extension meaningfully enhanced users' classification accuracy.

This statistically significant result implies that even with limitations in the training data, Scam Shield meaningfully enhances users' ability to detect phishing emails. Nonetheless, it is critical to acknowledge the relatively small sample size ($N = 14$), which may affect the robustness and generalisability of these findings.

To contextualise these results further, the average FP (Emails classified as "Scam" when the email was genuine) rate found across participants in Practical One was 10% (1 out of the 10 emails displayed), and the average FN (Emails classified as "Genuine" when they are phishing) rate was 20%. This shows that the average participant would misclassify an average of three emails in their practical, with two of them being phishing attempts that they deemed to be genuine. Of course, this leads to the threats associated with phishing attacks.

In contrast, Practical Two saw the average FN rate reduced by 50% (Now 10% as opposed to the 20%). This is a promising reduction in user's being deceived by phishing emails while Scam Shield is present, overall misclassification of emails reduced by 33%. It is important to note, the practical exercises were conducted over 20 total emails, these results may not be representative of long-term use.

FN and FP results for Practical One and Two

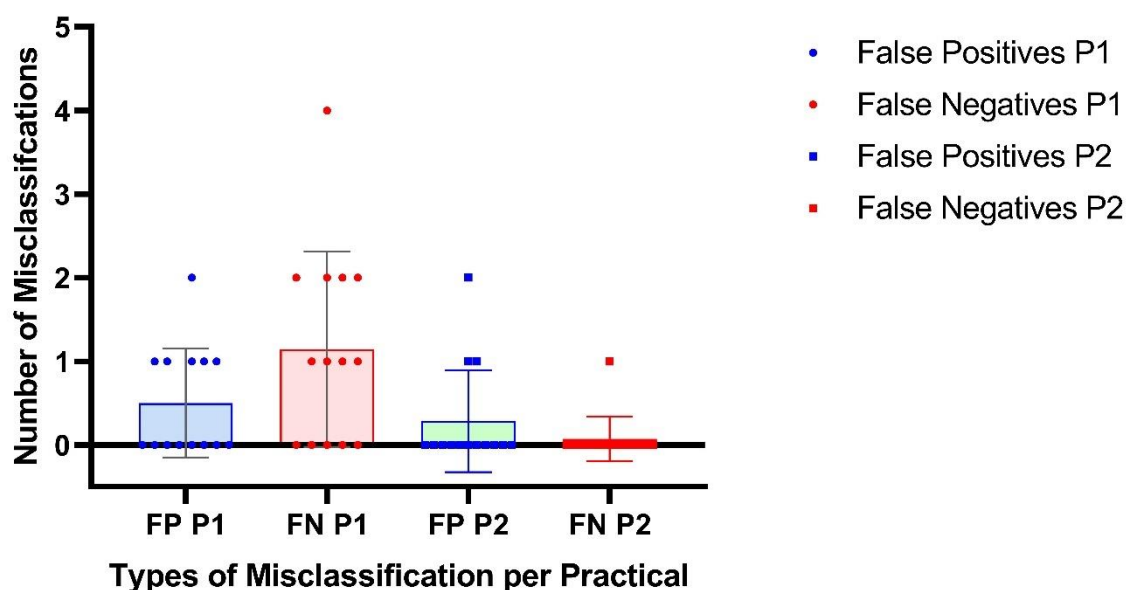


Figure 20: Bar graph showing the number of different misclassifications per practical, $N=14$.

5.3 Practical User Confidence

The interpretability and usability of Scam Shield during Practical Two were central to user performance improvements. In this section, users interacted with simulated emails enhanced by Scam Shield predictions, displayed alongside sender information and email content. This setup resulted in a statistically significant increase in user-reported confidence when identifying phishing emails ($P = 0.0078$), compared to Practical One. Confidence was self-assessed by the participant, reflecting their perceived ability to detect phishing emails. It is important to acknowledge the relatively low sample size ($N = 14$), which may limit the generalisability of this finding. Nonetheless, this result indicates adequate detection capabilities and demonstrates effective user engagement, aligning with current cybersecurity literature emphasising the importance of combining technical accuracy with clear user communication (Nurse, et al., 2011).

It is important to consider the potential influence of the interface on user confidence. Scam Shield presented predictions in a format such as “69.32% chance of scam”, which may have guided participants to trust the model. This aligns with the findings from Bach, et al. (2023), which highlighted how visual cues and perceived transparency contribute substantially to user trust in AI systems, sometimes more than the underlying accuracy itself. However, this form of testing occurred outside the actual extension and relied on a static testing website, potentially affecting user behaviour. The absence of real-time feedback and deeper explanatory features, such as highlighting suspicious text segments, may have created an inflated sense of certainty. Thus, while the model’s effectiveness is supported, further testing within

the extension environment is needed to assess how the true implementation would impact usability and users' trust.

5.3.1 System Usability and Participant Perception

The System Usability Scale (SUS) was used to quantitatively assess the usability of Scam Shield following user interaction in Practical Two. The mean SUS score recorded was 86.96, which significantly exceeds the industry standard benchmark of 68 for acceptable usability and surpasses the 80.3 threshold typically associated with "excellent" systems (Bangor, et al., 2008). These results indicate that participants found Scam Shield intuitive and easy to use, despite its technical functionality and real-time feedback mechanism.

Analysis of individual SUS items showed consistently high scores on positively worded statements, such as Q3 ("I thought the system was easy to use") and Q7 ("I would imagine most people would learn to use this system quickly"), both averaging close to the maximum score of 5. Conversely, negatively phrased items like Q2 ("I found the system unnecessarily complex") and Q8 ("I found the system very cumbersome to use") received minimal agreement, further reinforcing the perception of simplicity and accessibility.

These findings suggest that Scam Shield not only improves objective phishing detection accuracy (as demonstrated in participant scores) but also promotes a positive user experience, which is essential for sustained user engagement and adoption. The high usability score further supports the project's focus on privacy-preserving, user-friendly design, highlighting its potential as a deployable tool beyond the test environment.

It should be noted, however, that SUS responses were based on interaction with the Scam Shield testing website rather than the fully integrated browser extension. While the underlying model and logic were consistent, the website version lacked real-time inference, and certain interface features present in the extension.

5.4 Extension Implementation and Limitations

Scam Shield's current iteration is limited to Outlook, reducing its broad applicability across diverse email clients like Gmail. Furthermore, specific phishing strategies, such as fake invoices or complex social engineering schemes (e.g. romance scams), demonstrated variable detection accuracy, highlighting areas for model enhancement. Despite the LSTM model's lower complexity, particularly given its efficient training over just three epochs, interpretability remains a challenge. Unlike models with attention mechanisms that can highlight influential words or phrases, TensorFlow.js currently lacks support for attention layers, which makes it difficult to highlight specific features in the email (e.g. certain words or sentence structures) are most indicative of phishing. While server-side implementations could accommodate such layers and thus enhance interpretability, they would require transmitting the user's email content to an external server. This introduces significant privacy risks

and undermines the GDPR-aligned principle of data minimisation (Union, 2016). Consequently, Scam Shield opts for a client-side model to prioritise user privacy, an approach further examined in the next section.

5.5 Privacy

Deploying the Scam Shield model directly within the browser offers a critical privacy advantage by ensuring that users' email content is analysed locally and never transmitted to external servers (Panum, et al., 2020). This approach aligns closely with GDPR's principles of data minimisation, which states that personal data collection should be limited to what is strictly necessary for the intended purpose. By keeping all processing on the client side, Scam Shield avoids common vulnerabilities associated with server-based models, such as data interception during transmission, centralised breaches, or unauthorised third-party access.

This privacy-first design not only reinforces user trust but also positions Scam Shield as a responsible tool that respects data privacy. However, an important contradiction occurred during user testing: the use of InspectLet, a third-party analytics tool, to observe participant behaviour. While this enabled the collection of practical results, it temporarily diverged from the project's privacy priority. Although informed consent was obtained, this method still introduced potential risks by recording user behaviour remotely.

5.6 Testing website implementation and limitations

Finally, practical integration limitations required running model predictions locally offline, feeding results into the test website via a JSON file, rather than executing live predictions within the site itself. During development, attempts to integrate real-time model execution into the website encountered technical challenges, most notably, the model returned nearly identical prediction values for all email samples (ranging narrowly from 99.953... to 99.952...), regardless of the email's content. This suggested that either the tokenizer was not functioning correctly in the web context, or the input formatting was not consistent with the training conditions. These issues made it impossible to conduct a meaningful in-browser evaluation of model performance. As a workaround, precomputed predictions were injected into the site through a structured JSON file, allowing the testing process to proceed with consistent results for participants. However, this approach limited the realism of the test environment, as users were not interacting with a live model.

5.7 Summary

Overall, the Scam Shield model successfully demonstrated the feasibility and practical benefits of client-side phishing detection. It combined technically sound performance, facilitated by an efficiently trained LSTM architecture, with significant improvements in user awareness and confidence as shown in the user trials. The implementation effectively addressed challenges of latency and data privacy by

executing entirely within the browser, eliminating the need to send sensitive email data to remote servers. Furthermore, the statistical evaluation of participant performance and confidence provided evidence that Scam Shield meaningfully enhances both phishing detection accuracy and user trust in the tool.

In addition to performance outcomes, Scam Shield was also rated highly in terms of usability, as evidenced by a mean System Usability Scale score of 86.96%, which exceeds the threshold for excellent usability. This suggests that the tool's interface is intuitive and accessible, even to users without technical backgrounds, reinforcing its potential for real-world adoption.

Despite these achievements, several areas for improvement were also revealed. The limitations of the training datasets, in both size and recency, highlight the need for more contemporary, diverse data sources to improve generalisability and relevance to modern phishing techniques. Additionally, the absence of interpretability features due to frontend framework limitations (i.e., lack of support for attention layers in TensorFlow.js) constrains user understanding of why a particular email is classified as phishing. Although this compromise maintains strong data privacy compliance, it sacrifices some degree of transparency.

The model's usability within a single client (Outlook) is currently a big limitation of the Scam Shield extension. Technical issues in integrating live predictions into the practical testing website meant participants interacted with precomputed outputs rather than a fully dynamic system. This workaround, while effective in delivering consistent results, limited the realism of the evaluation environment.

Taken together, the results and methods show that Scam Shield offers a promising foundation for user-friendly, private and intelligent phishing detection within the browser. While the System Usability Scale results reinforce this, it is important to note that these scores were based on interactions within the controlled testing website. Although Scam Shield's predictions were model-driven in both the website and the actual browser extension, the web-based implementation lacked full real-time execution and certain interface refinements present in the extension itself. As such, the high SUS scores likely reflect the usability of the system's underlying logic and visual design but may underestimate the experience that users would encounter in the fully deployed browser extension, which provides a more integrated and polished interaction.

6. Future Work

While the Scam Shield project has demonstrated promising effectiveness in client-side phishing detection, several limitations emerged that present clear opportunities for future improvement. Addressing these will not only enhance the tool's performance but also its trustworthiness, adaptability, and real-world relevance.

One of the most critical limitations concerns the age and representativeness of the training dataset. Although CEAS_08 offered pre-labelled, accessible data to accelerate model development, they are significantly outdated and may not reflect the sophistication of modern phishing tactics. Future work should involve curating and annotating a contemporary dataset that captures current phishing campaigns, potentially by collaborating with cybersecurity organisations or email providers. Additionally, introducing an opt-in user reporting system with Scam Shield could enable users to voluntarily flag suspicious emails. These user-submitted emails could be anonymised and used to retrain the model periodically, helping to maintain relevance and adapt to evolving attack patterns while preserving user privacy.

Another key area of improvement lies in model interpretability. The current LSTM model, although effective, lacks transparency, users are not informed as to why a specific prediction was made. Due to the limitations of TensorFlow.js (which does not currently support attention mechanisms), integrating explainable features such as identifying which words or phrases indicated phishing or not phishing is technically infeasible on the client-side. As a potential solution, a server-side version of Scam Shield could be developed as an opt-in alternative. This version could enhance the interpretability of scans by utilising attention layers to provide visual explanations of which parts of an email contributed most to a phishing prediction. Although this would require transmitting data off-device, strict opt-in consent and robust encryption could ensure ethical and GDPR-compliant data handling.

In terms of usability, Scam Shield is currently only compatible with Microsoft Outlook, which limits its reach. Expanding compatibility to other major email clients such as Gmail would greatly improve accessibility. This would require refining the DOM element targeting and event detection strategies currently used in the extension, as each client structures email content differently.

During user testing, the Scam Shield model predictions were precomputed and injected into a static website, rather than being executed live in the browser. This compromise made due to technical difficulties with tokenizer loading and consistent model input formatting, reduced the realism of the evaluation environment. Future work should focus on resolving in-browser inference issues, ensuring consistent tokenisation and vocabulary matching between training and deployment environments. Testing should also be extended to include live model inference within the test website to better reflect real-world usage.

Finally, although the study showed a statistically significant increase in user performance and confidence with Scam Shield, the sample size of 14 participants limits generalisability. A larger-scale, more diverse study would help validate these initial findings, provide deeper insights into user behaviour, and uncover edge cases where the model may fail. Such studies could also explore long-term effects, whether regular Scam Shield usage helps users become better at detecting phishing emails even without assistance.

By addressing these areas, future versions of Scam Shield can evolve into a more powerful, trustworthy, and scalable anti-phishing solution, capable of continuous learning, greater interpretability, and broader real-world deployment.

7. Conclusion

This project set out to investigate whether a lightweight, client-side machine learning model could effectively detect phishing emails in real time without compromising user privacy. The project successfully developed Scam Shield, a Firefox browser extension that integrates an LSTM classification model using TensorFlow.js. Through extensive evaluation, including model performance analysis and a user study, the project demonstrates that browser-based phishing detection is both feasible and beneficial for improving user resilience against phishing threats.

The primary aims of the project were achieved through developing and training an LSTM model. The LSTM demonstrated robust performance metrics on the training and validation dataset, showing a 98% F1-Score and demonstrating rapid convergence within three training epochs. Integration into a browser environment was achieved without the need for external servers, preserving user privacy and data integrity. Furthermore, the user study with 14 participants confirmed that Scam Shield significantly enhanced user's ability to identify phishing emails, improving both accuracy and confidence levels, with statistically significant results ($P < 0.05$).

These findings reinforce existing literature that advocates for intelligent, adaptive defences against phishing attacks, while offering a novel contribution by demonstrating a fully client-side, deployable solution. Moreover, the results underscore the practical importance of user-centred security tools that not only automate detection but also actively support user education and decision-making.

However, several limitations emerged. The training data, delivered from a collection of older phishing emails, may not fully reflect the rapidly evolving tactics used by modern threat actors. The model's lack of interpretability and explainability features may also limit user trust and understanding of detection decisions. Additionally, the tool's current design supports only a single email client (Outlook), with broader deployment yet to be validated.

Future work should focus on curating more contemporary phishing datasets, incorporating explainability mechanisms into the model's outputs, expanding cross-client compatibility, and conducting larger-scale user studies across diverse populations. Such enhancements would further strengthen the tool's practical value and academic contribution.

In summary, Scam Shield provides a promising proof-of-concept for privacy-preserving, real-time phishing detection at the browser level. The success of this project highlights the potential for client-side machine learning models to augment

user security awareness and autonomy, representing a meaningful step forward in the ongoing battle against phishing threats.

References

- Abadi, M. et al., 2016. *TensorFlow: A system for large-scale machine learning*, Santa Clara: USENIX Association.
- Abuadbbba, A. et al., 2022. Towards Web Phishing Detection Limitations and Mitigation. *arXiv*.
- Alam, A. N. & Khandakar, A., 2024. *Phishing Email Dataset*. [Online]
Available at: <https://www.kaggle.com/datasets/naserabdullahalam/phishing-email-dataset>
[Accessed 4 December 2024].
- Alkhalil, Z., Hewage, C., Nawaf, L. & Khan, I., 2021. Phishing Attacks: A Recent Comprehensive Study and a New Anatomy. *Frontiers of Computer Science*, 09 March. Volume 3.
- Alshingiti, Z. et al., 2023. A Deep Learning-Based Phishing Detection System Using CNN, LSTM, and LSTM-CNN.. *Electronics*, Issue 12, p. 232.
- Atawneh, S. & Aljehani, H., 2023. Phishing Email Detection Model Using Deep Learning. *Electronics*, Issue 12.
- Bach, T. et al., 2023. *A Systematic Literature Review of User Trust in AI-Enabled Systems: An HCI Perspective*, s.l.: arXiv.
- Bangor, A., Kortum, T. P. & Miller, T. J., 2008. An empirical evaluation of the System Usability Scale. *International Journal of Human–Computer Interaction*, 24(6), p. 574–594.
- Brooke, J., 1996. SUS: a “quick and dirty” usability scale. In: P. T. B. W. B. a. M. I. Jordan, ed. *Usability Evaluation in Industry*. London: Taylor & Francis, p. 189–194.
- Brownlee, J., 2021. *Gentle Introduction to the Adam Optimization Algorithm for Deep Learning*. [Online]
Available at: <https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/#:~:text=Adam%20realizes%20the%20benefits%20of,Gradient%20Descent%20Optimization%20From%20Scratch>
[Accessed 12 February 2025].
- Cofense, 2024. *2024 Annual State of Email Security Report*. [Online]
Available at: <https://cofense.com/getmedia/db5a5ad7-b39a-45f5-bab7-eb165b9a0685/2024-cofense-annual-state-of-email-security-report.pdf>

developers, i.-l., 2024. *imbalanced-learn documentation - Version 0.13.0*. [Online]
Available at: <https://imbalanced-learn.org/stable/>
[Accessed 28 January 2025].

Fette, I., Sadeh, N. & Tomasic, A., 2007. Learning to detect phishing emails. *16th International World Wide Web Conference*, pp. 649-656.

GitHub, n.d. *Websites for you and your projects..* [Online]
Available at: <https://pages.github.com/>
[Accessed 15 March 2025].

Herr, M. P., Sherman, J. S. & Fazio, H. R., 1983. On the consequences of priming: Assimilation and contrast effects. *Journal of Experimental Social Psychology*, 19(4), pp. 323-340.

InspectLet, n.d. *Website analytics and session recording*. [Online]
Available at: <https://www.inspectlet.com/>
[Accessed 10 April 2025].

Kingma, P. D. & Ba, J., 2015. *Adam: A Method for Stochastic Optimization*. [Online]
Available at: <https://arxiv.org/abs/1412.6980>
[Accessed 12 February 2025].

Lim, B. et al., 2025. EXPLICATE: Enhancing Phishing Detection through Explainable AI and LLM-Powered Interpretability. *arXiv*, 22 March.

Nurse, J. R. C., Creese, S., Goldsmith, M. & Lamberts, K., 2011. Trustworthy and Effective Communication of Cybersecurity Risks: A Review. *Lecture Notes in Computer Science*, pp. 143-150.

Panum, T., Fogh, K., Varming, C. & Rasmussen, K., 2020. *Towards Adversarial Phishing Detection*. s.l., USENIX Association.

Paszke, A., Gross, S., Massa, F. & al., e., 2019. *PyTorch: An imperative style, high-performance deep learning library*. Vancouver, NeurIPS.

PyTorch, 2024. *PyTorch – An open source machine learning framework*. [Online]
Available at: <https://pytorch.org/>
[Accessed 10 February 2025].

Smilkov, D., Thorat, N., Assogba, Y. & al., e., 2019. *TensorFlow.js: Machine Learning for the Web and Beyond*. [Online]
Available at: <https://arxiv.org/abs/1901.05350>
[Accessed 10 February 2025].

TensorFlow.js, 2024. *TensorFlow.js – Machine Learning for JavaScript Developers*. [Online]

Available at: <https://www.tensorflow.org/js>
[Accessed 10 February 2025].

TensorFlow, 2024. *TensorFlow – An end-to-end open source machine learning platform*.
[Online]
Available at: <https://www.tensorflow.org/>
[Accessed 26 January 2025].

Union, E., 2016. *Article 5 GDPR – Principles relating to processing of personal data*.
[Online]
Available at: <https://gdpr-info.eu/art-5-gdpr/>
[Accessed 22 April 2025].

Appendices

Appendix A: Manifest.json

This file serves as the core configuration file for the Scam Shield browser extension, defining key properties, permissions, and scripts required for the effective and secure execution of the extension within the browser environment.

Manifest Definition

This primary declaration specifies essential metadata about the extension. `manifest_version` defines the schema version that dictates compatibility and functionality standards for the browser. In this case, `manifest_version 2` was set, this was due to compatibility issues brought with implementing V3, Firefox currently supports manifest V2 whereas Chrome does not.

Furthermore, the extension name, version and description are set here.

Permissions Declaration

Permissions are explicitly requested to ensure the extension has necessary access rights for functionality while maintaining security standards and transparency to users.

Interaction and script injection into currently active browser tabs is achieved by requesting “tabs” and “activeTab” and “storage” enables persistent local storage for model data and user preferences. “webRequest” and “webRequestBlocking” facilitates the monitoring, interception and management of HTTP requests, critical for real-time phishing analysis. Finally, URL permissions “http://*/*” and “https://*/*” allow the extension to access and analyse content across all websites for comprehensive phishing detection, this can be refined to specific email client sites.

Appendix B: Background.js

This script is responsible for initialising, loading, and implementing the Scam Shield model, enabling the extension's functionality to predict phishing attempts within the browser in real-time. It utilises TensorFlow.js, a vocabulary tokenizer, and a custom attention layer to improve the predictive accuracy.

Initialisation of Background Environment

Upon activation of the browser extension, the background.js script initiates a background process. This ensures that critical resources such as the machine learning model and tokenizer vocabulary are loaded once and persistently available for the extension's runtime.

```
let vocab = {}.
```

```
let scamShieldModel = null;
```

```
let modelLoaded = false;
```

These variables are set to empty, null, and false respectively to provide simple error checking, ensuring each vital resource is loaded prior to potential predictions occurring.

Summary of background.js

Stage	Function
Initialisation	Configures global variables and respective states.
Tokenizer	Ensures consistent text preprocessing.
Model Loading	Integrates the trained LSTM into the extension.
Text Preprocessing	Standardise model inputs.
Model Prediction	Performs inference for phishing detection.
Messaging and Integration	Bridges the back-end logic and front-end user interface.

Table 2: Stages of model utilisation in background.js

Appendix C: GDPR Research Data Management, Data Sign Off Form




GDPR Research Data Management Data Sign Off Form

For undergraduate or postgraduate student projects supervised by an Abertay staff member.

This form MUST be included in the student's thesis/dissertation. Note that failure to do this will mean that the student's project cannot be assessed/examined.

Part 1: Supervisors to Complete

By signing this form, you are confirming that you have checked and verified your student's data according to the criteria stated below (e.g., raw data, completed questionnaires, superlab/Éprime output, transcriptions etc.)

Student Name:	Ryan Denholm		
Student Number:	2104018		
Lead Supervisor Name:	Marc Kydd		
Lead Supervisor Signature			
Project title:	Scam Shield: Combating Phishing Emails in the Browser.		
Study route:	PhD <input type="checkbox"/>	MbR <input type="checkbox"/>	MPhil <input type="checkbox"/>
	Undergraduate <input checked="" type="checkbox"/>	PhD by Publication <input type="checkbox"/>	

Part 2: Student to Complete

	Initial here to confirm "Yes"
I confirm that I have handed over all manual records from my research project (e.g., consent forms, transcripts) to my supervisor for archiving/storage	Yes
I confirm that I have handed over all digital records from my research project (e.g., recordings, data files) to my supervisor for archiving/storage	Yes
I confirm that I no longer hold any digital records from my research project on any device other than the university network and the only data that I may retain is a copy of an anonymised data file(s) from my research	Yes
I understand that, for undergraduate projects, my supervisor may delete manual/digital records of data if there is no foreseeable use for that data (with the exception of consent forms, which should be retained for 10 years)	Yes

Student signature : Ryan Denholm



GDPR Research Data Management Data Sign Off Form

Date: 26/04/2025

Appendix D: Statistical Analyses Results

D.1 Participants' Practical Results

D.1.1 Normality and Lognormality Results

Normality and Lognormality Tests Tabular results		A	B
		Pre-Scam Shield Score	With-Scam Shield Score
1	Test for normal distribution		
2	Anderson-Darling test		
3	A2*	0.5226	3.126
4	P value	0.1508	<0.0001
5	Passed normality test (alpha=0.05)?	Yes	No
6	P value summary	ns	****
7			
8	D'Agostino & Pearson test		
9	K2	1.689	25.89
10	P value	0.4298	<0.0001
11	Passed normality test (alpha=0.05)?	Yes	No
12	P value summary	ns	****
13			
14	Shapiro-Wilk test		
15	W	0.8973	0.5029
16	P value	0.1029	<0.0001
17	Passed normality test (alpha=0.05)?	Yes	No
18	P value summary	ns	****
19			
20	Kolmogorov-Smirnov test		
21	KS distance	0.1660	0.4500
22	P value	>0.1000	<0.0001
23	Passed normality test (alpha=0.05)?	Yes	No
24	P value summary	ns	****
25			
26	Number of values	14	14

D.1.2 Wilcoxon Test Results

Wilcoxon test Tabular results		
4	vs.	vs.
5	Column A	Pre-Scam Shield Score
6		
7	Wilcoxon matched-pairs signed rank	
8	P value	0.0059
9	Exact or approximate P value?	Exact
10	P value summary	**
11	Significantly different (P < 0.05)?	Yes
12	One- or two-tailed P value?	Two-tailed
13	Sum of positive, negative ranks	93.00, -6.000
14	Sum of signed ranks (W)	87.00
15	Number of pairs	14
16	Number of ties (Pratt's method)	3
17		
18	Median of differences	
19	Median	10.00
20		
21	How effective was the pairing?	
22	rs (Spearman)	0.1838
23	P value (one tailed)	0.2885
24	P value summary	ns
25	Was the pairing significantly effective?	No

D.2 Participants' Confidence Results

D.2.1 Normality and Lognormality Results

Normality and Lognormality Tests Tabular results		A	B
		Pre-Scam Shield	With Scam Shield
1	Test for normal distribution		
2	Anderson-Darling test		
3	A2*	0.7706	1.281
4	P value	0.0341	0.0016
5	Passed normality test (alpha=0.05)?	No	No
6	P value summary	*	**
7			
8	D'Agostino & Pearson test		
9	K2	9.329	10.18
10	P value	0.0094	0.0061
11	Passed normality test (alpha=0.05)?	No	No
12	P value summary	**	**
13			
14	Shapiro-Wilk test		
15	W	0.8470	0.7617
16	P value	0.0202	0.0017
17	Passed normality test (alpha=0.05)?	No	No
18	P value summary	*	**
19			
20	Kolmogorov-Smirnov test		
21	KS distance	0.2220	0.2637
22	P value	0.0598	0.0092
23	Passed normality test (alpha=0.05)?	Yes	No
24	P value summary	ns	**
25			
26	Number of values	14	14

D.2.2 Wilcoxon Test Results

Wilcoxon test Tabular results		
1	Table Analyzed	25.04.20_ParticipantConfidenceData
2		
3	Column B	With Scam Shield
4	vs.	vs.
5	Column A	Pre-Scam Shield
6		
7	Wilcoxon matched-pairs signed rank test	
8	P value	0.0078
9	Exact or approximate P value?	Exact
10	P value summary	**
11	Significantly different (P < 0.05)?	Yes
12	One- or two-tailed P value?	Two-tailed
13	Sum of positive, negative ranks	89.00 , -6.000
14	Sum of signed ranks (W)	83.00
15	Number of pairs	14
16	Number of ties (Pratt's method)	4
17		
18	Median of differences	
19	Median	15.00
20		
21	How effective was the pairing?	
22	rs (Spearman)	0.5759
23	P value (one tailed)	0.0171
24	P value summary	*
25	Was the pairing significantly effective?	Yes

D.3 Model Metric Test Results

D.3.1 Precision Friedman Test Result

Friedman test ANOVA results		
1	Table Analyzed	25.04.20_PrecisionEvalTestData
2		
3	Friedman test	
4	P value	0.6597
5	Exact or approximate P value?	Approximate
6	P value summary	ns
7	Are means signif. different? ($P < 0.05$)	No
8	Number of groups	10
9	Friedman statistic	6.783
10		
11	Data summary	
12	Number of treatments (columns)	10
13	Number of subjects (rows)	3

D.3.2 Recall Friedman Test Result

Friedman test ANOVA results		
1	Table Analyzed	25.04.20_RecallEvalTestData
2		
3	Friedman test	
4	P value	0.8361
5	Exact or approximate P value?	Exact
6	P value summary	ns
7	Are means signif. different? ($P < 0.05$)	No
8	Number of groups	3
9	Friedman statistic	0.4828
10		
11	Data summary	
12	Number of treatments (columns)	3
13	Number of subjects (rows)	10

D.3.3 F1 Friedman Test Result

Friedman test ANOVA results		
1	Table Analyzed	25.04.20_F1EvalTestData
2		
3	Friedman test	
4	P value	0.6316
5	Exact or approximate P value?	Exact
6	P value summary	ns
7	Are means signif. different? (P < 0.05)	No
8	Number of groups	3
9	Friedman statistic	1.086
10		
11	Data summary	
12	Number of treatments (columns)	3
13	Number of subjects (rows)	10

Appendix E: Full SUS Data

	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH
	I think that I would like to use this system frequently.	I found the system unnecessarily complex.	I thought the system was easy to use.	I think that I would need the support of a technical person to be able to use this system.	I found the various functions in this system were well integrated.	I thought there was too much inconsistency in this system.	I would imagine that most people would learn to use this system very quickly.	I found the system very cumbersome to use.	I felt very confident using the system.	I needed to learn a lot of things before I could get going with this system.
1										
2	4	1	5	1	3	1	5	1	4	1
3	5	1	5	1	5	1	5	1	5	1
4	4	2	4	2	4	2	5	1	4	3
5	1	1	5	1	3	3	5	1	3	1
6	4	2	4	2	4	2	4	1	4	2
7	4	2	5	2	4	1	5	1	4	1
8	3	1	5	1	4	4	5	1	3	1
9	2	1	4	1	4	1	5	1	4	2
10	4	1	5	1	5	1	5	1	5	1
11	3	1	5	1	3	2	5	1	2	1
12	4	1	5	1	4	1	5	1	5	1
13	5	1	5	1	5	1	5	1	5	1
14	5	3	5	2	4	2	5	1	5	1
15	3	1	5	1	4	2	5	1	4	1